

- a. Draw arrow diagrams for U , V , and W .
 b. Indicate whether any of the relations U , V , and W are functions.
9. a. Find all relations from $\{0,1\}$ to $\{1\}$.
 b. Find all functions from $\{0,1\}$ to $\{1\}$.
 c. What fraction of the relations from $\{0,1\}$ to $\{1\}$ are functions?
10. Find four relations from $\{a, b\}$ to $\{x, y\}$ that are not functions from $\{a, b\}$ to $\{x, y\}$.
11. Define a relation P from \mathbf{R}^+ to \mathbf{R} as follows: For all real numbers x and y with $x > 0$,

$$(x, y) \in P \text{ means that } x = y^2.$$

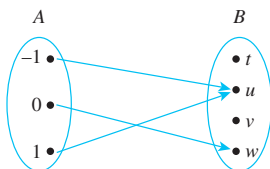
Is P a function? Explain.

12. Define a relation T from \mathbf{R} to \mathbf{R} as follows: For all real numbers x and y ,

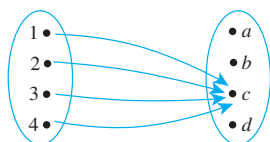
$$(x, y) \in T \text{ means that } y^2 - x^2 = 1.$$

Is T a function? Explain.

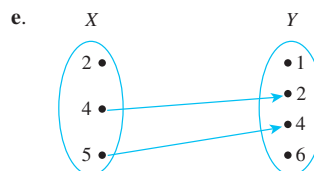
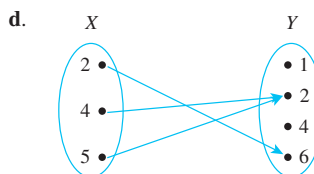
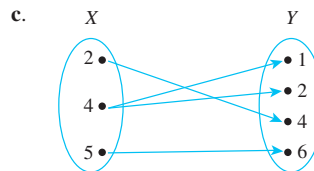
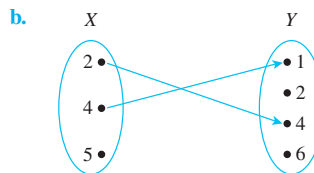
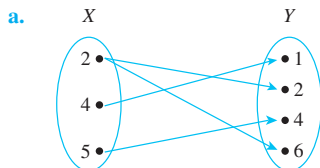
13. Let $A = \{-1, 0, 1\}$ and $B = \{t, u, v, w\}$. Define a function $F: A \rightarrow B$ by the following arrow diagram:



- a. Write the domain and co-domain of F .
 b. Find $F(-1)$, $F(0)$, and $F(1)$.
14. Let $C = \{1, 2, 3, 4\}$ and $D = \{a, b, c, d\}$. Define a function $G: C \rightarrow D$ by the following arrow diagram:



- a. Write the domain and co-domain of G .
 b. Find $G(1)$, $G(2)$, $G(3)$, and $G(4)$.
15. Let $X = \{2, 4, 5\}$ and $Y = \{1, 2, 4, 6\}$. Which of the following arrow diagrams determine functions from X to Y ?



16. Let f be the squaring function defined in Example 1.3.6. Find $f(-1)$, $f(0)$, and $f\left(\frac{1}{2}\right)$.
17. Let g be the successor function defined in Example 1.3.6. Find $g(-1000)$, $g(0)$, and $g(999)$.
18. Let h be the constant function defined in Example 1.3.6. Find $h\left(-\frac{12}{5}\right)$, $h\left(\frac{0}{1}\right)$, and $h\left(\frac{9}{17}\right)$.
19. Define functions f and g from \mathbf{R} to \mathbf{R} by the following formulas: For all $x \in \mathbf{R}$,

$$f(x) = 2x \quad \text{and} \quad g(x) = \frac{2x^3 + 2x}{x^2 + 1}.$$

Does $f = g$? Explain.

20. Define functions H and K from \mathbf{R} to \mathbf{R} by the following formulas: For all $x \in \mathbf{R}$,

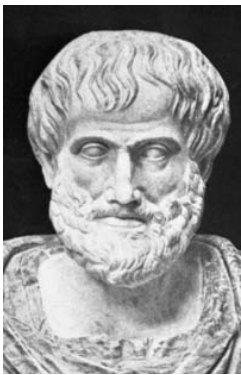
$$H(x) = (x - 2)^2 \quad \text{and} \quad K(x) = (x - 1)(x - 3) + 1.$$

Does $H = K$? Explain.

Answers for Test Yourself

1. a subset of the Cartesian product $A \times B$ 2. a. an element y of B such that $(x, y) \in F$ (i.e., such that x is related to y by F) b. $(x, y) \in F$ and $(x, z) \in F$; $y = z$ 3. the unique element of B that is related to x by F

THE LOGIC OF COMPOUND STATEMENTS



Bettmann/CORBIS

Aristotle
(384 B.C.–322 B.C.)

The first great treatises on logic were written by the Greek philosopher Aristotle. They were a collection of rules for deductive reasoning that were intended to serve as a basis for the study of every branch of knowledge. In the seventeenth century, the German philosopher and mathematician Gottfried Leibniz conceived the idea of using symbols to mechanize the process of deductive reasoning in much the same way that algebraic notation had mechanized the process of reasoning about numbers and their relationships. Leibniz's idea was realized in the nineteenth century by the English mathematicians George Boole and Augustus De Morgan, who founded the modern subject of symbolic logic. With research continuing to the present day, symbolic logic has provided, among other things, the theoretical basis for many areas of computer science such as digital logic circuit design (see Sections 2.4 and 2.5), relational database theory (see Section 8.1), automata theory and computability (see Section 7.4 and Chapter 12), and artificial intelligence (see Sections 3.3, 10.1, and 10.5).

2.1 Logical Form and Logical Equivalence

Logic is a science of the necessary laws of thought, without which no employment of the understanding and the reason takes place. —Immanuel Kant, 1785

The central concept of deductive logic is the concept of argument form. An argument is a sequence of statements aimed at demonstrating the truth of an assertion. The assertion at the end of the sequence is called the *conclusion*, and the preceding statements are called *premises*. To have confidence in the conclusion that you draw from an argument, you must be sure that the premises are acceptable on their own merits or follow from other statements that are known to be true.

In logic, the form of an argument is distinguished from its content. Logical analysis won't help you determine the intrinsic merit of an argument's content, but it will help you analyze an argument's form to determine whether the truth of the conclusion follows *necessarily* from the truth of the premises. For this reason logic is sometimes defined as the science of necessary inference or the science of reasoning.

Consider the following two arguments, for example. Although their content is very different, their logical form is the same. Both arguments are *valid* in the sense that if their premises are true, then their conclusions must also be true. (In Section 2.3 you will learn how to test whether an argument is valid.)

Argument 1 If the program syntax is faulty or if program execution results in division by zero, then the computer will generate an error message. Therefore, if the computer does

not generate an error message, then the program syntax is correct and program execution does not result in division by zero.

Argument 2 If x is a real number such that $x < -2$ or $x > 2$, then $x^2 > 4$. Therefore, if $x^2 \not> 4$, then $x \not< -2$ and $x \not> 2$.

To illustrate the logical form of these arguments, we use letters of the alphabet (such as p , q , and r) to represent the component sentences and the expression “not p ” to refer to the sentence “It is not the case that p .” Then the *common logical form* of both the previous arguments is as follows:

If p or q , then r .

Therefore, if not r , then not p and not q .

Example 2.1.1 Identifying Logical Form

Fill in the blanks below so that argument (b) has the same form as argument (a). Then represent the common form of the arguments using letters to stand for component sentences.

- If Jane is a math major or Jane is a computer science major, then Jane will take Math 150.
Jane is a computer science major.
Therefore, Jane will take Math 150.
- If logic is easy or (1) , then (2) .
I will study hard.
Therefore, I will get an A in this course.

Solution

- I (will) study hard.
- I will get an A in this course.

Common form: If p or q , then r .

q .

Therefore, r .



Statements

Most of the definitions of formal logic have been developed so that they agree with the natural or intuitive logic used by people who have been educated to think clearly and use language carefully. The differences that exist between formal and intuitive logic are necessary to avoid ambiguity and obtain consistency.

In any mathematical theory, new terms are defined by using those that have been previously defined. However, this process has to start somewhere. A few initial terms necessarily remain undefined. In logic, the words *sentence*, *true*, and *false* are the initial undefined terms.

• Definition

A **statement** (or **proposition**) is a sentence that is true or false but not both.

For example, “Two plus two equals four” and “Two plus two equals five” are both statements, the first because it is true and the second because it is false. On the other

hand, the truth or falsity of “He is a college student” depends on the reference for the pronoun *he*. For some values of *he* the sentence is true; for others it is false. If the sentence were preceded by other sentences that made the pronoun’s reference clear, then the sentence would be a statement. Considered on its own, however, the sentence is neither true nor false, and so it is not a statement. We will discuss ways of transforming sentences of this form into statements in Section 3.1.

Similarly, “ $x + y > 0$ ” is not a statement because for some values of x and y the sentence is true, whereas for others it is false. For instance, if $x = 1$ and $y = 2$, the sentence is true; if $x = -1$ and $y = 0$, the sentence is false.

Compound Statements

We now introduce three symbols that are used to build more complicated logical expressions out of simpler ones. The symbol \sim denotes *not*, \wedge denotes *and*, and \vee denotes *or*. Given a statement p , the sentence “ $\sim p$ ” is read “not p ” or “It is not the case that p ” and is called the **negation of p** . In some computer languages the symbol \neg is used in place of \sim . Given another statement q , the sentence “ $p \wedge q$ ” is read “ p and q ” and is called the **conjunction of p and q** . The sentence “ $p \vee q$ ” is read “ p or q ” and is called the **disjunction of p and q** .

In expressions that include the symbol \sim as well as \wedge or \vee , the **order of operations** specifies that \sim is performed first. For instance, $\sim p \wedge q = (\sim p) \wedge q$. In logical expressions, as in ordinary algebraic expressions, the order of operations can be overridden through the use of parentheses. Thus $\sim(p \wedge q)$ represents the negation of the conjunction of p and q . In this, as in most treatments of logic, the symbols \wedge and \vee are considered coequal in order of operation, and an expression such as $p \wedge q \vee r$ is considered ambiguous. This expression must be written as either $(p \wedge q) \vee r$ or $p \wedge (q \vee r)$ to have meaning.

A variety of English words translate into logic as \wedge , \vee , or \sim . For instance, the word *but* translates the same as *and* when it links two independent clauses, as in “Jim is tall but he is not heavy.” Generally, the word *but* is used in place of *and* when the part of the sentence that follows is, in some way, unexpected. Another example involves the words *neither-nor*. When Shakespeare wrote, “Neither a borrower nor a lender be,” he meant, “Do not be a borrower and do not be a lender.” So if p and q are statements, then

p but q	means	p and q
neither p nor q	means	$\sim p$ and $\sim q$.

Example 2.1.2 Translating from English to Symbols: *But* and *Neither-Nor*

Write each of the following sentences symbolically, letting h = “It is hot” and s = “It is sunny.”

- It is not hot but it is sunny.
- It is neither hot nor sunny.

Solution

- The given sentence is equivalent to “It is not hot and it is sunny,” which can be written symbolically as $\sim h \wedge s$.
- To say it is neither hot nor sunny means that it is not hot and it is not sunny. Therefore, the given sentence can be written symbolically as $\sim h \wedge \sim s$. ■

The notation for inequalities involves *and* and *or* statements. For instance, if x , a , and b are particular real numbers, then

$x \leq a$	means	$x < a$	or	$x = a$
$a \leq x \leq b$	means	$a \leq x$	and	$x \leq b$

Note that the inequality $2 \leq x \leq 1$ is not satisfied by any real numbers because

$$2 \leq x \leq 1 \quad \text{means} \quad 2 \leq x \quad \text{and} \quad x \leq 1,$$

and this is false no matter what number x happens to be. By the way, the point of specifying x , a , and b to be *particular* real numbers is to ensure that sentences such as “ $x < a$ ” and “ $x \geq b$ ” are either true or false and hence that they are statements.

Example 2.1.3 And, Or, and Inequalities

Suppose x is a particular real number. Let p , q , and r symbolize “ $0 < x$,” “ $x < 3$,” and “ $x = 3$,” respectively. Write the following inequalities symbolically:

- a. $x \leq 3$ b. $0 < x < 3$ c. $0 < x \leq 3$

Solution

- a. $q \vee r$ b. $p \wedge q$ c. $p \wedge (q \vee r)$ ■

Truth Values

In Examples 2.1.2 and 2.1.3 we built compound sentences out of component statements and the terms *not*, *and*, and *or*. If such sentences are to be statements, however, they must have well-defined **truth values**—they must be either true or false. We now define such compound sentences as statements by specifying their truth values in terms of the statements that compose them.

The negation of a statement is a statement that exactly expresses what it would mean for the statement to be false.

• Definition

If p is a statement variable, the **negation** of p is “not p ” or “It is not the case that p ” and is denoted $\sim p$. It has opposite truth value from p : if p is true, $\sim p$ is false; if p is false, $\sim p$ is true.

The truth values for negation are summarized in a *truth table*.

Truth Table for $\sim p$

p	$\sim p$
T	F
F	T

In ordinary language the sentence “It is hot and it is sunny” is understood to be true when both conditions—being hot and being sunny—are satisfied. If it is hot but not sunny, or sunny but not hot, or neither hot nor sunny, the sentence is understood to be false. The formal definition of truth values for an *and* statement agrees with this general understanding.

• Definition

If p and q are statement variables, the **conjunction** of p and q is “ p and q ,” denoted $p \wedge q$. It is true when, and only when, both p and q are true. If either p or q is false, or if both are false, $p \wedge q$ is false.

The truth values for conjunction can also be summarized in a truth table. The table is obtained by considering the four possible combinations of truth values for p and q . Each combination is displayed in one row of the table; the corresponding truth value for the whole statement is placed in the right-most column of that row. Note that the only row containing a T is the first one since the only way for an *and* statement to be true is for both component statements to be true.

Truth Table for $p \wedge q$

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

By the way, the order of truth values for p and q in the table above is TT, TF, FT, FF. It is not absolutely necessary to write the truth values in this order, although it is customary to do so. We will use this order for all truth tables involving two statement variables. In Example 2.1.5 we will show the standard order for truth tables that involve three statement variables.

In the case of disjunction—statements of the form “ p or q ”—intuitive logic offers two alternative interpretations. In ordinary language *or* is sometimes used in an exclusive sense (p or q but not both) and sometimes in an inclusive sense (p or q or both). A waiter who says you may have “coffee, tea, or milk” uses the word *or* in an exclusive sense: Extra payment is generally required if you want more than one beverage. On the other hand, a waiter who offers “cream or sugar” uses the word *or* in an inclusive sense: You are entitled to both cream and sugar if you wish to have them.

Mathematicians and logicians avoid possible ambiguity about the meaning of the word *or* by understanding it to mean the inclusive “and/or.” The symbol \vee comes from the Latin word *vel*, which means *or* in its inclusive sense. To express the exclusive *or*, the phrase p or q but not both is used.

• Definition

If p and q are statement variables, the **disjunction** of p and q is “ p or q ,” denoted $p \vee q$. It is true when either p is true, or q is true, or both p and q are true; it is false only when both p and q are false.

Note The statement “ $2 \leq 2$ ” means that 2 is less than 2 or 2 equals 2. It is true because $2 = 2$.

Here is the truth table for disjunction:

Truth Table for $p \vee q$

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Evaluating the Truth of More General Compound Statements

Now that truth values have been assigned to $\sim p$, $p \wedge q$, and $p \vee q$, consider the question of assigning truth values to more complicated expressions such as $\sim p \vee q$, $(p \vee q) \wedge \sim(p \wedge q)$, and $(p \wedge q) \vee r$. Such expressions are called *statement forms* (or *propositional forms*). The close relationship between statement forms and *Boolean expressions* is discussed in Section 2.4.

• Definition

A **statement form** (or **propositional form**) is an expression made up of statement variables (such as p , q , and r) and logical connectives (such as \sim , \wedge , and \vee) that becomes a statement when actual statements are substituted for the component statement variables. The **truth table** for a given statement form displays the truth values that correspond to all possible combinations of truth values for its component statement variables.

To compute the truth values for a statement form, follow rules similar to those used to evaluate algebraic expressions. For each combination of truth values for the statement variables, first evaluate the expressions within the innermost parentheses, then evaluate the expressions within the next innermost set of parentheses, and so forth until you have the truth values for the complete expression.

Example 2.1.4 Truth Table for Exclusive Or

Construct the truth table for the statement form $(p \vee q) \wedge \sim(p \wedge q)$. Note that when *or* is used in its exclusive sense, the statement “ p or q ” means “ p or q but not both” or “ p or q and not both p and q ,” which translates into symbols as $(p \vee q) \wedge \sim(p \wedge q)$. This is sometimes abbreviated $p \oplus q$ or $p \text{ XOR } q$.

Solution Set up columns labeled p , q , $p \vee q$, $p \wedge q$, $\sim(p \wedge q)$, and $(p \vee q) \wedge \sim(p \wedge q)$. Fill in the p and q columns with all the logically possible combinations of T's and F's. Then use the truth tables for \vee and \wedge to fill in the $p \vee q$ and $p \wedge q$ columns with the appropriate truth values. Next fill in the $\sim(p \wedge q)$ column by taking the opposites of the truth values for $p \wedge q$. For example, the entry for $\sim(p \wedge q)$ in the first row is F because in the first row the truth value of $p \wedge q$ is T. Finally, fill in the $(p \vee q) \wedge \sim(p \wedge q)$ column by considering the truth table for an *and* statement together with the computed truth values for $p \vee q$ and $\sim(p \wedge q)$. For example, the entry in the first row is F because the entry for $p \vee q$ is T, the entry for $\sim(p \wedge q)$ is F, and an *and* statement is false unless both components are true. The entry in the second row is T because both components are true in this row.

Truth Table for *Exclusive Or*: $(p \vee q) \wedge \sim(p \wedge q)$

p	q	$p \vee q$	$p \wedge q$	$\sim(p \wedge q)$	$(p \vee q) \wedge \sim(p \wedge q)$
T	T	T	T	F	F
T	F	T	F	T	T
F	T	T	F	T	T
F	F	F	F	T	F

Example 2.1.5 Truth Table for $(p \wedge q) \vee \sim r$

Construct a truth table for the statement form $(p \wedge q) \vee \sim r$.

Solution Make columns headed p , q , r , $p \wedge q$, $\sim r$, and $(p \wedge q) \vee \sim r$. Enter the eight logically possible combinations of truth values for p , q , and r in the three left-most columns. Then fill in the truth values for $p \wedge q$ and for $\sim r$. Complete the table by considering the truth values for $(p \wedge q)$ and for $\sim r$ and the definition of an *or* statement. Since an *or* statement is false only when both components are false, the only rows in which the entry is F are the third, fifth, and seventh rows because those are the only rows in which the expressions $p \wedge q$ and $\sim r$ are both false. The entry for all the other rows is T.

p	q	r	$p \wedge q$	$\sim r$	$(p \wedge q) \vee \sim r$
T	T	T	T	F	T
T	T	F	T	T	T
T	F	T	F	F	F
T	F	F	F	T	T
F	T	T	F	F	F
F	T	F	F	T	T
F	F	T	F	F	F
F	F	F	F	T	T

The essential point about assigning truth values to compound statements is that it allows you—using logic alone—to judge the truth of a compound statement on the basis of your knowledge of the truth of its component parts. Logic does not help you determine the truth or falsity of the component statements. Rather, logic helps link these separate pieces of information together into a coherent whole.

Logical Equivalence

The statements

6 is greater than 2 and 2 is less than 6

are two different ways of saying the same thing. Why? Because of the definition of the phrases *greater than* and *less than*. By contrast, although the statements

(1) Dogs bark and cats meow and (2) Cats meow and dogs bark

are also two different ways of saying the same thing, the reason has nothing to do with the definition of the words. It has to do with the logical form of the statements. Any two statements whose logical forms are related in the same way as (1) and (2) would either both be true or both be false. You can see this by examining the following truth table, where the statement variables p and q are substituted for the component statements “Dogs bark” and “Cats meow,” respectively. The table shows that for each combination of truth values for p and q , $p \wedge q$ is true when, and only when, $q \wedge p$ is true. In such a case, the statement forms are called *logically equivalent*, and we say that (1) and (2) are *logically equivalent statements*.

p	q	$p \wedge q$	$q \wedge p$
T	T	T	T
T	F	F	F
F	T	F	F
F	F	F	F

↑ ↑
 $p \wedge q$ and $q \wedge p$ always
 have the same truth
 values, so they are
 logically equivalent

• Definition

Two *statement forms* are called **logically equivalent** if, and only if, they have identical truth values for each possible substitution of statements for their statement variables. The logical equivalence of statement forms P and Q is denoted by writing $P \equiv Q$.

Two *statements* are called **logically equivalent** if, and only if, they have logically equivalent forms when identical component statement variables are used to replace identical component statements.

Testing Whether Two Statement Forms P and Q Are Logically Equivalent

- Construct a truth table with one column for the truth values of P and another column for the truth values of Q .
- Check each combination of truth values of the statement variables to see whether the truth value of P is the same as the truth value of Q .
 - If in each row the truth value of P is the same as the truth value of Q , then P and Q are logically equivalent.
 - If in some row P has a different truth value from Q , then P and Q are not logically equivalent.

Example 2.1.6 Double Negative Property: $\sim(\sim p) \equiv p$

Construct a truth table to show that the negation of the negation of a statement is logically equivalent to the statement, annotating the table with a sentence of explanation.

Solution

p	$\sim p$	$\sim(\sim p)$
T	F	T
F	T	F

p and $\sim(\sim p)$ always have the same truth values, so they are logically equivalent

There are two ways to show that statement forms P and Q are *not* logically equivalent. As indicated previously, one is to use a truth table to find rows for which their truth values differ. The other way is to find concrete statements for each of the two forms, one of which is true and the other of which is false. The next example illustrates both of these ways.

Example 2.1.7 Showing Nonequivalence

Show that the statement forms $\sim(p \wedge q)$ and $\sim p \wedge \sim q$ are not logically equivalent.

Solution

- a. This method uses a truth table annotated with a sentence of explanation.

p	q	$\sim p$	$\sim q$	$p \wedge q$	$\sim(p \wedge q)$	$\sim p \wedge \sim q$
T	T	F	F	T	F	F
T	F	F	T	F	T	F
F	T	T	F	F	T	F
F	F	T	T	F	T	T

$\sim(p \wedge q)$ and $\sim p \wedge \sim q$ have different truth values in rows 2 and 3, so they are not logically equivalent

- b. This method uses an example to show that $\sim(p \wedge q)$ and $\sim p \wedge \sim q$ are not logically equivalent. Let p be the statement “ $0 < 1$ ” and let q be the statement “ $1 < 0$.” Then

$\sim(p \wedge q)$ is “It is not the case that both $0 < 1$ and $1 < 0$,”

which is true. On the other hand,

$$\sim p \wedge \sim q \quad \text{is} \quad "0 \not\leq 1 \quad \text{and} \quad 1 \not\leq 0,"$$

which is false. This example shows that there are concrete statements you can substitute for p and q to make one of the statement forms true and the other false. Therefore, the statement forms are not logically equivalent. ■

Example 2.1.8 Negations of *And* and *Or*: De Morgan's Laws

For the statement “John is tall and Jim is redheaded” to be true, both components must be true. So for the statement to be false, one or both components must be false. Thus the negation can be written as “John is not tall or Jim is not redheaded.” In general, the negation of the conjunction of two statements is logically equivalent to the disjunction of their negations. That is, statements of the forms $\sim(p \wedge q)$ and $\sim p \vee \sim q$ are logically equivalent. Check this using truth tables.

Solution

p	q	$\sim p$	$\sim q$	$p \wedge q$	$\sim(p \wedge q)$	$\sim p \vee \sim q$
T	T	F	F	T	F	F
T	F	F	T	F	T	T
F	T	T	F	F	T	T
F	F	T	T	F	T	T

\uparrow \uparrow
 $\sim(p \wedge q)$ and $\sim p \vee \sim q$ always
 have the same truth values, so they
 are logically equivalent



Augustus De Morgan
(1806–1871)

Symbolically,

$$\sim(p \wedge q) \equiv \sim p \vee \sim q.$$

In the exercises at the end of this section you are asked to show the analogous law that the negation of the disjunction of two statements is logically equivalent to the conjunction of their negations:

$$\sim(p \vee q) \equiv \sim p \wedge \sim q.$$

The two logical equivalences of Example 2.1.8 are known as **De Morgan's laws** of logic in honor of Augustus De Morgan, who was the first to state them in formal mathematical terms.

De Morgan's Laws

The negation of an *and* statement is logically equivalent to the *or* statement in which each component is negated.

The negation of an *or* statement is logically equivalent to the *and* statement in which each component is negated.

Example 2.1.9 Applying De Morgan's Laws

Write negations for each of the following statements:

- John is 6 feet tall and he weighs at least 200 pounds.
- The bus was late or Tom's watch was slow.

Solution

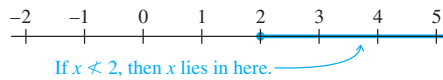
- a. John is not 6 feet tall or he weighs less than 200 pounds.
 b. The bus was not late and Tom's watch was not slow.

Since the statement “neither p nor q ” means the same as “ $\sim p$ and $\sim q$,” an alternative answer for (b) is “Neither was the bus late nor was Tom's watch slow.” ■

If x is a particular real number, saying that x is not less than 2 ($x \not< 2$) means that x does not lie to the left of 2 on the number line. This is equivalent to saying that either $x = 2$ or x lies to the right of 2 on the number line ($x = 2$ or $x > 2$). Hence,

$$x \not< 2 \text{ is equivalent to } x \geq 2.$$

Pictorially,



Similarly,

$$\begin{aligned} x \not> 2 & \text{ is equivalent to } x \leq 2, \\ x \not\leq 2 & \text{ is equivalent to } x > 2, \text{ and} \\ x \not\geq 2 & \text{ is equivalent to } x < 2. \end{aligned}$$

Example 2.1.10 Inequalities and De Morgan's Laws

Use De Morgan's laws to write the negation of $-1 < x \leq 4$.

Solution The given statement is equivalent to

$$-1 < x \text{ and } x \leq 4.$$

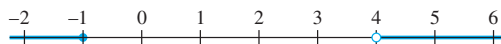
By De Morgan's laws, the negation is

$$-1 \not< x \text{ or } x \not\leq 4,$$

which is equivalent to

$$-1 \geq x \text{ or } x > 4.$$

Pictorially, if $-1 \geq x$ or $x > 4$, then x lies in the shaded region of the number line, as shown below.



Caution! The negation of $-1 < x \leq 4$ is *not* $-1 \not< x \not\leq 4$. It is also not $-1 \geq x > 4$.

De Morgan's laws are frequently used in writing computer programs. For instance, suppose you want your program to delete all files modified outside a certain range of dates, say from date 1 through date 2 inclusive. You would use the fact that

$$\sim(\text{date1} \leq \text{file_modification_date} \leq \text{date2})$$

is equivalent to

$$(\text{file_modification_date} < \text{date1}) \quad \text{or} \quad (\text{date2} < \text{file_modification_date}).$$

Example 2.1.11 A Cautionary Example

According to De Morgan's laws, the negation of

p : Jim is tall and Jim is thin

is

$\sim p$: Jim is not tall or Jim is not thin

because the negation of an *and* statement is the *or* statement in which the two components are negated.

Unfortunately, a potentially confusing aspect of the English language can arise when you are taking negations of this kind. Note that statement p can be written more compactly as

p' : Jim is tall and thin.

When it is so written, another way to negate it is

$\sim(p')$: Jim is not tall and thin.

But in this form the negation looks like an *and* statement. Doesn't that violate De Morgan's laws?

Actually no violation occurs. The reason is that in formal logic the words *and* and *or* are allowed only between complete statements, not between sentence fragments.

One lesson to be learned from this example is that when you apply De Morgan's laws, you must have complete statements on either side of each *and* and on either side of each *or*.



Caution! Although the laws of logic are extremely useful, they should be used as an *aid* to thinking, not as a mechanical substitute for it.

Tautologies and Contradictions

It has been said that all of mathematics reduces to tautologies. Although this is formally true, most working mathematicians think of their subject as having substance as well as form. Nonetheless, an intuitive grasp of basic logical tautologies is part of the equipment of anyone who reasons with mathematics.

• Definition

A **tautology** is a statement form that is always true regardless of the truth values of the individual statements substituted for its statement variables. A statement whose form is a tautology is a **tautological statement**.

A **contradiction** is a statement form that is always false regardless of the truth values of the individual statements substituted for its statement variables. A statement whose form is a contradiction is a **contradictory statement**.

According to this definition, the truth of a tautological statement and the falsity of a contradictory statement are due to the logical structure of the statements themselves and are independent of the meanings of the statements.

Example 2.1.12 Tautologies and Contradictions

Show that the statement form $p \vee \sim p$ is a tautology and that the statement form $p \wedge \sim p$ is a contradiction.

Solution

p	$\sim p$	$p \vee \sim p$	$p \wedge \sim p$
T	F	T	F
F	T	T	F

\uparrow \uparrow
 all T's so all F's so
 $p \vee \sim p$ is $p \wedge \sim p$ is a
 a tautology contradiction

Example 2.1.13 Logical Equivalence Involving Tautologies and Contradictions

If **t** is a tautology and **c** is a contradiction, show that $p \wedge \mathbf{t} \equiv p$ and $p \wedge \mathbf{c} \equiv \mathbf{c}$.

Solution

p	t	$p \wedge \mathbf{t}$	p	c	$p \wedge \mathbf{c}$
T	T	T	T	F	F
F	T	F	F	F	F

\uparrow \uparrow
 same truth same truth
 values, so values, so
 $p \wedge \mathbf{t} \equiv p$ $p \wedge \mathbf{c} \equiv \mathbf{c}$

Summary of Logical Equivalences

Knowledge of logically equivalent statements is very useful for constructing arguments. It often happens that it is difficult to see how a conclusion follows from one form of a statement, whereas it is easy to see how it follows from a logically equivalent form of the statement. A number of logical equivalences are summarized in Theorem 2.1.1 for future reference.

Theorem 2.1.1 Logical Equivalences

Given any statement variables p, q , and r , a tautology **t** and a contradiction **c**, the following logical equivalences hold.

- | | | |
|----------------------------------|---|---|
| 1. <i>Commutative laws:</i> | $p \wedge q \equiv q \wedge p$ | $p \vee q \equiv q \vee p$ |
| 2. <i>Associative laws:</i> | $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ | $(p \vee q) \vee r \equiv p \vee (q \vee r)$ |
| 3. <i>Distributive laws:</i> | $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ | $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ |
| 4. <i>Identity laws:</i> | $p \wedge \mathbf{t} \equiv p$ | $p \vee \mathbf{c} \equiv p$ |
| 5. <i>Negation laws:</i> | $p \vee \sim p \equiv \mathbf{t}$ | $p \wedge \sim p \equiv \mathbf{c}$ |
| 6. <i>Double negative law:</i> | $\sim(\sim p) \equiv p$ | |
| 7. <i>Idempotent laws:</i> | $p \wedge p \equiv p$ | $p \vee p \equiv p$ |
| 8. <i>Universal bound laws:</i> | $p \vee \mathbf{t} \equiv \mathbf{t}$ | $p \wedge \mathbf{c} \equiv \mathbf{c}$ |
| 9. <i>De Morgan's laws:</i> | $\sim(p \wedge q) \equiv \sim p \vee \sim q$ | $\sim(p \vee q) \equiv \sim p \wedge \sim q$ |
| 10. <i>Absorption laws:</i> | $p \vee (p \wedge q) \equiv p$ | $p \wedge (p \vee q) \equiv p$ |
| 11. <i>Negations of t and c:</i> | $\sim \mathbf{t} \equiv \mathbf{c}$ | $\sim \mathbf{c} \equiv \mathbf{t}$ |

The proofs of laws 4 and 6, the first parts of laws 1 and 5, and the second part of law 9 have already been given as examples in the text. Proofs of the other parts of the theorem are left as exercises. In fact, it can be shown that the first five laws of Theorem 2.1.1 form a core from which the other laws can be derived. The first five laws are the axioms for a mathematical structure known as a Boolean algebra, which is discussed in Section 6.4.

The equivalences of Theorem 2.1.1 are general laws of thought that occur in all areas of human endeavor. They can also be used in a formal way to rewrite complicated statement forms more simply.

Example 2.1.14 Simplifying Statement Forms

Use Theorem 2.1.1 to verify the logical equivalence

$$\sim(\sim p \wedge q) \wedge (p \vee q) \equiv p.$$

Solution Use the laws of Theorem 2.1.1 to replace sections of the statement form on the left by logically equivalent expressions. Each time you do this, you obtain a logically equivalent statement form. Continue making replacements until you obtain the statement form on the right.

$$\begin{aligned} \sim(\sim p \wedge q) \wedge (p \vee q) &\equiv (\sim(\sim p) \vee \sim q) \wedge (p \vee q) && \text{by De Morgan's laws} \\ &\equiv (p \vee \sim q) \wedge (p \vee q) && \text{by the double negative law} \\ &\equiv p \vee (\sim q \wedge q) && \text{by the distributive law} \\ &\equiv p \vee (q \wedge \sim q) && \text{by the commutative law for } \wedge \\ &\equiv p \vee \mathbf{c} && \text{by the negation law} \\ &\equiv p && \text{by the identity law.} \quad \blacksquare \end{aligned}$$

Skill in simplifying statement forms is useful in constructing logically efficient computer programs and in designing digital logic circuits.

Although the properties in Theorem 2.1.1 can be used to prove the logical equivalence of two statement forms, they cannot be used to prove that statement forms are not logically equivalent. On the other hand, truth tables can always be used to determine both equivalence and nonequivalence, and truth tables are easy to program on a computer. When truth tables are used, however, checking for equivalence always requires 2^n steps, where n is the number of variables. Sometimes you can quickly see that two statement forms are equivalent by Theorem 2.1.1, whereas it would take quite a bit of calculating to show their equivalence using truth tables. For instance, it follows immediately from the associative law for \wedge that $p \wedge (\sim q \wedge \sim r) \equiv (p \wedge \sim q) \wedge \sim r$, whereas a truth table verification requires constructing a table with eight rows.

Test Yourself

Answers to Test Yourself questions are located at the end of each section.

1. An *and* statement is true if, and only if, both components are ____.
2. An *or* statement is false if, and only if, both components are ____.
3. Two statement forms are logically equivalent if, and only if, they always have ____.
4. De Morgan's laws say (1) that the negation of an *and* statement is logically equivalent to the ____ statement in which each component is ____, and (2) that the negation of an *or* statement is logically equivalent to the ____ statement in which each component is ____.
5. A tautology is a statement that is always ____.
6. A contradiction is a statement that is always ____.

Exercise Set 2.1*

In each of 1–4 represent the common form of each argument using letters to stand for component sentences, and fill in the blanks so that the argument in part (b) has the same logical form as the argument in part (a).

1. a. If all integers are rational, then the number 1 is rational.
All integers are rational.
Therefore, the number 1 is rational.
b. If all algebraic expressions can be written in prefix notation, then _____.
_____.
Therefore, $(a + 2b)(a^2 - b)$ can be written in prefix notation.
2. a. If all computer programs contain errors, then this program contains an error.
This program does not contain an error.
Therefore, it is not the case that all computer programs contain errors.
b. If _____, then _____.
2 is not odd.
Therefore, it is not the case that all prime numbers are odd.
3. a. This number is even or this number is odd.
This number is not even.
Therefore, this number is odd.
b. _____ or logic is confusing.
My mind is not shot.
Therefore, _____.
4. a. If n is divisible by 6, then n is divisible by 3.
If n is divisible by 3, then the sum of the digits of n is divisible by 3.
Therefore, if n is divisible by 6, then the sum of the digits of n is divisible by 3.
(Assume that n is a particular, fixed integer.)
b. If this function is _____ then this function is differentiable.
If this function is _____ then this function is continuous.
Therefore, if this function is a polynomial, then this function _____.
5. Indicate which of the following sentences are statements.
 - a. 1,024 is the smallest four-digit number that is a perfect square.
 - b. She is a mathematics major.
 - c. $128 = 2^6$
 - d. $x = 2^6$

Write the statements in 6–9 in symbolic form using the symbols \sim , \vee , and \wedge and the indicated letters to represent component statements.

6. Let s = “stocks are increasing” and i = “interest rates are steady.”

- a. Stocks are increasing but interest rates are steady.
 - b. Neither are stocks increasing nor are interest rates steady.
7. Juan is a math major but not a computer science major.
(m = “Juan is a math major,” c = “Juan is a computer science major”)
 8. Let h = “John is healthy,” w = “John is wealthy,” and s = “John is wise.”
 - a. John is healthy and wealthy but not wise.
 - b. John is not wealthy but he is healthy and wise.
 - c. John is neither healthy, wealthy, nor wise.
 - d. John is neither wealthy nor wise, but he is healthy.
 - e. John is wealthy, but he is not both healthy and wise.
 9. Either this polynomial has degree 2 or it has degree 3 but not both. (n = “This polynomial has degree 2,” k = “This polynomial has degree 3”)
 10. Let p be the statement “DATAENDFLAG is off,” q the statement “ERROR equals 0,” and r the statement “SUM is less than 1,000.” Express the following sentences in symbolic notation.
 - a. DATAENDFLAG is off, ERROR equals 0, and SUM is less than 1,000.
 - b. DATAENDFLAG is off but ERROR is not equal to 0.
 - c. DATAENDFLAG is off; however, ERROR is not 0 or SUM is greater than or equal to 1,000.
 - d. DATAENDFLAG is on and ERROR equals 0 but SUM is greater than or equal to 1,000.
 - e. Either DATAENDFLAG is on or it is the case that both ERROR equals 0 and SUM is less than 1,000.
 11. In the following sentence, is the word *or* used in its inclusive or exclusive sense? A team wins the playoffs if it wins two games in a row or a total of *three* games.

Write truth tables for the statement forms in 12–15.

12. $\sim p \wedge q$
13. $\sim(p \wedge q) \vee (p \vee q)$
14. $p \wedge (q \wedge r)$
15. $p \wedge (\sim q \vee r)$

Determine whether the statement forms in 16–24 are logically equivalent. In each case, construct a truth table and include a sentence justifying your answer. Your sentence should show that you understand the meaning of logical equivalence.

16. $p \vee (p \wedge q)$ and p
17. $\sim(p \wedge q)$ and $\sim p \wedge \sim q$
18. $p \vee \mathbf{t}$ and \mathbf{t}
19. $p \wedge \mathbf{t}$ and p
20. $p \wedge \mathbf{c}$ and $p \vee \mathbf{c}$
21. $(p \wedge q) \wedge r$ and $p \wedge (q \wedge r)$

*For exercises with blue numbers or letters, solutions are given in Appendix B. The symbol **H** indicates that only a hint or a partial solution is given. The symbol ***** signals that an exercise is more challenging than usual.

22. $p \wedge (q \vee r)$ and $(p \wedge q) \vee (p \wedge r)$

23. $(p \wedge q) \vee r$ and $p \wedge (q \vee r)$

24. $(p \vee q) \vee (p \wedge r)$ and $(p \vee q) \wedge r$

Use De Morgan's laws to write negations for the statements in 25–31.

25. Hal is a math major and Hal's sister is a computer science major.

26. Sam is an orange belt and Kate is a red belt.

27. The connector is loose or the machine is unplugged.

28. The units digit of 4^{67} is 4 or it is 6.

29. This computer program has a logical error in the first ten lines or it is being run with an incomplete data set.

30. The dollar is at an all-time high and the stock market is at a record low.

31. The train is late or my watch is fast.

Assume x is a particular real number and use De Morgan's laws to write negations for the statements in 32–37.

32. $-2 < x < 7$

33. $-10 < x < 2$

34. $x < 2$ or $x > 5$

35. $x \leq -1$ or $x > 1$

36. $1 > x \geq -3$

37. $0 > x \geq -7$

In 38 and 39, imagine that num_orders and num_instock are particular values, such as might occur during execution of a computer program. Write negations for the following statements.

38. $(\text{num_orders} > 100 \text{ and } \text{num_instock} \leq 500)$ or $\text{num_instock} < 200$

39. $(\text{num_orders} < 50 \text{ and } \text{num_instock} > 300)$ or $(50 \leq \text{num_orders} < 75 \text{ and } \text{num_instock} > 500)$

Use truth tables to establish which of the statement forms in 40–43 are tautologies and which are contradictions.

40. $(p \wedge q) \vee (\sim p \vee (p \wedge \sim q))$

41. $(p \wedge \sim q) \wedge (\sim p \vee q)$

42. $((\sim p \wedge q) \wedge (q \wedge r)) \wedge \sim q$

43. $(\sim p \vee q) \vee (p \wedge \sim q)$

In 44 and 45, determine whether the statements in (a) and (b) are logically equivalent.

44. Assume x is a particular real number.

a. $x < 2$ or it is not the case that $1 < x < 3$.

b. $x \leq 1$ or either $x < 2$ or $x \geq 3$.

45. a. Bob is a double math and computer science major and Ann is a math major, but Ann is not a double math and computer science major.

b. It is not the case that both Bob and Ann are double math and computer science majors, but it is the case that Ann is a math major and Bob is a double math and computer science major.

★ 46. In Example 2.1.4, the symbol \oplus was introduced to denote *exclusive or*, so $p \oplus q \equiv (p \vee q) \wedge \sim(p \wedge q)$. Hence the truth table for *exclusive or* is as follows:

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

a. Find simpler statement forms that are logically equivalent to $p \oplus p$ and $(p \oplus p) \oplus p$.

b. Is $(p \oplus q) \oplus r \equiv p \oplus (q \oplus r)$? Justify your answer.

c. Is $(p \oplus q) \wedge r \equiv (p \wedge r) \oplus (q \wedge r)$? Justify your answer.

★ 47. In logic and in standard English, a double negative is equivalent to a positive. There is one fairly common English usage in which a “double positive” is equivalent to a negative. What is it? Can you think of others?

In 48 and 49 below, a logical equivalence is derived from Theorem 2.1.1. Supply a reason for each step.

$$\begin{aligned}
 48. \quad (p \wedge \sim q) \vee (p \wedge q) &\equiv p \wedge (\sim q \vee q) && \text{by (a)} \\
 &\equiv p \wedge (q \vee \sim q) && \text{by (b)} \\
 &\equiv p \wedge \mathbf{t} && \text{by (c)} \\
 &\equiv p && \text{by (d)}
 \end{aligned}$$

Therefore, $(p \wedge \sim q) \vee (p \wedge q) \equiv p$.

$$\begin{aligned}
 49. \quad (p \vee \sim q) \wedge (\sim p \vee \sim q) & \\
 &\equiv (\sim q \vee p) \wedge (\sim q \vee \sim p) && \text{by (a)} \\
 &\equiv \sim q \vee (p \wedge \sim p) && \text{by (b)} \\
 &\equiv \sim q \vee \mathbf{f} && \text{by (c)} \\
 &\equiv \sim q && \text{by (d)}
 \end{aligned}$$

Therefore, $(p \vee \sim q) \wedge (\sim p \vee \sim q) \equiv \sim q$.

Use Theorem 2.1.1 to verify the logical equivalences in 50–54. Supply a reason for each step.

50. $(p \wedge \sim q) \vee p \equiv p$ 51. $p \wedge (\sim q \vee p) \equiv p$

52. $\sim(p \vee \sim q) \vee (\sim p \wedge \sim q) \equiv \sim p$

53. $\sim((\sim p \wedge q) \vee (\sim p \wedge \sim q)) \vee (p \wedge q) \equiv p$

54. $(p \wedge (\sim(\sim p \vee q))) \vee (p \wedge q) \equiv p$

Answers for Test Yourself

1. true 2. false 3. the same truth values 4. *or*; negated; *and*; negated 5. true 6. false

2.2 Conditional Statements

... hypothetical reasoning implies the subordination of the real to the realm of the possible ... — Jean Piaget, 1972

When you make a logical inference or deduction, you reason *from* a hypothesis *to* a conclusion. Your aim is to be able to say, “If such and such is known, *then* something or other must be the case.”

Let p and q be statements. A sentence of the form “If p then q ” is denoted symbolically by “ $p \rightarrow q$ ”; p is called the *hypothesis* and q is called the *conclusion*. For instance, consider the following statement:

If 4,686 is divisible by 6, then 4,686 is divisible by 3

hypothesis conclusion

Such a sentence is called *conditional* because the truth of statement q is conditioned on the truth of statement p .

The notation $p \rightarrow q$ indicates that \rightarrow is a connective, like \wedge or \vee , that can be used to join statements to create new statements. To define $p \rightarrow q$ as a statement, therefore, we must specify the truth values for $p \rightarrow q$ as we specified truth values for $p \wedge q$ and for $p \vee q$. As is the case with the other connectives, the formal definition of truth values for \rightarrow (if-then) is based on its everyday, intuitive meaning. Consider an example.

Suppose you go to interview for a job at a store and the owner of the store makes you the following promise:

If you show up for work Monday morning, then you will get the job.

Under what circumstances are you justified in saying the owner spoke falsely? That is, under what circumstances is the above sentence false? The answer is: You *do* show up for work Monday morning and you do *not* get the job.

After all, the owner’s promise only says you will get the job *if* a certain condition (showing up for work Monday morning) is met; it says nothing about what will happen if the condition is *not* met. So if the condition is not met, you cannot in fairness say the promise is false regardless of whether or not you get the job.

The above example was intended to convince you that *the only combination of circumstances in which you would call a conditional sentence false occurs when the hypothesis is true and the conclusion is false*. In all other cases, you would not call the sentence false. This implies that the only row of the truth table for $p \rightarrow q$ that should be filled in with an F is the row where p is T and q is F. No other row should contain an F. But each row of a truth table must be filled in with either a T or an F. Thus all other rows of the truth table for $p \rightarrow q$ must be filled in with T’s.

Truth Table for $p \rightarrow q$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

• Definition

If p and q are statement variables, the **conditional** of q by p is “If p then q ” or “ p implies q ” and is denoted $p \rightarrow q$. It is false when p is true and q is false; otherwise it is true. We call p the **hypothesis** (or **antecedent**) of the conditional and q the **conclusion** (or **consequent**).

A conditional statement that is true by virtue of the fact that its hypothesis is false is often called **vacuously true** or **true by default**. Thus the statement “If you show up for work Monday morning, then you will get the job” is vacuously true if you do not show up for work Monday morning. In general, when the “if” part of an if-then statement is false, the statement as a whole is said to be true, regardless of whether the conclusion is true or false.

Example 2.2.1 A Conditional Statement with a False Hypothesis

Consider the statement:

If $0 = 1$ then $1 = 2$.

As strange as it may seem, since the hypothesis of this statement is false, the statement as a whole is true. ■

The philosopher Willard Van Orman Quine advises against using the phrase “ p implies q ” to mean “ $p \rightarrow q$ ” because the word *implies* suggests that q can be logically deduced from p and this is often not the case. Nonetheless, the phrase is used by many people, probably because it is a convenient replacement for the \rightarrow symbol. And, of course, in many cases a conclusion can be deduced from a hypothesis, even when the hypothesis is false.

In expressions that include \rightarrow as well as other logical operators such as \wedge , \vee , and \sim , the **order of operations** is that \rightarrow is performed last. Thus, according to the specification of order of operations in Section 2.1, \sim is performed first, then \wedge and \vee , and finally \rightarrow .

Note For example, if $0 = 1$, then, by adding 1 to both sides of the equation, you can deduce that $1 = 2$.

Example 2.2.2 Truth Table for $p \vee \sim q \rightarrow \sim p$

Construct a truth table for the statement form $p \vee \sim q \rightarrow \sim p$.

Solution By the order of operations given above, the following two expressions are equivalent: $p \vee \sim q \rightarrow \sim p$ and $(p \vee (\sim q)) \rightarrow (\sim p)$, and this order governs the construction of the truth table. First fill in the four possible combinations of truth values for p and q , and then enter the truth values for $\sim p$ and $\sim q$ using the definition of negation. Next fill in the $p \vee \sim q$ column using the definition of \vee . Finally, fill in the $p \vee \sim q \rightarrow \sim p$ column using the definition of \rightarrow . The only rows in which the hypothesis $p \vee \sim q$ is true and the conclusion $\sim p$ is false are the first and second rows. So you put F’s in those two rows and T’s in the other two rows.

		conclusion		hypothesis	
p	q	$\sim p$	$\sim q$	$p \vee \sim q$	$p \vee \sim q \rightarrow \sim p$
T	T	F	F	T	F
T	F	F	T	T	F
F	T	T	F	F	T
F	F	T	T	T	T

and q be

You are fired.

Then the given statement is $\sim p \vee q$. Also p is

You do not get to work on time.

So the equivalent if-then version, $p \rightarrow q$, is

If you do not get to work on time, then you are fired. ■

The Negation of a Conditional Statement

By definition, $p \rightarrow q$ is false if, and only if, its hypothesis, p , is true and its conclusion, q , is false. It follows that

The negation of “if p then q ” is logically equivalent to “ p and not q .”

This can be restated symbolically as follows:

$$\sim(p \rightarrow q) \equiv p \wedge \sim q$$

You can also obtain this result by starting from the logical equivalence $p \rightarrow q \equiv \sim p \vee q$. Take the negation of both sides to obtain

$$\begin{aligned} \sim(p \rightarrow q) &\equiv \sim(\sim p \vee q) \\ &\equiv \sim(\sim p) \wedge (\sim q) && \text{by De Morgan's laws} \\ &\equiv p \wedge \sim q && \text{by the double negative law.} \end{aligned}$$

Yet another way to derive this result is to construct truth tables for $\sim(p \rightarrow q)$ and for $p \wedge \sim q$ and to check that they have the same truth values. (See exercise 13(b) at the end of this section.)

Example 2.2.5 Negations of If-Then Statements

Write negations for each of the following statements:

- If my car is in the repair shop, then I cannot get to class.
- If Sara lives in Athens, then she lives in Greece.

Solution

- My car is in the repair shop and I can get to class.
- Sara lives in Athens and she does not live in Greece. (Sara might live in Athens, Georgia; Athens, Ohio; or Athens, Wisconsin.) ■

It is tempting to write the negation of an if-then statement as another if-then statement. Please resist that temptation!



Caution! Remember that the negation of an if-then statement does not start with the word *if*.

The Contrapositive of a Conditional Statement

One of the most fundamental laws of logic is the equivalence between a conditional statement and its contrapositive.

• Definition

The **contrapositive** of a conditional statement of the form “If p then q ” is

If $\sim q$ then $\sim p$.

Symbolically,

The contrapositive of $p \rightarrow q$ is $\sim q \rightarrow \sim p$.

The fact is that

A conditional statement is logically equivalent to its contrapositive.

You are asked to establish this equivalence in exercise 26 at the end of this section.

Example 2.2.6 Writing the Contrapositive

Write each of the following statements in its equivalent contrapositive form:

- a. If Howard can swim across the lake, then Howard can swim to the island.
- b. If today is Easter, then tomorrow is Monday.

Solution

- a. If Howard cannot swim to the island, then Howard cannot swim across the lake.
- b. If tomorrow is not Monday, then today is not Easter. ■

When you are trying to solve certain problems, you may find that the contrapositive form of a conditional statement is easier to work with than the original statement. Replacing a statement by its contrapositive may give the extra push that helps you over the top in your search for a solution. This logical equivalence is also the basis for one of the most important laws of deduction, *modus tollens* (to be explained in Section 2.3), and for the contrapositive method of proof (to be explained in Section 4.6).

The Converse and Inverse of a Conditional Statement

The fact that a conditional statement and its contrapositive are logically equivalent is very important and has wide application. Two other variants of a conditional statement are *not* logically equivalent to the statement.

• Definition

Suppose a conditional statement of the form “If p then q ” is given.

1. The **converse** is “If q then p .”
2. The **inverse** is “If $\sim p$ then $\sim q$.”

Symbolically,

The converse of $p \rightarrow q$ is $q \rightarrow p$,

and

The inverse of $p \rightarrow q$ is $\sim p \rightarrow \sim q$.

Example 2.2.7 Writing the Converse and the Inverse

Write the converse and inverse of each of the following statements:

- a. If Howard can swim across the lake, then Howard can swim to the island.
- b. If today is Easter, then tomorrow is Monday.

Solution

- a. *Converse:* If Howard can swim to the island, then Howard can swim across the lake.
Inverse: If Howard cannot swim across the lake, then Howard cannot swim to the island.
- b. *Converse:* If tomorrow is Monday, then today is Easter.
Inverse: If today is not Easter, then tomorrow is not Monday. ■



Caution! Many people believe that if a conditional statement is true, then its converse and inverse must also be true. This is not correct!

Note that while the statement “If today is Easter, then tomorrow is Monday” is always true, both its converse and inverse are false on every Sunday except Easter.

1. A conditional statement and its converse are *not* logically equivalent.
2. A conditional statement and its inverse are *not* logically equivalent.
3. The converse and the inverse of a conditional statement are logically equivalent to each other.

In exercises 24, 25, and 27 at the end of this section, you are asked to use truth tables to verify the statements in the box above. Note that the truth of statement 3 also follows from the observation that the inverse of a conditional statement is the contrapositive of its converse.

Only If and the Biconditional

To say “ p only if q ” means that p can take place *only* if q takes place also. That is, if q does not take place, then p cannot take place. Another way to say this is that if p occurs, then q must also occur (by the logical equivalence between a statement and its contrapositive).

• Definition

It p and q are statements,

p **only if** q means “if not q then not p ,”

or, equivalently,

“if p then q .”

Example 2.2.8 Converting Only If to If-Then

Rewrite the following statement in if-then form in two ways, one of which is the contrapositive of the other.

John will break the world’s record for the mile run only if he runs the mile in under four minutes.

Solution *Version 1:* If John does not run the mile in under four minutes, then he will not break the world’s record.

Version 2: If John breaks the world’s record, then he will have run the mile in under four minutes. ■



Caution! “ p only if q ” does *not* mean “ p if q .”

Note that it is possible for “ p only if q ” to be true at the same time that “ p if q ” is false. For instance, to say that John will break the world’s record only if he runs the mile in under four minutes does not mean that John will break the world’s record if he runs the mile in under four minutes. His time could be under four minutes but still not be fast enough to break the record.

• Definition

Given statement variables p and q , the **biconditional of p and q** is “ p if, and only if, q ” and is denoted $p \leftrightarrow q$. It is true if both p and q have the same truth values and is false if p and q have opposite truth values. The words *if and only if* are sometimes abbreviated **iff**.

The biconditional has the following truth table:

Truth Table for $p \leftrightarrow q$

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

In order of operations \leftrightarrow is coequal with \rightarrow . As with \wedge and \vee , the only way to indicate precedence between them is to use parentheses. The full hierarchy of operations for the five logical operators is on the next page.

The occurrence of r is *necessary* to obtain the occurrence of s . Note that because of the equivalence between a statement and its contrapositive,

r is a necessary condition for s also means “if s then r .”

Consequently,

r is a necessary and sufficient condition for s means “ r if, and only if, s .”

Example 2.2.10 Interpreting Necessary and Sufficient Conditions

Consider the statement “If John is eligible to vote, then he is at least 18 years old.” The truth of the condition “John is eligible to vote” is *sufficient* to ensure the truth of the condition “John is at least 18 years old.” In addition, the condition “John is at least 18 years old” is *necessary* for the condition “John is eligible to vote” to be true. If John were younger than 18, then he would not be eligible to vote. ■

Example 2.2.11 Converting a Sufficient Condition to If-Then Form

Rewrite the following statement in the form “If A then B ”:

Pia’s birth on U.S. soil is a sufficient condition
for her to be a U.S. citizen.

Solution If Pia was born on U.S. soil, then she is a U.S. citizen. ■

Example 2.2.12 Converting a Necessary Condition to If-Then Form

Use the contrapositive to rewrite the following statement in two ways:

George’s attaining age 35 is a necessary condition
for his being president of the United States.

Solution *Version 1:* If George has not attained the age of 35, then he cannot be president of the United States.

Version 2: If George can be president of the United States, then he has attained the age of 35. ■

Remarks

1. *In logic, a hypothesis and conclusion are not required to have related subject matters.*

In ordinary speech we never say things like “If computers are machines, then Babe Ruth was a baseball player” or “If $2 + 2 = 5$, then Mickey Mouse is president of the United States.” We formulate a sentence like “If p then q ” only if there is some connection of content between p and q .

In logic, however, the two parts of a conditional statement need not have related meanings. The reason? If there were such a requirement, who would enforce it? What one person perceives as two unrelated clauses may seem related to someone else. There would have to be a central arbiter to check each conditional sentence before anyone could use it, to be sure its clauses were in proper relation. This is impractical, to say the least!

Thus a statement like “if computers are machines, then Babe Ruth was a baseball player” is allowed, and it is even called true because both its hypothesis and its conclusion are true. Similarly, the statement “If $2 + 2 = 5$, then Mickey Mouse is president of the United States” is allowed and is called true because its hypothesis is false, even though doing so may seem ridiculous.

In mathematics it often happens that a carefully formulated definition that successfully covers the situations for which it was primarily intended is later seen to be satisfied by some extreme cases that the formulator did not have in mind. But those are the breaks, and it is important to get into the habit of exploring definitions fully to seek out and understand *all* their instances, even the unusual ones.

2. *In informal language, simple conditionals are often used to mean biconditionals.*

The formal statement “ p if, and only if, q ” is seldom used in ordinary language. Frequently, when people intend the biconditional they leave out either the *and only if* or the *if and*. That is, they say either “ p if q ” or “ p only if q ” when they really mean “ p if, and only if, q .” For example, consider the statement “You will get dessert if, and only if, you eat your dinner.” Logically, this is equivalent to the conjunction of the following two statements.

Statement 1: If you eat your dinner, then you will get dessert.

Statement 2: You will get dessert only if you eat your dinner.

or

If you do not eat your dinner, then you will not get dessert.

Now how many parents in the history of the world have said to their children “You will get dessert if, and only if, you eat your dinner”? Not many! Most say either “If you eat your dinner, you will get dessert” (these take the positive approach—they emphasize the reward) or “You will get dessert only if you eat your dinner” (these take the negative approach—they emphasize the punishment). Yet the parents who promise the reward intend to suggest the punishment as well, and those who threaten the punishment will certainly give the reward if it is earned. Both sets of parents expect that their conditional statements will be interpreted as biconditionals.

Since we often (correctly) interpret conditional statements as biconditionals, it is not surprising that we may come to believe (mistakenly) that conditional statements are always logically equivalent to their inverses and converses. In formal settings, however, statements must have unambiguous interpretations. If-then statements can’t sometimes mean “if-then” and other times mean “if and only if.” When using language in mathematics, science, or other situations where precision is important, it is essential to interpret if-then statements according to the formal definition and not to confuse them with their converses and inverses.

Test Yourself

1. An *if-then* statement is false if, and only if, the hypothesis is ____ and the conclusion is ____.
2. The negation of “if p then q ” is ____.
3. The converse of “if p then q ” is ____.
4. The contrapositive of “if p then q ” is ____.
5. The inverse of “if p then q ” is ____.
6. A conditional statement and its contrapositive are ____.
7. A conditional statement and its converse are not ____.
8. “ R is a sufficient condition for S ” means “if ____ then ____.”
9. “ R is a necessary condition for S ” means “if ____ then ____.”
10. “ R only if S ” means “if ____ then ____.”

Exercise Set 2.2

Rewrite the statements in 1–4 in if-then form.

1. This loop will repeat exactly N times if it does not contain a **stop** or a **go to**.
2. I am on time for work if I catch the 8:05 bus.
3. Freeze or I'll shoot.
4. Fix my ceiling or I won't pay my rent.

Construct truth tables for the statement forms in 5–11.

5. $\sim p \vee q \rightarrow \sim q$
6. $(p \vee q) \vee (\sim p \wedge q) \rightarrow q$
7. $p \wedge \sim q \rightarrow r$
8. $\sim p \vee q \rightarrow r$
9. $p \wedge \sim r \leftrightarrow q \vee r$
10. $(p \rightarrow r) \leftrightarrow (q \rightarrow r)$

$$11. (p \rightarrow (q \rightarrow r)) \leftrightarrow ((p \wedge q) \rightarrow r)$$

12. Use the logical equivalence established in Example 2.2.3, $p \vee q \rightarrow r \equiv (p \rightarrow r) \wedge (q \rightarrow r)$, to rewrite the following statement. (Assume that x represents a fixed real number.)

If $x > 2$ or $x < -2$, then $x^2 > 4$.

13. Use truth tables to verify the following logical equivalences. Include a few words of explanation with your answers.

$$\text{a. } p \rightarrow q \equiv \sim p \vee q \quad \text{b. } \sim(p \rightarrow q) \equiv p \wedge \sim q.$$

- H** 14. **a.** Show that the following statement forms are all logically equivalent.

$$p \rightarrow q \vee r, \quad p \wedge \sim q \rightarrow r, \quad \text{and} \quad p \wedge \sim r \rightarrow q$$

- b.** Use the logical equivalences established in part (a) to rewrite the following sentence in two different ways. (Assume that n represents a fixed integer.)

If n is prime, then n is odd or n is 2.

15. Determine whether the following statement forms are logically equivalent:

$$p \rightarrow (q \rightarrow r) \quad \text{and} \quad (p \rightarrow q) \rightarrow r$$

In 16 and 17, write each of the two statements in symbolic form and determine whether they are logically equivalent. Include a truth table and a few words of explanation.

16. If you paid full price, you didn't buy it at Crown Books. You didn't buy it at Crown Books or you paid full price.
17. If 2 is a factor of n and 3 is a factor of n , then 6 is a factor of n . 2 is not a factor of n or 3 is not a factor of n or 6 is a factor of n .
18. Write each of the following three statements in symbolic form and determine which pairs are logically equivalent. Include truth tables and a few words of explanation.

If it walks like a duck and it talks like a duck, then it is a duck.

Either it does not walk like a duck or it does not talk like a duck, or it is a duck.

If it does not walk like a duck and it does not talk like a duck, then it is not a duck.

19. True or false? The negation of "If Sue is Luiz's mother, then Ali is his cousin" is "If Sue is Luiz's mother, then Ali is not his cousin."
20. Write negations for each of the following statements. (Assume that all variables represent fixed quantities or entities, as appropriate.)
 - a. If P is a square, then P is a rectangle.
 - b. If today is New Year's Eve, then tomorrow is January.
 - c. If the decimal expansion of r is terminating, then r is rational.
 - d. If n is prime, then n is odd or n is 2.
 - e. If x is nonnegative, then x is positive or x is 0.
 - f. If Tom is Ann's father, then Jim is her uncle and Sue is her aunt.
 - g. If n is divisible by 6, then n is divisible by 2 and n is divisible by 3.
21. Suppose that p and q are statements so that $p \rightarrow q$ is false. Find the truth values of each of the following:

$$\text{a. } \sim p \rightarrow q \quad \text{b. } p \vee q \quad \text{c. } q \rightarrow p$$

- H** 22. Write contrapositives for the statements of exercise 20.

- H** 23. Write the converse and inverse for each statement of exercise 20.

Use truth tables to establish the truth of each statement in 24–27.

24. A conditional statement is not logically equivalent to its converse.
25. A conditional statement is not logically equivalent to its inverse.
26. A conditional statement and its contrapositive are logically equivalent to each other.
27. The converse and inverse of a conditional statement are logically equivalent to each other.

- H** 28. "Do you mean that you think you can find out the answer to it?" said the March Hare.

"Exactly so," said Alice.

"Then you should say what you mean," the March Hare went on.

"I do," Alice hastily replied; "at least—at least I mean what I say—that's the same thing, you know."

“Not the same thing a bit!” said the Hatter. “Why, you might just as well say that ‘I see what I eat’ is the same thing as ‘I eat what I see’!”

—from “A Mad Tea-Party” in *Alice in Wonderland*,
by Lewis Carroll

The Hatter is right. “I say what I mean” is not the same thing as “I mean what I say.” Rewrite each of these two sentences in if-then form and explain the logical relation between them. (This exercise is referred to in the introduction to Chapter 4.)

If statement forms P and Q are logically equivalent, then $P \leftrightarrow Q$ is a tautology. Conversely, if $P \leftrightarrow Q$ is a tautology, then P and Q are logically equivalent. Use \leftrightarrow to convert each of the logical equivalences in 29–31 to a tautology. Then use a truth table to verify each tautology.

$$29. p \rightarrow (q \vee r) \equiv (p \wedge \sim q) \rightarrow r$$

$$30. p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$

$$31. p \rightarrow (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$$

Rewrite each of the statements in 32 and 33 as a conjunction of two if-then statements.

32. This quadratic equation has two distinct real roots if, and only if, its discriminant is greater than zero.

33. This integer is even if, and only if, it equals twice some integer.

Rewrite the statements in 34 and 35 in if-then form in two ways, one of which is the contrapositive of the other.

34. The Cubs will win the pennant only if they win tomorrow’s game.

35. Sam will be allowed on Signe’s racing boat only if he is an expert sailor.

36. Taking the long view on your education, you go to the Prestige Corporation and ask what you should do in college to be hired when you graduate. The personnel director replies that you will be hired *only if* you major in mathematics or computer science, get a B average or better, and take accounting. You do, in fact, become a math major, get a B⁺ average, and take accounting. You return to Prestige Corporation, make a formal application, and are turned down. Did the personnel director lie to you?

Some programming languages use statements of the form “ r unless s ” to mean that as long as s does not happen, then r will happen. More formally:

Definition: If r and s are statements,
 r unless s means if $\sim s$ then r .

In 37–39, rewrite the statements in if-then form.

37. Payment will be made on the fifth unless a new hearing is granted.

38. Ann will go unless it rains.

39. This door will not open unless a security code is entered.

Rewrite the statements in 40 and 41 in if-then form.

40. Catching the 8:05 bus is a sufficient condition for my being on time for work.

41. Having two 45° angles is a sufficient condition for this triangle to be a right triangle.

Use the contrapositive to rewrite the statements in 42 and 43 in if-then form in two ways.

42. Being divisible by 3 is a necessary condition for this number to be divisible by 9.

43. Doing homework regularly is a necessary condition for Jim to pass the course.

Note that “a sufficient condition for s is r ” means r is a sufficient condition for s and that “a necessary condition for s is r ” means r is a necessary condition for s . Rewrite the statements in 44 and 45 in if-then form.

44. A sufficient condition for Jon’s team to win the championship is that it win the rest of its games.

45. A necessary condition for this computer program to be correct is that it not produce error messages during translation.

46. “If compound X is boiling, then its temperature must be at least 150°C.” Assuming that this statement is true, which of the following must also be true?

- a. If the temperature of compound X is at least 150°C, then compound X is boiling.
- b. If the temperature of compound X is less than 150°C, then compound X is not boiling.
- c. Compound X will boil only if its temperature is at least 150°C.
- d. If compound X is not boiling, then its temperature is less than 150°C.
- e. A necessary condition for compound X to boil is that its temperature be at least 150°C.
- f. A sufficient condition for compound X to boil is that its temperature be at least 150°C.

In 47–50 (a) use the logical equivalences $p \rightarrow q \equiv \sim p \vee q$ and $p \leftrightarrow q \equiv (\sim p \vee q) \wedge (\sim q \vee p)$ to rewrite the given statement forms without using the symbol \rightarrow or \leftrightarrow , and (b) use the logical equivalence $p \vee q \equiv \sim(\sim p \wedge \sim q)$ to rewrite each statement form using only \wedge and \sim .

$$47. p \wedge \sim q \rightarrow r \qquad 48. p \vee \sim q \rightarrow r \vee q$$

$$49. (p \rightarrow r) \leftrightarrow (q \rightarrow r)$$

$$50. (p \rightarrow (q \rightarrow r)) \leftrightarrow ((p \wedge q) \rightarrow r)$$

51. Given any statement form, is it possible to find a logically equivalent form that uses only \sim and \wedge ? Justify your answer.

Answers for Test Yourself

1. true; false 2. $p \wedge \sim q$ 3. if q then p 4. if $\sim q$ then $\sim p$ 5. if $\sim p$ then $\sim q$ 6. logically equivalent 7. logically equivalent 8. $R; S$ 9. $S; R$ 10. $R; S$

2.3 Valid and Invalid Arguments

“Contrariwise,” continued Tweedledee, “if it was so, it might be; and if it were so, it would be; but as it isn’t, it ain’t. That’s logic.” — Lewis Carroll, *Through the Looking Glass*

In mathematics and logic an argument is not a dispute. It is a sequence of statements ending in a conclusion. In this section we show how to determine whether an argument is valid—that is, whether the conclusion follows *necessarily* from the preceding statements. We will show that this determination depends only on the form of an argument, not on its content.

It was shown in Section 2.1 that the logical form of an argument can be abstracted from its content. For example, the argument

If Socrates is a man, then Socrates is mortal.
Socrates is a man.
 \therefore Socrates is mortal.

has the abstract form

If p then q
 p
 $\therefore q$

When considering the abstract form of an argument, think of p and q as variables for which statements may be substituted. An argument form is called *valid* if, and only if, whenever statements are substituted that make all the premises true, the conclusion is also true.

• Definition

An **argument** is a sequence of statements, and an **argument form** is a sequence of statement forms. All statements in an argument and all statement forms in an argument form, except for the final one, are called **premises** (or **assumptions** or **hypotheses**). The final statement or statement form is called the **conclusion**. The symbol \therefore , which is read “therefore,” is normally placed just before the conclusion.

To say that an *argument form* is **valid** means that no matter what particular statements are substituted for the statement variables in its premises, if the resulting premises are all true, then the conclusion is also true. To say that an *argument* is **valid** means that its form is valid.

The crucial fact about a valid argument is that the truth of its conclusion follows *necessarily* or *inescapably* or *by logical form alone* from the truth of its premises. It is impossible to have a valid argument with true premises and a false conclusion. When an argument is valid and its premises are true, the truth of the conclusion is said to be *inferred* or *deduced* from the truth of the premises. If a conclusion “ain’t necessarily so,” then it isn’t a valid deduction.

Testing an Argument Form for Validity

1. Identify the premises and conclusion of the argument form.
2. Construct a truth table showing the truth values of all the premises and the conclusion.
3. A row of the truth table in which all the premises are true is called a **critical row**. If there is a critical row in which the conclusion is false, then it is possible for an argument of the given form to have true premises and a false conclusion, and so the argument form is invalid. If the conclusion in *every* critical row is true, then the argument form is valid.

Example 2.3.1 Determining Validity or Invalidity

Determine whether the following argument form is valid or invalid by drawing a truth table, indicating which columns represent the premises and which represent the conclusion, and annotating the table with a sentence of explanation. When you fill in the table, you only need to indicate the truth values for the conclusion in the rows where all the premises are true (the critical rows) because the truth values of the conclusion in the other rows are irrelevant to the validity or invalidity of the argument.

$$\begin{aligned} p &\rightarrow q \vee \sim r \\ q &\rightarrow p \wedge r \\ \therefore p &\rightarrow r \end{aligned}$$

Solution The truth table shows that even though there are several situations in which the premises and the conclusion are all true (rows 1, 7, and 8), there is one situation (row 4) where the premises are true and the conclusion is false.

p	q	r	$\sim r$	$q \vee \sim r$	$p \wedge r$	premises		conclusion
						$p \rightarrow q \vee \sim r$	$q \rightarrow p \wedge r$	$p \rightarrow r$
T	T	T	F	T	T	T	T	T
T	T	F	T	T	F	T	F	
T	F	T	F	F	T	F	T	
T	F	F	T	T	F	T	T	F
F	T	T	F	T	F	T	F	
F	T	F	T	T	F	T	F	
F	F	T	F	F	F	T	T	T
F	F	F	T	T	F	T	T	T

This row shows that an argument of this form can have true premises and a false conclusion. Hence this form of argument is invalid.

Modus Ponens and Modus Tollens

An argument form consisting of two premises and a conclusion is called a **syllogism**. The first and second premises are called the **major premise** and **minor premise**, respectively.

The most famous form of syllogism in logic is called **modus ponens**. It has the following form:

If p then q .
 p
 $\therefore q$

Here is an argument of this form:

If the sum of the digits of 371,487 is divisible by 3,
 then 371,487 is divisible by 3.

The sum of the digits of 371,487 is divisible by 3.

\therefore 371,487 is divisible by 3.

The term *modus ponens* is Latin meaning “method of affirming” (the conclusion is an affirmation). Long before you saw your first truth table, you were undoubtedly being convinced by arguments of this form. Nevertheless, it is instructive to prove that modus ponens is a valid form of argument, if for no other reason than to confirm the agreement between the formal definition of validity and the intuitive concept. To do so, we construct a truth table for the premises and conclusion.

		premises		conclusion	
p	q	$p \rightarrow q$	p	q	
T	T	T	T	T	← critical row
T	F	F	T		
F	T	T	F		
F	F	T	F		

The first row is the only one in which both premises are true, and the conclusion in that row is also true. Hence the argument form is valid.

Now consider another valid argument form called **modus tollens**. It has the following form:

If p then q .
 $\sim q$
 $\therefore \sim p$

Here is an example of modus tollens:

If Zeus is human, then Zeus is mortal.

Zeus is not mortal.

\therefore Zeus is not human.

An intuitive explanation for the validity of modus tollens uses proof by contradiction. It goes like this:

Suppose

(1) If Zeus is human, then Zeus is mortal; and

(2) Zeus is not mortal.

Must Zeus necessarily be nonhuman?

Yes!

Because, if Zeus were human, then by (1) he would be mortal.

But by (2) he is not mortal.

Hence, Zeus cannot be human.

Modus tollens is Latin meaning “method of denying” (the conclusion is a denial). The validity of modus tollens can be shown to follow from modus ponens together with the fact that a conditional statement is logically equivalent to its contrapositive. Or it can be established formally by using a truth table. (See exercise 13.)

Studies by cognitive psychologists have shown that although nearly 100% of college students have a solid, intuitive understanding of modus ponens, less than 60% are able to apply modus tollens correctly.* Yet in mathematical reasoning, modus tollens is used almost as often as modus ponens. Thus it is important to study the form of modus tollens carefully to learn to use it effectively.

Example 2.3.2 Recognizing Modus Ponens and Modus Tollens

Use modus ponens or modus tollens to fill in the blanks of the following arguments so that they become valid inferences.

- a. If there are more pigeons than there are pigeonholes, then at least two pigeons roost in the same hole.

There are more pigeons than there are pigeonholes.

\therefore _____.

- b. If 870,232 is divisible by 6, then it is divisible by 3.

870,232 is not divisible by 3.

\therefore _____.

Solution

- a. At least two pigeons roost in the same hole. by modus ponens

- b. 870,232 is not divisible by 6. by modus tollens ■

Additional Valid Argument Forms: Rules of Inference

A **rule of inference** is a form of argument that is valid. Thus modus ponens and modus tollens are both rules of inference. The following are additional examples of rules of inference that are frequently used in deductive reasoning.

Example 2.3.3 Generalization

The following argument forms are valid:

a. p

$\therefore p \vee q$

b. q

$\therefore p \vee q$

These argument forms are used for making generalizations. For instance, according to the first, if p is true, then, more generally, “ p or q ” is true for *any* other statement q . As an example, suppose you are given the job of counting the upperclassmen at your school. You ask what class Anton is in and are told he is a junior.

**Cognitive Psychology and Its Implications*, 3d ed. by John R. Anderson (New York: Freeman, 1990), pp. 292–297.

You reason as follows:

Anton is a junior.

\therefore (more generally) Anton is a junior or Anton is a senior.

Knowing that upperclassman means junior or senior, you add Anton to your list. ■

Example 2.3.4 Specialization

The following argument forms are valid:

$$\begin{array}{l} \text{a. } p \wedge q \\ \therefore p \end{array}$$

$$\begin{array}{l} \text{b. } p \wedge q \\ \therefore q \end{array}$$

These argument forms are used for specializing. When classifying objects according to some property, you often know much more about them than whether they do or do not have that property. When this happens, you discard extraneous information as you concentrate on the particular property of interest.

For instance, suppose you are looking for a person who knows graph algorithms to work with you on a project. You discover that Ana knows both numerical analysis and graph algorithms. You reason as follows:

Ana knows numerical analysis and Ana knows graph algorithms.

\therefore (in particular) Ana knows graph algorithms.

Accordingly, you invite her to work with you on your project. ■

Both generalization and specialization are used frequently in mathematics to tailor facts to fit into hypotheses of known theorems in order to draw further conclusions. Elimination, transitivity, and proof by division into cases are also widely used tools.

Example 2.3.5 Elimination

The following argument forms are valid:

$$\begin{array}{l} \text{a. } p \vee q \\ \sim q \\ \therefore p \end{array}$$

$$\begin{array}{l} \text{b. } p \vee q \\ \sim p \\ \therefore q \end{array}$$

These argument forms say that when you have only two possibilities and you can rule one out, the other must be the case. For instance, suppose you know that for a particular number x ,

$$x - 3 = 0 \quad \text{or} \quad x + 2 = 0.$$

If you also know that x is not negative, then $x \neq -2$, so

$$x + 2 \neq 0.$$

By elimination, you can then conclude that

$$\therefore x - 3 = 0. \quad \blacksquare$$

Example 2.3.6 Transitivity

The following argument form is valid:

$$\begin{array}{l} p \rightarrow q \\ q \rightarrow r \\ \therefore p \rightarrow r \end{array}$$

Many arguments in mathematics contain chains of if-then statements. From the fact that one statement implies a second and the second implies a third, you can conclude that the first statement implies the third. Here is an example:

If 18,486 is divisible by 18, then 18,486 is divisible by 9.

If 18,486 is divisible by 9, then the sum of the digits of 18,486 is divisible by 9.

∴ If 18,486 is divisible by 18, then the sum of the digits of 18,486 is divisible by 9. ■

Example 2.3.7 Proof by Division into Cases

The following argument form is valid:

$$\begin{array}{l} p \vee q \\ p \rightarrow r \\ q \rightarrow r \\ \hline \therefore r \end{array}$$

It often happens that you know one thing or another is true. If you can show that in either case a certain conclusion follows, then this conclusion must also be true. For instance, suppose you know that x is a particular nonzero real number. The trichotomy property of the real numbers says that any number is positive, negative, or zero. Thus (by elimination) you know that x is positive or x is negative. You can deduce that $x^2 > 0$ by arguing as follows:

x is positive or x is negative.

If x is positive, then $x^2 > 0$.

If x is negative, then $x^2 > 0$.

∴ $x^2 > 0$. ■

The rules of valid inference are used constantly in problem solving. Here is an example from everyday life.

Example 2.3.8 Application: A More Complex Deduction

You are about to leave for school in the morning and discover that you don't have your glasses. You know the following statements are true:

- If I was reading the newspaper in the kitchen, then my glasses are on the kitchen table.
- If my glasses are on the kitchen table, then I saw them at breakfast.
- I did not see my glasses at breakfast.
- I was reading the newspaper in the living room or I was reading the newspaper in the kitchen.
- If I was reading the newspaper in the living room then my glasses are on the coffee table.

Where are the glasses?

Solution Let RK = I was reading the newspaper in the kitchen.

GK = My glasses are on the kitchen table.

SB = I saw my glasses at breakfast.

RL = I was reading the newspaper in the living room.

GC = My glasses are on the coffee table.

Here is a sequence of steps you might use to reach the answer, together with the rules of inference that allow you to draw the conclusion of each step:

1. $RK \rightarrow GK$ by (a)
 $GK \rightarrow SB$ by (d)
 $\therefore RK \rightarrow SB$ by transitivity
2. $RK \rightarrow SB$ by the conclusion of (1)
 $\sim SB$ by (c)
 $\therefore \sim RK$ by modus tollens
3. $RL \vee RK$ by (d)
 $\sim RK$ by the conclusion of (2)
 $\therefore RL$ by elimination
4. $RL \rightarrow GC$ by (e)
 RL by the conclusion of (3)
 $\therefore GC$ by modus ponens

Thus the glasses are on the coffee table. ■

Fallacies

A **fallacy** is an error in reasoning that results in an invalid argument. Three common fallacies are **using ambiguous premises**, and treating them as if they were unambiguous, **circular reasoning** (assuming what is to be proved without having derived it from the premises), and **jumping to a conclusion** (without adequate grounds). In this section we discuss two other fallacies, called *converse error* and *inverse error*, which give rise to arguments that superficially resemble those that are valid by modus ponens and modus tollens but are not, in fact, valid.

As in previous examples, you can show that an argument is invalid by constructing a truth table for the argument form and finding at least one critical row in which all the premises are true but the conclusion is false. Another way is to find an argument of the same form with true premises and a false conclusion.

For an argument to be valid, every argument of the same form whose premises are all true must have a true conclusion. It follows that for an argument to be invalid means that there is an argument of that form whose premises are all true and whose conclusion is false.

Example 2.3.9 Converse Error

Show that the following argument is invalid:

If Zeke is a cheater, then Zeke sits in the back row.
 Zeke sits in the back row.
 \therefore Zeke is a cheater.

Solution Many people recognize the invalidity of the above argument intuitively, reasoning something like this: The first premise gives information about Zeke *if* it is known he is a cheater. It doesn't give any information about him if it is not already known that he is a cheater. One can certainly imagine a person who is not a cheater but happens to sit in the

back row. Then if that person's name is substituted for Zeke, the first premise is true by default and the second premise is also true but the conclusion is false.

The general form of the previous argument is as follows:

$$\begin{array}{l} p \rightarrow q \\ q \\ \therefore p \end{array}$$

In exercise 12(a) at the end of this section you are asked to use a truth table to show that this form of argument is invalid. ■

The fallacy underlying this invalid argument form is called the **converse error** because the conclusion of the argument would follow from the premises if the premise $p \rightarrow q$ were replaced by its converse. Such a replacement is not allowed, however, because a conditional statement is not logically equivalent to its converse. Converse error is also known as the *fallacy of affirming the consequent*.

Another common error in reasoning is called the *inverse error*.

Example 2.3.10 Inverse Error

Consider the following argument:

If interest rates are going up, stock market prices will go down.
Interest rates are not going up.
 \therefore Stock market prices will not go down.

Note that this argument has the following form:

$$\begin{array}{l} p \rightarrow q \\ \sim p \\ \therefore \sim q \end{array}$$

You are asked to give a truth table verification of the invalidity of this argument form in exercise 12(b) at the end of this section.

The fallacy underlying this invalid argument form is called the **inverse error** because the conclusion of the argument would follow from the premises if the premise $p \rightarrow q$ were replaced by its inverse. Such a replacement is not allowed, however, because a conditional statement is not logically equivalent to its inverse. Inverse error is also known as the *fallacy of denying the antecedent*. ■

Sometimes people lump together the ideas of validity and truth. If an argument seems valid, they accept the conclusion as true. And if an argument seems fishy (really a slang expression for invalid), they think the conclusion must be false. This is not correct!

Example 2.3.11 A Valid Argument with a False Premise and a False Conclusion

The argument below is valid by modus ponens. But its major premise is false, and so is its conclusion.

If John Lennon was a rock star, then John Lennon had red hair.
John Lennon was a rock star.
 \therefore John Lennon had red hair. ■



Caution! In logic, the words *true* and *valid* have very different meanings. A valid argument may have a false conclusion, and an invalid argument may have a true conclusion.

Example 2.3.12 An Invalid Argument with True Premises and a True Conclusion

The argument below is invalid by the converse error, but it has a true conclusion.

If New York is a big city, then New York has tall buildings.
 New York has tall buildings.
 \therefore New York is a big city.

• **Definition**

An argument is called **sound** if, and only if, it is valid *and* all its premises are true. An argument that is not sound is called **unsound**.

The important thing to note is that validity is a property of argument *forms*: If an argument is valid, then so is every other argument that has the same form. Similarly, if an argument is invalid, then so is every other argument that has the same form. What characterizes a valid argument is that no argument whose form is valid can have all true premises and a false conclusion. For each valid argument, there are arguments of that form with all true premises and a true conclusion, with at least one false premise and a true conclusion, and with at least one false premise and a false conclusion. On the other hand, for each invalid argument, there are arguments of that form with every combination of truth values for the premises and conclusion, including all true premises and a false conclusion. The bottom line is that we can only be sure that the conclusion of an argument is true when we know that the argument is sound, that is, when we know both that the argument is valid and that it has all true premises.

Contradictions and Valid Arguments

The concept of logical contradiction can be used to make inferences through a technique of reasoning called the *contradiction rule*. Suppose p is some statement whose truth you wish to deduce.

Contradiction Rule

If you can show that the supposition that statement p is false leads logically to a contradiction, then you can conclude that p is true.

Example 2.3.13 Contradiction Rule

Show that the following argument form is valid:

$\sim p \rightarrow \mathbf{c}$, where \mathbf{c} is a contradiction
 $\therefore p$

Solution Construct a truth table for the premise and the conclusion of this argument.

premises			conclusion	
p	$\sim p$	\mathbf{c}	$\sim p \rightarrow \mathbf{c}$	p
T	F	F	T	T
F	T	F	F	

There is only one critical row in which the premise is true, and in this row the conclusion is also true. Hence this form of argument is valid.

The contradiction rule is the logical heart of the method of proof by contradiction. A slight variation also provides the basis for solving many logical puzzles by eliminating contradictory answers: *If an assumption leads to a contradiction, then that assumption must be false.*

Example 2.3.14 Knights and Knaves

The logician Raymond Smullyan describes an island containing two types of people: knights who always tell the truth and knaves who always lie.* You visit the island and are approached by two natives who speak to you as follows:

A says: *B* is a knight.

B says: *A* and I are of opposite type.

What are *A* and *B*?

Solution *A* and *B* are both knaves. To see this, reason as follows:

Suppose *A* is a knight.

∴ What *A* says is true. by definition of knight

∴ *B* is also a knight. That's what *A* said.

∴ What *B* says is true. by definition of knight

∴ *A* and *B* are of opposite types. That's what *B* said.

∴ We have arrived at the following contradiction: *A* and *B* are both knights and *A* and *B* are of opposite type.

∴ The supposition is false. by the contradiction rule

∴ *A* is not a knight. negation of supposition

∴ *A* is a knave. by elimination: It's given that all inhabitants are knights or knaves, so since *A* is not a knight, *A* is a knave.

∴ What *A* says is false.

∴ *B* is not a knight.

∴ *B* is also a knave. by elimination

This reasoning shows that if the problem has a solution at all, then *A* and *B* must both be knaves. It is conceivable, however, that the problem has no solution. The problem statement could be inherently contradictory. If you look back at the solution, though, you can see that it does work out for both *A* and *B* to be knaves. ■

Summary of Rules of Inference

Table 2.3.1 summarizes some of the most important rules of inference.

*Raymond Smullyan has written a delightful series of whimsical yet profound books of logical puzzles starting with *What Is the Name of This Book?* (Englewood Cliffs, New Jersey: Prentice-Hall, 1978). Other good sources of logical puzzles are the many excellent books of Martin Gardner, such as *Aha! Insight* and *Aha! Gotcha* (New York: W. H. Freeman, 1978, 1982).



Raymond Smullyan
(born 1919)

Indiana University Archives

Table 2.3.1 Valid Argument Forms

Modus Ponens	$p \rightarrow q$ p $\therefore q$	Elimination	a. $p \vee q$ $\sim q$ $\therefore p$	b. $p \vee q$ $\sim p$ $\therefore q$
Modus Tollens	$p \rightarrow q$ $\sim q$ $\therefore \sim p$	Transitivity	$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	
Generalization	a. p $\therefore p \vee q$	b. q $\therefore p \vee q$	Proof by Division into Cases $p \vee q$ $p \rightarrow r$ $q \rightarrow r$ $\therefore r$	
Specialization	a. $p \wedge q$ $\therefore p$	b. $p \wedge q$ $\therefore q$		
Conjunction	p q $\therefore p \wedge q$	Contradiction Rule	$\sim p \rightarrow c$ $\therefore p$	

Test Yourself

- For an argument to be valid means that every argument of the same form whose premises _____ has a _____ conclusion.
- For an argument to be invalid means that there is an argument of the same form whose premises _____ and whose conclusion _____.
- For an argument to be sound means that it is _____ and its premises _____. In this case we can be sure that its conclusion _____.

Exercise Set 2.3

Use modus ponens or modus tollens to fill in the blanks in the arguments of 1–5 so as to produce valid inferences.

- If $\sqrt{2}$ is rational, then $\sqrt{2} = a/b$ for some integers a and b .
It is not true that $\sqrt{2} = a/b$ for some integers a and b .
 \therefore _____.
- If $1 - 0.99999 \dots$ is less than every positive real number, then it equals zero.

 \therefore The number $1 - 0.99999 \dots$ equals zero.
- If logic is easy, then I am a monkey's uncle.
I am not a monkey's uncle.
 \therefore _____.
- If this figure is a quadrilateral, then the sum of its interior angles is 360° .
The sum of the interior angles of this figure is not 360° .
 \therefore _____.

- If they were unsure of the address, then they would have telephoned.

 \therefore They were sure of the address.

Use truth tables to determine whether the argument forms in 6–11 are valid. Indicate which columns represent the premises and which represent the conclusion, and include a sentence explaining how the truth table supports your answer. Your explanation should show that you understand what it means for a form of argument to be valid or invalid.

- $p \rightarrow q$
 $q \rightarrow p$
 $\therefore p \vee q$
- p
 $p \rightarrow q$
 $\sim q \vee r$
 $\therefore r$
- $p \vee q$
 $p \rightarrow \sim q$
 $p \rightarrow r$
 $\therefore r$
- $p \wedge q \rightarrow \sim r$
 $p \vee \sim q$
 $\sim q \rightarrow p$
 $\therefore \sim r$

10. $p \rightarrow r$
 $q \rightarrow r$
 $\therefore p \vee q \rightarrow r$
11. $p \rightarrow q \vee r$
 $\sim q \vee \sim r$
 $\therefore \sim p \vee \sim r$

12. Use truth tables to show that the following forms of argument are invalid.

- a. $p \rightarrow q$
 q
 $\therefore p$
 (converse error)
- b. $p \rightarrow q$
 $\sim p$
 $\therefore \sim q$
 (inverse error)

Use truth tables to show that the argument forms referred to in 13–21 are valid. Indicate which columns represent the premises and which represent the conclusion, and include a sentence explaining how the truth table supports your answer. Your explanation should show that you understand what it means for a form of argument to be valid.

13. Modus tollens:

$$\begin{array}{l} p \rightarrow q \\ \sim q \\ \hline \therefore \sim p \end{array}$$

14. Example 2.3.3(a) 15. Example 2.3.3(b)
 16. Example 2.3.4(a) 17. Example 2.3.4(b)
 18. Example 2.3.5(a) 19. Example 2.3.5(b)
 20. Example 2.3.6 21. Example 2.3.7

Use symbols to write the logical form of each argument in 22 and 23, and then use a truth table to test the argument for validity. Indicate which columns represent the premises and which represent the conclusion, and include a few words of explanation showing that you understand the meaning of validity.

22. If Tom is not on team A, then Hua is on team B.
 If Hua is not on team B, then Tom is on team A.
 \therefore Tom is not on team A or Hua is not on team B.
23. Oleg is a math major or Oleg is an economics major.
 If Oleg is a math major, then Oleg is required to take Math 362.
 \therefore Oleg is an economics major or Oleg is not required to take Math 362.

Some of the arguments in 24–32 are valid, whereas others exhibit the converse or the inverse error. Use symbols to write the logical form of each argument. If the argument is valid, identify the rule of inference that guarantees its validity. Otherwise, state whether the converse or the inverse error is made.

24. If Jules solved this problem correctly, then Jules obtained the answer 2.
 Jules obtained the answer 2.
 \therefore Jules solved this problem correctly.
25. This real number is rational or it is irrational.
 This real number is not rational.
 \therefore This real number is irrational.

26. If I go to the movies, I won't finish my homework. If I don't finish my homework, I won't do well on the exam tomorrow.
 \therefore If I go to the movies, I won't do well on the exam tomorrow.
27. If this number is larger than 2, then its square is larger than 4.
 This number is not larger than 2.
 \therefore The square of this number is not larger than 4.
28. If there are as many rational numbers as there are irrational numbers, then the set of all irrational numbers is infinite.
 The set of all irrational numbers is infinite.
 \therefore There are as many rational numbers as there are irrational numbers.
29. If at least one of these two numbers is divisible by 6, then the product of these two numbers is divisible by 6.
 Neither of these two numbers is divisible by 6.
 \therefore The product of these two numbers is not divisible by 6.
30. If this computer program is correct, then it produces the correct output when run with the test data my teacher gave me.
 This computer program produces the correct output when run with the test data my teacher gave me.
 \therefore This computer program is correct.
31. Sandra knows Java and Sandra knows C++.
 \therefore Sandra knows C++.
32. If I get a Christmas bonus, I'll buy a stereo.
 If I sell my motorcycle, I'll buy a stereo.
 \therefore If I get a Christmas bonus or I sell my motorcycle, then I'll buy a stereo.
33. Give an example (other than Example 2.3.11) of a valid argument with a false conclusion.
34. Give an example (other than Example 2.3.12) of an invalid argument with a true conclusion.
35. Explain in your own words what distinguishes a valid form of argument from an invalid one.
36. Given the following information about a computer program, find the mistake in the program.
- There is an undeclared variable or there is a syntax error in the first five lines.
 - If there is a syntax error in the first five lines, then there is a missing semicolon or a variable name is misspelled.
 - There is not a missing semicolon.
 - There is not a misspelled variable name.

37. In the back of an old cupboard you discover a note signed by a pirate famous for his bizarre sense of humor and love of logical puzzles. In the note he wrote that he had hidden treasure somewhere on the property. He listed five true statements (a–e below) and challenged the reader to use them to figure out the location of the treasure.

- If this house is next to a lake, then the treasure is not in the kitchen.
- If the tree in the front yard is an elm, then the treasure is in the kitchen.
- This house is next to a lake.
- The tree in the front yard is an elm or the treasure is buried under the flagpole.
- If the tree in the back yard is an oak, then the treasure is in the garage.

Where is the treasure hidden?

38. You are visiting the island described in Example 2.3.14 and have the following encounters with natives.

- Two natives *A* and *B* address you as follows:
A says: Both of us are knights.
B says: *A* is a knave.
 What are *A* and *B*?
- Another two natives *C* and *D* approach you but only *C* speaks.
C says: Both of us are knaves.
 What are *C* and *D*?
- You then encounter natives *E* and *F*.
E says: *F* is a knave.
F says: *E* is a knave.
 How many knaves are there?
- Finally, you meet a group of six natives, *U*, *V*, *W*, *X*, *Y*, and *Z*, who speak to you as follows:
U says: None of us is a knight.
V says: At least three of us are knights.
W says: At most three of us are knights.
X says: Exactly five of us are knights.
Y says: Exactly two of us are knights.
Z says: Exactly one of us is a knight.
 Which are knights and which are knaves?

39. The famous detective Percule Hoirot was called in to solve a baffling murder mystery. He determined the following facts:

- Lord Hazelton, the murdered man, was killed by a blow on the head with a brass candlestick.
- Either Lady Hazelton or a maid, Sara, was in the dining room at the time of the murder.

- If the cook was in the kitchen at the time of the murder, then the butler killed Lord Hazelton with a fatal dose of strychnine.
 - If Lady Hazelton was in the dining room at the time of the murder, then the chauffeur killed Lord Hazelton.
 - If the cook was not in the kitchen at the time of the murder, then Sara was not in the dining room when the murder was committed.
 - If Sara was in the dining room at the time the murder was committed, then the wine steward killed Lord Hazelton.
- Is it possible for the detective to deduce the identity of the murderer from these facts? If so, who did murder Lord Hazelton? (Assume there was only one cause of death.)

40. Sharky, a leader of the underworld, was killed by one of his own band of four henchmen. Detective Sharp interviewed the men and determined that all were lying except for one. He deduced who killed Sharky on the basis of the following statements:

- Socko: Lefty killed Sharky.
 - Fats: Muscles didn't kill Sharky.
 - Lefty: Muscles was shooting craps with Socko when Sharky was knocked off.
 - Muscles: Lefty didn't kill Sharky.
- Who did kill Sharky?

In 41–44 a set of premises and a conclusion are given. Use the valid argument forms listed in Table 2.3.1 to deduce the conclusion from the premises, giving a reason for each step as in Example 2.3.8. Assume all variables are statement variables.

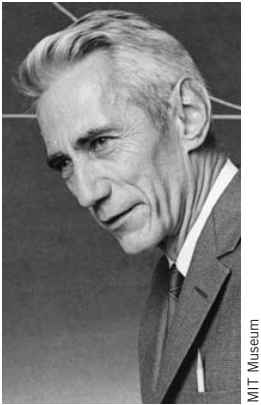
- | | |
|---|------------------------------------|
| 41. a. $\sim p \vee q \rightarrow r$ | 42. a. $p \vee q$ |
| b. $s \vee \sim q$ | b. $q \rightarrow r$ |
| c. $\sim t$ | c. $p \wedge s \rightarrow t$ |
| d. $p \rightarrow t$ | d. $\sim r$ |
| e. $\sim p \wedge r \rightarrow \sim s$ | e. $\sim q \rightarrow u \wedge s$ |
| f. $\therefore \sim q$ | f. $\therefore t$ |
| 43. a. $\sim p \rightarrow r \wedge \sim s$ | 44. a. $p \rightarrow q$ |
| b. $t \rightarrow s$ | b. $r \vee s$ |
| c. $u \rightarrow \sim p$ | c. $\sim s \rightarrow \sim t$ |
| d. $\sim w$ | d. $\sim q \vee s$ |
| e. $u \vee w$ | e. $\sim s$ |
| f. $\therefore \sim t$ | f. $\sim p \wedge r \rightarrow u$ |
| | g. $w \vee t$ |
| | h. $\therefore u \wedge w$ |

Answers for Test Yourself

1. are all true; true 2. are all true; is false 3. valid; are all true; is true

2.4 Application: Digital Logic Circuits

Only connect! — E. M. Forster, *Howards End*



Claude Shannon
(1916–2001)

In the late 1930s, a young M.I.T. graduate student named Claude Shannon noticed an analogy between the operations of switching devices, such as telephone switching circuits, and the operations of logical connectives. He used this analogy with striking success to solve problems of circuit design and wrote up his results in his master’s thesis, which was published in 1938.

The drawing in Figure 2.4.1(a) shows the appearance of the two positions of a simple switch. When the switch is closed, current can flow from one terminal to the other; when it is open, current cannot flow. Imagine that such a switch is part of the circuit shown in Figure 2.4.1(b). The light bulb turns on if, and only if, current flows through it. And this happens if, and only if, the switch is closed.

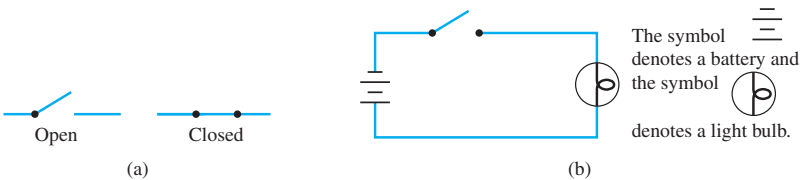


Figure 2.4.1

Now consider the more complicated circuits of Figures 2.4.2(a) and 2.4.2(b).

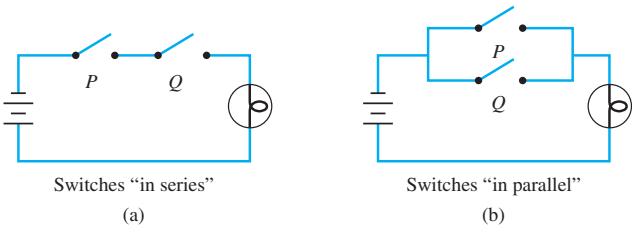


Figure 2.4.2

In the circuit of Figure 2.4.2(a) current flows and the light bulb turns on if, and only if, *both* switches *P* and *Q* are closed. The switches in this circuit are said to be **in series**. In the circuit of Figure 2.4.2(b) current flows and the light bulb turns on if, and only if, *at least one* of the switches *P* or *Q* is closed. The switches in this circuit are said to be **in parallel**. All possible behaviors of these circuits are described by Table 2.4.1.

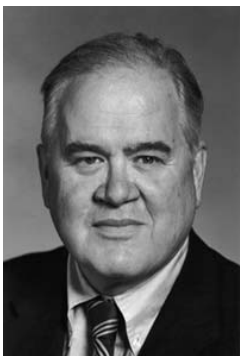
Table 2.4.1

(a) Switches in Series			(b) Switches in Parallel		
Switches		Light Bulb	Switches		Light Bulb
<i>P</i>	<i>Q</i>	State	<i>P</i>	<i>Q</i>	State
closed	closed	on	closed	closed	on
closed	open	off	closed	open	on
open	closed	off	open	closed	on
open	open	off	open	open	off

Observe that if the words *closed* and *on* are replaced by T and *open* and *off* are replaced by F, Table 2.4.1(a) becomes the truth table for *and* and Table 2.4.1(b) becomes the truth table for *or*. Consequently, the switching circuit of Figure 2.4.2(a) is said to correspond to the logical expression $P \wedge Q$, and that of Figure 2.4.2(b) is said to correspond to $P \vee Q$.

More complicated circuits correspond to more complicated logical expressions. This correspondence has been used extensively in the design and study of circuits.

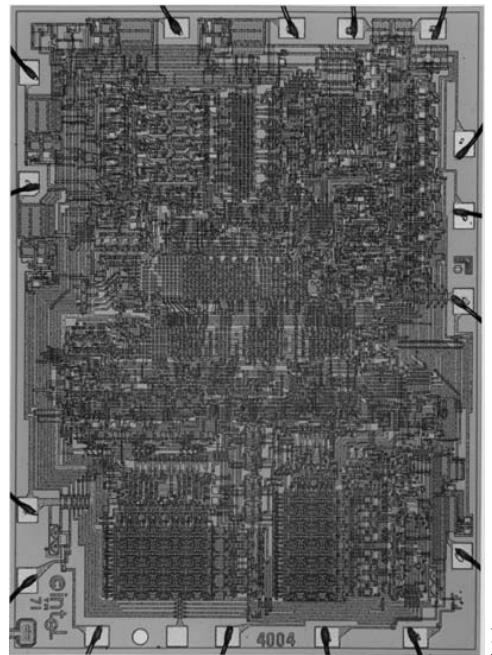
In the 1940s and 1950s, switches were replaced by electronic devices, with the physical states of closed and open corresponding to electronic states such as high and low voltages. The new electronic technology led to the development of modern digital systems such as electronic computers, electronic telephone switching systems, traffic light controls, electronic calculators, and the control mechanisms used in hundreds of other types of electronic equipment. The basic electronic components of a digital system are called *digital logic circuits*. The word *logic* indicates the important role of logic in the design of such circuits, and the word *digital* indicates that the circuits process discrete, or separate, signals as opposed to continuous ones.



John W. Tukey
(1915–2000)

Courtesy of IBM

The Intel 4004, introduced in 1971, is generally considered to be the first commercially viable microprocessor or central processing unit (CPU) contained on a chip about the size of a fingernail. It consisted of 2,300 transistors and could execute 70,000 instructions per second, essentially the same computing power as the first electronic computer, the ENIAC, built in 1946, which filled an entire room. Modern microprocessors consist of several CPUs on one chip, contain close to a billion transistors and many hundreds of millions of logic circuits, and can compute hundreds of millions of instructions per second.



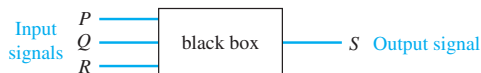
Intel

Electrical engineers continue to use the language of logic when they refer to values of signals produced by an electronic switch as being “true” or “false.” But they generally use the symbols 1 and 0 rather than T and F to denote these values. The symbols 0 and 1 are called **bits**, short for *binary digits*. This terminology was introduced in 1946 by the statistician John Tukey.

Black Boxes and Gates

Combinations of signal bits (1’s and 0’s) can be transformed into other combinations of signal bits (1’s and 0’s) by means of various circuits. Because a variety of different

technologies are used in circuit construction, computer engineers and digital system designers find it useful to think of certain basic circuits as black boxes. The inside of a black box contains the detailed implementation of the circuit and is often ignored while attention is focused on the relation between the **input** and the **output** signals.



The operation of a black box is completely specified by constructing an **input/output table** that lists all its possible input signals together with their corresponding output signals. For example, the black box pictured above has three input signals. Since each of these signals can take the value 1 or 0, there are eight possible combinations of input signals. One possible correspondence of input to output signals is as follows:

An Input/Output Table

Input			Output
<i>P</i>	<i>Q</i>	<i>R</i>	<i>S</i>
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

The third row, for instance, indicates that for inputs $P = 1$, $Q = 0$, and $R = 1$, the output S is 0.

An efficient method for designing more complicated circuits is to build them by connecting less complicated black box circuits. Three such circuits are known as NOT-, AND-, and OR-gates.

A **NOT-gate** (or **inverter**) is a circuit with one input signal and one output signal. If the input signal is 1, the output signal is 0. Conversely, if the input signal is 0, then the output signal is 1. An **AND-gate** is a circuit with two input signals and one output signal. If both input signals are 1, then the output signal is 1. Otherwise, the output signal is 0. An **OR-gate** also has two input signals and one output signal. If both input signals are 0, then the output signal is 0. Otherwise, the output signal is 1.

The actions of NOT-, AND-, and OR-gates are summarized in Figure 2.4.3, where P and Q represent input signals and R represents the output signal. It should be clear from Figure 2.4.3 that the actions of the NOT-, AND-, and OR-gates on signals correspond exactly to those of the logical connectives \sim , \wedge , and \vee on statements, if the symbol 1 is identified with T and the symbol 0 is identified with F.

Gates can be combined into circuits in a variety of ways. If the rules shown on the next page are obeyed, the result is a **combinational circuit**, one whose output at any time is determined entirely by its input at that time without regard to previous inputs.

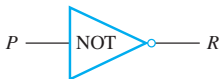

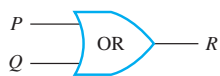
Type of Gate	Symbolic Representation	Action												
NOT		<table><tr><th>Input</th><th>Output</th></tr><tr><th>P</th><th>R</th></tr><tr><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td></tr></table>	Input	Output	P	R	1	0	0	1				
Input	Output													
P	R													
1	0													
0	1													
AND		<table><tr><th>Input</th><th>Output</th></tr><tr><th>P Q</th><th>R</th></tr><tr><td>1 1</td><td>1</td></tr><tr><td>1 0</td><td>0</td></tr><tr><td>0 1</td><td>0</td></tr><tr><td>0 0</td><td>0</td></tr></table>	Input	Output	P Q	R	1 1	1	1 0	0	0 1	0	0 0	0
Input	Output													
P Q	R													
1 1	1													
1 0	0													
0 1	0													
0 0	0													
OR		<table><tr><th>Input</th><th>Output</th></tr><tr><th>P Q</th><th>R</th></tr><tr><td>1 1</td><td>1</td></tr><tr><td>1 0</td><td>1</td></tr><tr><td>0 1</td><td>1</td></tr><tr><td>0 0</td><td>0</td></tr></table>	Input	Output	P Q	R	1 1	1	1 0	1	0 1	1	0 0	0
Input	Output													
P Q	R													
1 1	1													
1 0	1													
0 1	1													
0 0	0													

Figure 2.4.3

Rules for a Combinational Circuit

Never combine two input wires. 2.4.1

A single input wire can be split partway and used as input for two separate gates. 2.4.2

An output wire can be used as input. 2.4.3

No output of a gate can eventually feed back into that gate. 2.4.4

Rule (2.4.4) is violated in more complex circuits, called **sequential circuits**, whose output at any given time depends both on the input at that time and also on previous inputs. These circuits are discussed in Section 12.2.

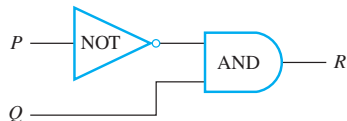
The Input/Output Table for a Circuit

If you are given a set of input signals for a circuit, you can find its output by tracing through the circuit gate by gate.

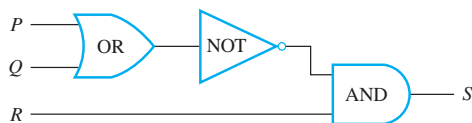
Example 2.4.1 Determining Output for a Given Input

Indicate the output of the circuits shown below for the given input signals.

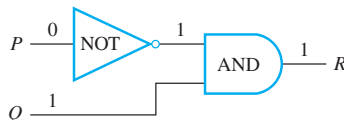
- a. Input signals: $P = 0$ and $Q = 1$



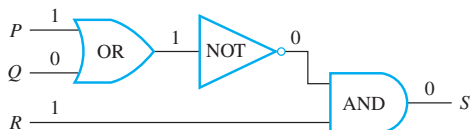
- b. Input signals: $P = 1$, $Q = 0$, $R = 1$

**Solution**

- a. Move from left to right through the diagram, tracing the action of each gate on the input signals. The NOT-gate changes $P = 0$ to a 1, so both inputs to the AND-gate are 1; hence the output R is 1. This is illustrated by annotating the diagram as shown below.



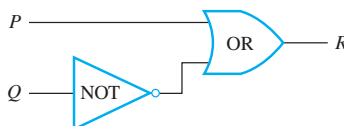
- b. The output of the OR-gate is 1 since one of the input signals, P , is 1. The NOT-gate changes this 1 into a 0, so the two inputs to the AND-gate are 0 and $R = 1$. Hence the output S is 0. The trace is shown below.

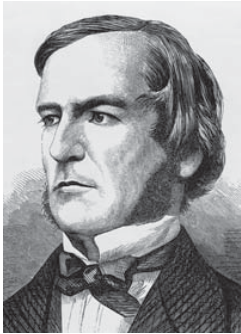


To construct the entire input/output table for a circuit, trace through the circuit to find the corresponding output signals for each possible combination of input signals.

Example 2.4.2 Constructing the Input/Output Table for a Circuit

Construct the input/output table for the following circuit.





George Boole
(1815–1864)

Solution List the four possible combinations of input signals, and find the output for each by tracing through the circuit.

Input		Output
P	Q	R
1	1	1
1	0	1
0	1	0
0	0	1

Note Strictly speaking, only meaningful expressions such as $(\sim p \wedge q) \vee (p \wedge r)$ and $\sim(\sim(p \wedge q) \vee r)$ are allowed as Boolean, not meaningless ones like $p \sim q((rs \vee \wedge q \sim)$. We use recursion to give a careful definition of Boolean expressions in Section 5.9.

The Boolean Expression Corresponding to a Circuit

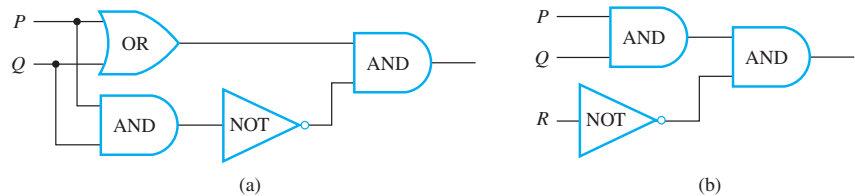
In logic, variables such as p , q and r represent statements, and a statement can have one of only two truth values: T (true) or F (false). A statement form is an expression, such as $p \wedge (\sim q \vee r)$, composed of statement variables and logical connectives.

As noted earlier, one of the founders of symbolic logic was the English mathematician George Boole. In his honor, any variable, such as a statement variable or an input signal, that can take one of only two values is called a **Boolean variable**. An expression composed of Boolean variables and the connectives \sim , \wedge , and \vee is called a **Boolean expression**.

Given a circuit consisting of combined NOT-, AND-, and OR-gates, a corresponding Boolean expression can be obtained by tracing the actions of the gates on the input variables.

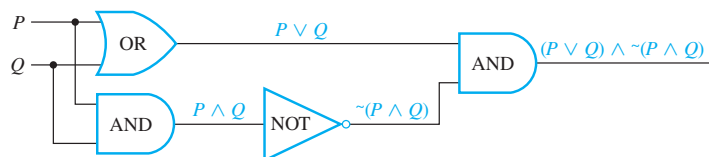
Example 2.4.3 Finding a Boolean Expression for a Circuit

Find the Boolean expressions that correspond to the circuits shown below. A dot indicates a soldering of two wires; wires that cross without a dot are assumed not to touch.



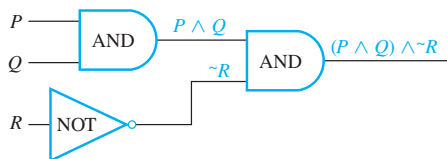
Solution

- a. Trace through the circuit from left to right, indicating the output of each gate symbolically, as shown below.



The final expression obtained, $(P \vee Q) \wedge \sim(P \wedge Q)$, is the expression for exclusive or: P or Q but not both.

- b. The Boolean expression corresponding to the circuit is $(P \wedge Q) \wedge \sim R$, as shown below.



Observe that the output of the circuit shown in Example 2.4.3(b) is 1 for exactly one combination of inputs ($P = 1$, $Q = 1$, and $R = 0$) and is 0 for all other combinations of inputs. For this reason, the circuit can be said to “recognize” one particular combination of inputs. The output column of the input/output table has a 1 in exactly one row and 0’s in all other rows.

• Definition

A **recognizer** is a circuit that outputs a 1 for exactly one particular combination of input signals and outputs 0’s for all other combinations.

Input/Output Table for a Recognizer

P	Q	R	$(P \wedge Q) \wedge \sim R$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

The Circuit Corresponding to a Boolean Expression

The preceding examples showed how to find a Boolean expression corresponding to a circuit. The following example shows how to construct a circuit corresponding to a Boolean expression.

Example 2.4.4 Constructing Circuits for Boolean Expressions

Construct circuits for the following Boolean expressions.

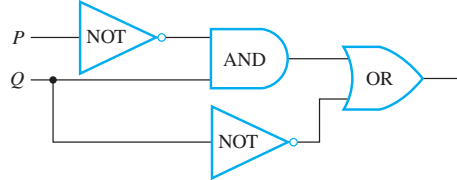
a. $(\sim P \wedge Q) \vee \sim Q$

b. $((P \wedge Q) \wedge (R \wedge S)) \wedge T$

Solution

- a. Write the input variables in a column on the left side of the diagram. Then go from the right side of the diagram to the left, working from the outermost part of the expression to the innermost part. Since the last operation executed when evaluating $(\sim P \wedge Q) \vee \sim Q$ is \vee , put an OR-gate at the extreme right of the diagram. One input to this gate is $\sim P \wedge Q$, so draw an AND-gate to the left of the OR-gate and show its

output coming into the OR-gate. Since one input to the AND-gate is $\sim P$, draw a line from P to a NOT-gate and from there to the AND-gate. Since the other input to the AND-gate is Q , draw a line from Q directly to the AND-gate. The other input to the OR-gate is $\sim Q$, so draw a line from Q to a NOT-gate and from the NOT-gate to the OR-gate. The circuit you obtain is shown below.



- b. To start constructing this circuit, put one AND-gate at the extreme right for the \wedge between $((P \wedge Q) \wedge (R \wedge S))$ and T . To the left of that put the AND-gate corresponding to the \wedge between $P \wedge Q$ and $R \wedge S$. To the left of that put the AND-gates corresponding to the \wedge 's between P and Q and between R and S . The circuit is shown in Figure 2.4.4.

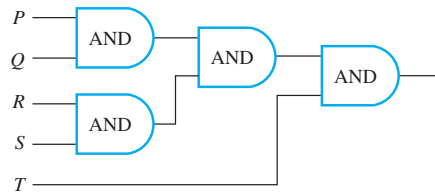


Figure 2.4.4

It follows from Theorem 2.1.1 that all the ways of adding parentheses to $P \wedge Q \wedge R \wedge S \wedge T$ are logically equivalent. Thus, for example,

$$((P \wedge Q) \wedge (R \wedge S)) \wedge T \equiv (P \wedge (Q \wedge R)) \wedge (S \wedge T).$$

It also follows that the circuit in Figure 2.4.5, which corresponds to $(P \wedge (Q \wedge R)) \wedge (S \wedge T)$, has the same input/output table as the circuit in Figure 2.4.4, which corresponds to $((P \wedge Q) \wedge (R \wedge S)) \wedge T$.

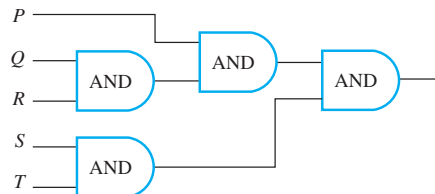


Figure 2.4.5

Each of the circuits in Figures 2.4.4 and 2.4.5 is, therefore, an implementation of the expression $P \wedge Q \wedge R \wedge S \wedge T$. Such a circuit is called a **multiple-input AND-gate** and is represented by the diagram shown in Figure 2.4.6. **Multiple-input OR-gates** are constructed similarly.

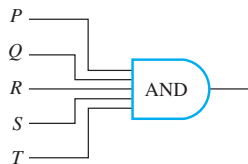


Figure 2.4.6

Finding a Circuit That Corresponds to a Given Input/Output Table

To this point, we have discussed how to construct the input/output table for a circuit, how to find the Boolean expression corresponding to a given circuit, and how to construct the circuit corresponding to a given Boolean expression. Now we address the question of how to design a circuit (or find a Boolean expression) corresponding to a given input/output table. The way to do this is to put several recognizers together in parallel.

Example 2.4.5 Designing a Circuit for a Given Input/Output Table

Design a circuit for the following input/output table:

Input			Output
P	Q	R	S
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

Solution First construct a Boolean expression with this table as its truth table. To do this, identify each row for which the output is 1—in this case, the first, third, and fourth rows. For each such row, construct an *and* expression that produces a 1 (or true) for the exact combination of input values for that row and a 0 (or false) for all other combinations of input values. For example, the expression for the first row is $P \wedge Q \wedge R$ because $P \wedge Q \wedge R$ is 1 if $P = 1$ and $Q = 1$ and $R = 1$, and it is 0 for all other values of P , Q , and R . The expression for the third row is $P \wedge \sim Q \wedge R$ because $P \wedge \sim Q \wedge R$ is 1 if $P = 1$ and $Q = 0$ and $R = 1$, and it is 0 for all other values of P , Q , and R . Similarly, the expression for the fourth row is $P \wedge \sim Q \wedge \sim R$.

Now any Boolean expression with the given table as its truth table has the value 1 in case $P \wedge Q \wedge R = 1$, or in case $P \wedge \sim Q \wedge R = 1$, or in case $P \wedge \sim Q \wedge \sim R = 1$, and in no other cases. It follows that a Boolean expression with the given truth table is

$$(P \wedge Q \wedge R) \vee (P \wedge \sim Q \wedge R) \vee (P \wedge \sim Q \wedge \sim R). \quad 2.4.5$$

The circuit corresponding to this expression has the diagram shown in Figure 2.4.7. Observe that expression (2.4.5) is a disjunction of terms that are themselves conjunctions in which one of P or $\sim P$, one of Q or $\sim Q$, and one of R or $\sim R$ all appear. Such expressions are said to be in **disjunctive normal form** or **sum-of-products form**.

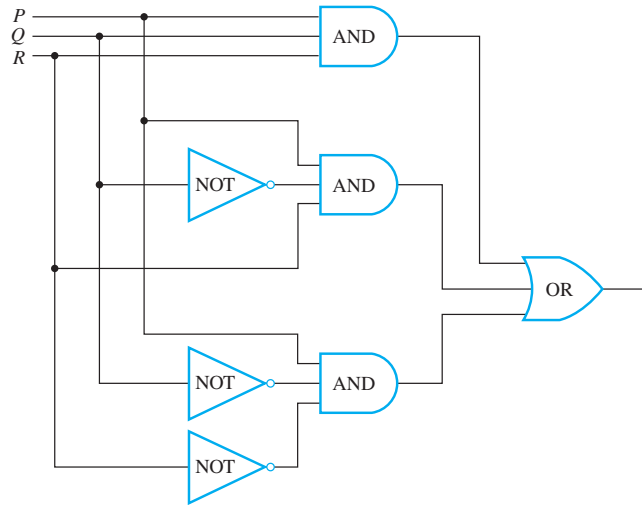


Figure 2.4.7

Simplifying Combinational Circuits

Consider the two combinational circuits shown in Figure 2.4.8.

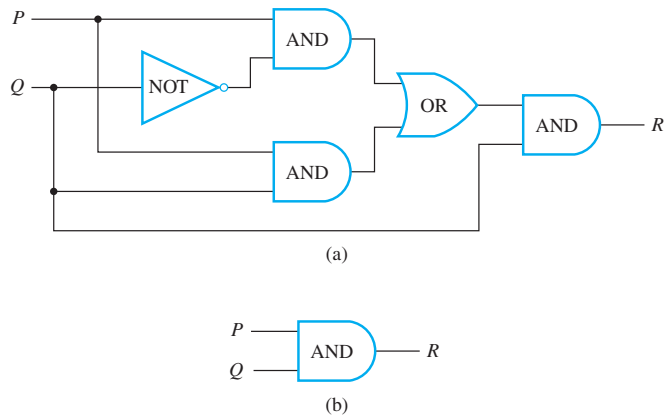


Figure 2.4.8

If you trace through circuit (a), you will find that its input/output table is

Input		Output
<i>P</i>	<i>Q</i>	<i>R</i>
1	1	1
1	0	0
0	1	0
0	0	0

which is the same as the input/output table for circuit (b). Thus these two circuits do the same job in the sense that they transform the same combinations of input signals

into the same output signals. Yet circuit (b) is simpler than circuit (a) in that it contains many fewer logic gates. Thus, as part of an integrated circuit, it would take less space and require less power.

• Definition

Two digital logic circuits are **equivalent** if, and only if, their input/output tables are identical.

Since logically equivalent statement forms have identical truth tables, you can determine that two circuits are equivalent by finding the Boolean expressions corresponding to the circuits and showing that these expressions, regarded as statement forms, are logically equivalent. Example 2.4.6 shows how this procedure works for circuits (a) and (b) in Figure 2.4.8.

Example 2.4.6 Showing That Two Circuits Are Equivalent

Find the Boolean expressions for each circuit in Figure 2.4.8. Use Theorem 2.1.1 to show that these expressions are logically equivalent when regarded as statement forms.

Solution The Boolean expressions that correspond to circuits (a) and (b) are $((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q$ and $P \wedge Q$, respectively. By Theorem 2.1.1,

$$\begin{aligned}
 & ((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q \\
 & \equiv (P \wedge (\sim Q \vee Q)) \wedge Q && \text{by the distributive law} \\
 & \equiv (P \wedge (Q \vee \sim Q)) \wedge Q && \text{by the commutative law for } \vee \\
 & \equiv (P \wedge \mathbf{t}) \wedge Q && \text{by the negation law} \\
 & \equiv P \wedge Q && \text{by the identity law.}
 \end{aligned}$$

It follows that the truth tables for $((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q$ and $P \wedge Q$ are the same. Hence the input/output tables for the circuits corresponding to these expressions are also the same, and so the circuits are equivalent. ■

In general, you can simplify a combinational circuit by finding the corresponding Boolean expression, using the properties listed in Theorem 2.1.1 to find a Boolean expression that is shorter and logically equivalent to it (when both are regarded as statement forms), and constructing the circuit corresponding to this shorter Boolean expression.

NAND and NOR Gates





Harvard University Archives

H. M. Sheffer
(1882–1964)

Another way to simplify a circuit is to find an equivalent circuit that uses the least number of different kinds of logic gates. Two gates not previously introduced are particularly useful for this: NAND-gates and NOR-gates. A NAND-gate is a single gate that acts like an AND-gate followed by a NOT-gate. A NOR-gate acts like an OR-gate followed by a NOT-gate. Thus the output signal of a NAND-gate is 0 when, and only when, both input signals are 1, and the output signal for a NOR-gate is 1 when, and only when, both input signals are 0. The logical symbols corresponding to these gates are $|$ (for NAND) and \downarrow (for NOR), where $|$ is called a **Sheffer stroke** (after H. M. Sheffer, 1882–1964) and \downarrow is called a **Peirce arrow** (after C. S. Peirce, 1839–1914; see page 101). Thus

$$P | Q \equiv \sim(P \wedge Q) \quad \text{and} \quad P \downarrow Q \equiv \sim(P \vee Q).$$

The table below summarizes the actions of NAND and NOR gates.

Type of Gate	Symbolic Representation	Action																		
NAND		<table> <tr> <th colspan="2">Input</th><th>Output</th></tr> <tr> <th>P</th><th>Q</th><th>$R = P \mid Q$</th></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> </table>	Input		Output	P	Q	$R = P \mid Q$	1	1	0	1	0	1	0	1	1	0	0	1
Input		Output																		
P	Q	$R = P \mid Q$																		
1	1	0																		
1	0	1																		
0	1	1																		
0	0	1																		
NOR		<table> <tr> <th colspan="2">Input</th><th>Output</th></tr> <tr> <th>P</th><th>Q</th><th>$R = P \downarrow Q$</th></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> </table>	Input		Output	P	Q	$R = P \downarrow Q$	1	1	0	1	0	0	0	1	0	0	0	1
Input		Output																		
P	Q	$R = P \downarrow Q$																		
1	1	0																		
1	0	0																		
0	1	0																		
0	0	1																		

It can be shown that any Boolean expression is equivalent to one written entirely with Sheffer strokes or entirely with Peirce arrows. Thus any digital logic circuit is equivalent to one that uses only NAND-gates or only NOR-gates. Example 2.4.7 develops part of the derivation of this result; the rest is left for the exercises.

Example 2.4.7 Rewriting Expressions Using the Sheffer Stroke

Use Theorem 2.1.1 and the definition of Sheffer stroke to show that

$$\text{a. } \sim P \equiv P \mid P \quad \text{and} \quad \text{b. } P \vee Q \equiv (P \mid P) \mid (Q \mid Q).$$

Solution

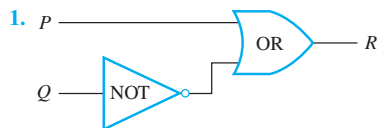
$$\begin{aligned}
 \text{a. } \sim P &\equiv \sim(P \wedge P) && \text{by the idempotent law for } \wedge \\
 &\equiv P \mid P && \text{by definition of } \mid. \\
 \text{b. } P \vee Q &\equiv \sim(\sim(P \vee Q)) && \text{by the double negative law} \\
 &\equiv \sim(\sim P \wedge \sim Q) && \text{by De Morgan's laws} \\
 &\equiv \sim((P \mid P) \wedge (Q \mid Q)) && \text{by part (a)} \\
 &\equiv (P \mid P) \mid (Q \mid Q) && \text{by definition of } \mid.
 \end{aligned}$$

Test Yourself

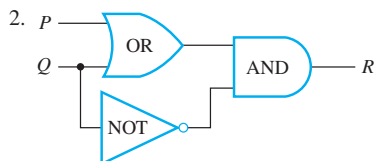
- The input/output table for a digital logic circuit is a table that shows ____.
- The Boolean expression that corresponds to a digital logic circuit is ____.
- A recognizer is a digital logic circuit that ____.
- Two digital logic circuits are equivalent if, and only if, ____.
- A NAND-gate is constructed by placing a ____ gate immediately following an ____ gate.
- A NOR-gate is constructed by placing a ____ gate immediately following an ____ gate.

Exercise Set 2.4

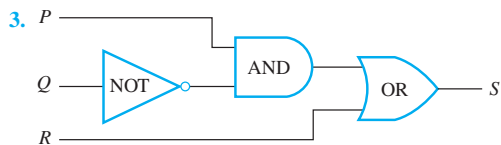
Give the output signals for the circuits in 1–4 if the input signals are as indicated.



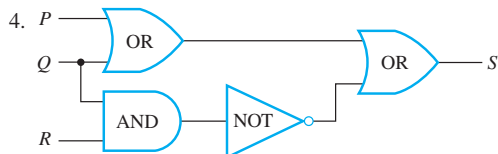
input signals: $P = 1$ and $Q = 1$



input signals: $P = 1$ and $Q = 0$



input signals: $P = 1$, $Q = 0$, $R = 0$



input signals: $P = 0$, $Q = 0$, $R = 0$

In 5–8, write an input/output table for the circuit in the referenced exercise.

5. Exercise 1

6. Exercise 2

7. Exercise 3

8. Exercise 4

In 9–12, find the Boolean expression that corresponds to the circuit in the referenced exercise.

9. Exercise 1

10. Exercise 2

11. Exercise 3

12. Exercise 4

Construct circuits for the Boolean expressions in 13–17.

13. $\sim P \vee Q$

14. $\sim(P \vee Q)$

15. $P \vee (\sim P \wedge \sim Q)$

16. $(P \wedge Q) \vee \sim R$

17. $(P \wedge \sim Q) \vee (\sim P \wedge R)$

For each of the tables in 18–21, construct (a) a Boolean expression having the given table as its truth table and (b) a circuit having the given table as its input/output table.

18.

P	Q	R	S
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

19.

P	Q	R	S
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	0

20.

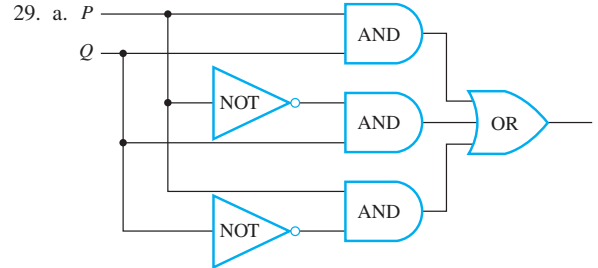
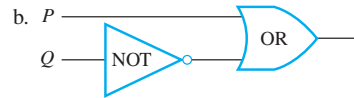
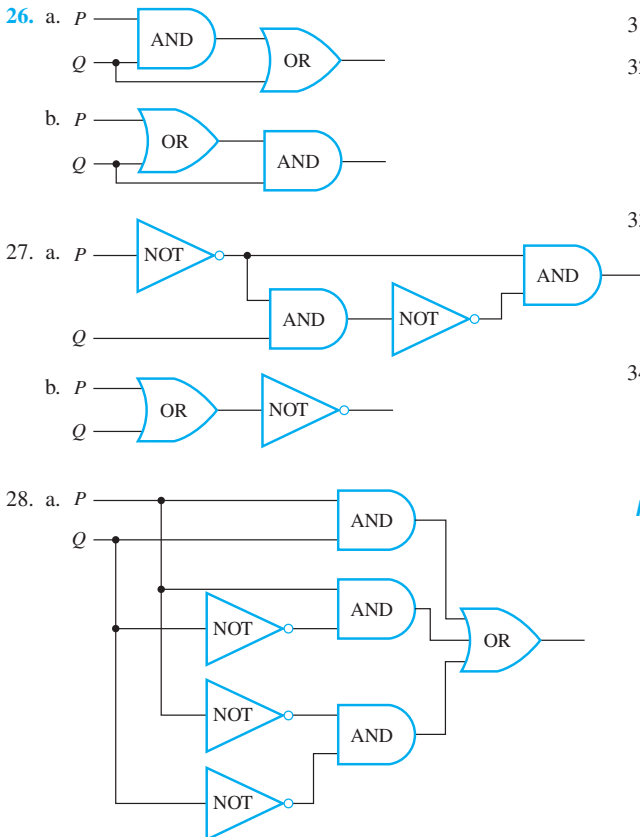
P	Q	R	S
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	1

21.

P	Q	R	S
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	0

22. Design a circuit to take input signals P , Q , and R and output a 1 if, and only if, P and Q have the same value and Q and R have opposite values.
23. Design a circuit to take input signals P , Q , and R and output a 1 if, and only if, all three of P , Q , and R have the same value.
24. The lights in a classroom are controlled by two switches: one at the back and one at the front of the room. Moving either switch to the opposite position turns the lights off if they are on and on if they are off. Assume the lights have been installed so that when both switches are in the down position, the lights are off. Design a circuit to control the switches.
25. An alarm system has three different control panels in three different locations. To enable the system, switches in at least two of the panels must be in the on position. If fewer than two are in the on position, the system is disabled. Design a circuit to control the switches.

Use the properties listed in Theorem 2.1.1 to show that each pair of circuits in 26–29 have the same input/output table. (Find the Boolean expressions for the circuits and show that they are logically equivalent when regarded as statement forms.)



For the circuits corresponding to the Boolean expressions in each of 30 and 31 there is an equivalent circuit with at most two logic gates. Find such a circuit.

30. $(P \wedge Q) \vee (\sim P \wedge Q) \vee (\sim P \wedge \sim Q)$

31. $(\sim P \wedge \sim Q) \vee (\sim P \wedge Q) \vee (P \wedge \sim Q)$

32. The Boolean expression for the circuit in Example 2.4.5 is

$$(P \wedge Q \wedge R) \vee (P \wedge \sim Q \wedge R) \vee (P \wedge \sim Q \wedge \sim R)$$

(a disjunctive normal form). Find a circuit with at most three logic gates that is equivalent to this circuit.

33. a. Show that for the Sheffer stroke $|$,

$$P \wedge Q \equiv (P | Q) | (P | Q).$$

- b. Use the results of Example 2.4.7 and part (a) above to write $P \wedge (\sim Q \vee R)$ using only Sheffer strokes.

34. Show that the following logical equivalences hold for the Peirce arrow \downarrow , where $P \downarrow Q \equiv \sim(P \vee Q)$.

a. $\sim P \equiv P \downarrow P$

b. $P \vee Q \equiv (P \downarrow Q) \downarrow (P \downarrow Q)$

c. $P \wedge Q \equiv (P \downarrow P) \downarrow (Q \downarrow Q)$

H d. Write $P \rightarrow Q$ using Peirce arrows only.

e. Write $P \leftrightarrow Q$ using Peirce arrows only.

Answers for Test Yourself

- the output signal(s) that correspond to all possible combinations of input signals to the circuit
- a Boolean expression that represents the input signals as variables and indicates the successive actions of the logic gates on the input signals
- outputs a 1 for exactly one particular combination of input signals and outputs 0's for all other combinations
- they have the same input/output table
- NOT; AND
- NOT; OR

2.5 Application: Number Systems and Circuits for Addition

Counting in binary is just like counting in decimal if you are all thumbs. — Glaser and Way

In elementary school, you learned the meaning of decimal notation: that to interpret a string of decimal digits as a number, you mentally multiply each digit by its place value. For instance, 5,049 has a 5 in the thousands place, a 0 in the hundreds place, a 4 in the tens place, and a 9 in the ones place. Thus

$$5,049 = 5 \cdot (1,000) + 0 \cdot (100) + 4 \cdot (10) + 9 \cdot (1).$$

Using exponential notation, this equation can be rewritten as

$$5,049 = 5 \cdot 10^3 + 0 \cdot 10^2 + 4 \cdot 10^1 + 9 \cdot 10^0.$$

More generally, decimal notation is based on the fact that any positive integer can be written uniquely as a sum of products of the form

$$d \cdot 10^n,$$

where each n is a nonnegative integer and each d is one of the decimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. The word *decimal* comes from the Latin root *deci*, meaning “ten.” Decimal (or base 10) notation expresses a number as a string of digits in which each digit’s position indicates the power of 10 by which it is multiplied. The right-most position is the ones place (or 10^0 place), to the left of that is the tens place (or 10^1 place), to the left of that is the hundreds place (or 10^2 place), and so forth, as illustrated below.

Place	10^3 thousands	10^2 hundreds	10^1 tens	10^0 ones
Decimal Digit	5	0	4	9

Binary Representation of Numbers

There is nothing sacred about the number 10; we use 10 as a base for our usual number system because we happen to have ten fingers. In fact, any integer greater than 1 can serve as a base for a number system. In computer science, **base 2 notation**, or **binary notation**, is of special importance because the signals used in modern electronics are always in one of only two states. (The Latin root *bi* means “two.”)

In Section 5.4, we show that any integer can be represented uniquely as a sum of products of the form

$$d \cdot 2^n,$$

where each n is an integer and each d is one of the binary digits (or bits) 0 or 1. For example,

$$\begin{aligned} 27 &= 16 + 8 + 2 + 1 \\ &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0. \end{aligned}$$

In binary notation, as in decimal notation, we write just the binary digits, and not the powers of the base. In binary notation, then,

$$27_{10} = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1\ 1\ 0\ 1\ 1_2$$

where the subscripts indicate the base, whether 10 or 2, in which the number is written. The places in binary notation correspond to the various powers of 2. The right-most position is the ones place (or 2^0 place), to the left of that is the twos place (or 2^1 place), to the left of that is the fours place (or 2^2 place), and so forth, as illustrated below.

Place	2^4 sixteens	2^3 eights	2^2 fours	2^1 twos	2^0 ones
Binary Digit	1	1	0	1	1

As in the decimal notation, leading zeros may be added or dropped as desired. For example,

$$003_{10} = 3_{10} = 1 \cdot 2^1 + 1 \cdot 2^0 = 11_2 = 011_2.$$

Example 2.5.1 Binary Notation for Integers from 1 to 9

Derive the binary notation for the integers from 1 to 9.

Solution

1_{10}	$=$	$1 \cdot 2^0$	$=$	1_2
2_{10}	$=$	$1 \cdot 2^1 + 0 \cdot 2^0$	$=$	10_2
3_{10}	$=$	$1 \cdot 2^1 + 1 \cdot 2^0$	$=$	11_2
4_{10}	$=$	$1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	$=$	100_2
5_{10}	$=$	$1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	$=$	101_2
6_{10}	$=$	$1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$	$=$	110_2
7_{10}	$=$	$1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	$=$	111_2
8_{10}	$=$	$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	$=$	1000_2
9_{10}	$=$	$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	$=$	1001_2



A list of powers of 2 is useful for doing binary-to-decimal and decimal-to-binary conversions. See Table 2.5.1.

Table 2.5.1 Powers of 2

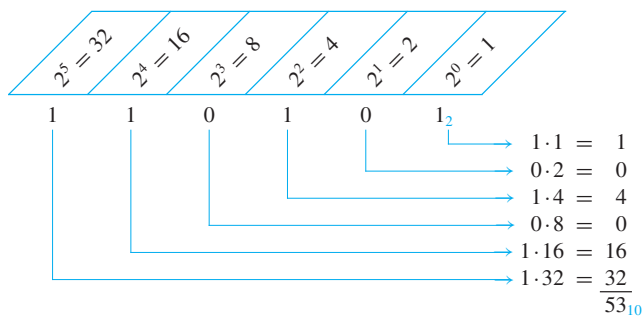
Power of 2	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Form	1024	512	256	128	64	32	16	8	4	2	1

Example 2.5.2 Converting a Binary to a Decimal Number

Represent 110101_2 in decimal notation.

$$\begin{aligned}
 \text{Solution} \quad 110101_2 &= 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\
 &= 32 + 16 + 4 + 1 \\
 &= 53_{10}
 \end{aligned}$$

Alternatively, the schema below may be used.

**Example 2.5.3 Converting a Decimal to a Binary Number**

Represent 209 in binary notation.

Solution Use Table 2.5.1 to write 209 as a sum of powers of 2, starting with the highest power of 2 that is less than 209 and continuing to lower powers.

Since 209 is between 128 and 256, the highest power of 2 that is less than 209 is 128. Hence

$$209_{10} = 128 + \text{a smaller number.}$$

Now $209 - 128 = 81$, and 81 is between 64 and 128, so the highest power of 2 that is less than 81 is 64. Hence

$$209_{10} = 128 + 64 + \text{a smaller number.}$$

Continuing in this way, you obtain

$$\begin{aligned}
 209_{10} &= 128 + 64 + 16 + 1 \\
 &= 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0.
 \end{aligned}$$

For each power of 2 that occurs in the sum, there is a 1 in the corresponding position of the binary number. For each power of 2 that is missing from the sum, there is a 0 in the corresponding position of the binary number. Thus

$$209_{10} = 11010001_2$$

Another procedure for converting from decimal to binary notation is discussed in Section 5.1.



Caution! Do not read 10_2 as “ten”; it is the number two. Read 10_2 as “one oh base two.”

Binary Addition and Subtraction

The computational methods of binary arithmetic are analogous to those of decimal arithmetic. In binary arithmetic the number 2 ($= 10_2$ in binary notation) plays a role similar to that of the number 10 in decimal arithmetic.

Example 2.5.4 Addition in Binary Notation

Add 1101_2 and 111_2 using binary notation.

Solution Because $2_{10} = 10_2$ and $1_{10} = 1_2$, the translation of $1_{10} + 1_{10} = 2_{10}$ to binary notation is

$$\begin{array}{r} 1_2 \\ + 1_2 \\ \hline 10_2 \end{array}$$

It follows that adding two 1's together results in a carry of 1 when binary notation is used. Adding three 1's together also results in a carry of 1 since $3_{10} = 11_2$ ("one one base two").

$$\begin{array}{r} 1_2 \\ + 1_2 \\ + 1_2 \\ \hline 11_2 \end{array}$$

Thus the addition can be performed as follows:

$$\begin{array}{rccccccc} & 1 & 1 & 1 & & & \leftarrow \text{carry row} \\ & 1 & 1 & 0 & 1_2 & & \\ + & & & 1 & 1 & 1_2 & \\ \hline 1 & 0 & 1 & 0 & 0_2 & & \end{array}$$

Example 2.5.5 Subtraction in Binary Notation

Subtract 1011_2 from 11000_2 using binary notation.

Solution In decimal subtraction the fact that $10_{10} - 1_{10} = 9_{10}$ is used to borrow across several columns. For example, consider the following:

$$\begin{array}{rccccccc} & 9 & 9 & & & & \\ & \swarrow & \swarrow & 1 & & & \leftarrow \text{borrow row} \\ 1 & 0 & 0 & 0 & 0_{10} & & \\ - & & 5 & 8_{10} & & & \\ \hline & 9 & 4 & 2_{10} & & & \end{array}$$

In binary subtraction it may also be necessary to borrow across more than one column. But when you borrow a 1_2 from 10_2 , what remains is 1_2 .

$$\begin{array}{r} 10_2 \\ - 1_2 \\ \hline 1_2 \end{array}$$

Thus the subtraction can be performed as follows:

$$\begin{array}{rccccccc} & 0 & 1 & 1 & & & \\ & \swarrow & \swarrow & \swarrow & 1 & & \leftarrow \text{borrow row} \\ 1 & 1 & 0 & 0 & 0_2 & & \\ - & & 1 & 0 & 1 & 1_2 & \\ \hline & 1 & 1 & 0 & 1_2 & & \end{array}$$

Circuits for Computer Addition

Consider the question of designing a circuit to produce the sum of two binary digits P and Q . Both P and Q can be either 0 or 1. And the following facts are known:

$$1_2 + 1_2 = 10_2,$$

$$1_2 + 0_2 = 1_2 = 01_2,$$

$$0_2 + 1_2 = 1_2 = 01_2,$$

$$0_2 + 0_2 = 0_2 = 00_2.$$

It follows that the circuit to be designed must have two outputs—one for the left binary digit (this is called the **carry**) and one for the right binary digit (this is called the **sum**). The carry output is 1 if both P and Q are 1; it is 0 otherwise. Thus the carry can be produced using the AND-gate circuit that corresponds to the Boolean expression $P \wedge Q$. The sum output is 1 if either P or Q , but not both, is 1. The sum can, therefore, be produced using a circuit that corresponds to the Boolean expression for *exclusive or*: $(P \vee Q) \wedge \sim(P \wedge Q)$. (See Example 2.4.3(a).) Hence, a circuit to add two binary digits P and Q can be constructed as in Figure 2.5.1. This circuit is called a **half-adder**.

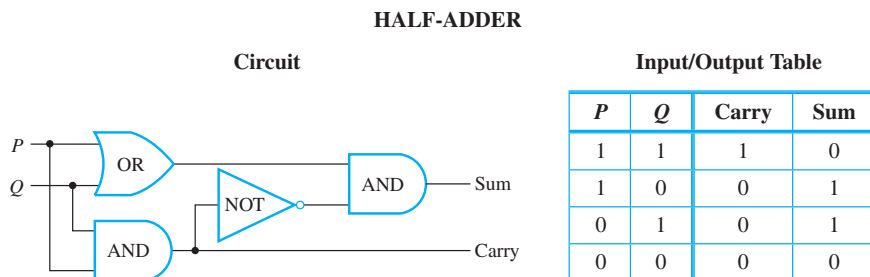


Figure 2.5.1 Circuit to Add $P + Q$, Where P and Q Are Binary Digits

Now consider the question of how to construct a circuit to add two binary integers, each with more than one digit. Because the addition of two binary digits may result in a carry to the next column to the left, it may be necessary to add three binary digits at certain points. In the following example, the sum in the right column is the sum of two binary digits, and, because of the carry, the sum in the left column is the sum of three binary digits.

$$\begin{array}{r}
 1 \quad \leftarrow \text{carry row} \\
 1 \quad 1_2 \\
 + 1 \quad 1_2 \\
 \hline
 1 \quad 1 \quad 0_2
 \end{array}$$

Thus, in order to construct a circuit that will add multidigit binary numbers, it is necessary to incorporate a circuit that will compute the sum of three binary digits. Such a circuit is called a **full-adder**. Consider a general addition of three binary digits P , Q , and R that results in a carry (or left-most digit) C and a sum (or right-most digit) S .

$$\begin{array}{r}
 P \\
 + Q \\
 + R \\
 \hline
 CS
 \end{array}$$

The operation of the full-adder is based on the fact that addition is a binary operation: Only two numbers can be added at one time. Thus P is first added to Q and then the

result is added to R . For instance, consider the following addition:

$$\begin{array}{r} 1_2 \\ + 0_2 \\ + 1_2 \\ \hline 10_2 \end{array} \left\{ \begin{array}{l} 1_2 + 0_2 = 01_2 \\ 1_2 + 1_2 = 10_2 \end{array} \right.$$

The process illustrated here can be broken down into steps that use half-adder circuits.

Step 1: Add P and Q using a half-adder to obtain a binary number with two digits.

$$\begin{array}{r} P \\ + Q \\ \hline C_1 S_1 \end{array}$$

Step 2: Add R to the sum $C_1 S_1$ of P and Q .

$$\begin{array}{r} C_1 S_1 \\ + R \\ \hline \end{array}$$

To do this, proceed as follows:

Step 2a: Add R to S_1 using a half-adder to obtain the two-digit number $C_2 S$.

$$\begin{array}{r} S_1 \\ + R \\ \hline C_2 S \end{array}$$

Then S is the right-most digit of the entire sum of P , Q , and R .

Step 2b: Determine the left-most digit, C , of the entire sum as follows: First note that it is impossible for both C_1 and C_2 to be 1's. For if $C_1 = 1$, then P and Q are both 1, and so $S_1 = 0$. Consequently, the addition of S_1 and R gives a binary number $C_2 S_1$ where $C_2 = 0$. Next observe that C will be a 1 in the case that the addition of P and Q gives a carry of 1 or in the case that the addition of S_1 (the right-most digit of $P + Q$) and R gives a carry of 1. In other words, $C = 1$ if, and only if, $C_1 = 1$ or $C_2 = 1$. It follows that the circuit shown in Figure 2.5.2 will compute the sum of three binary digits.

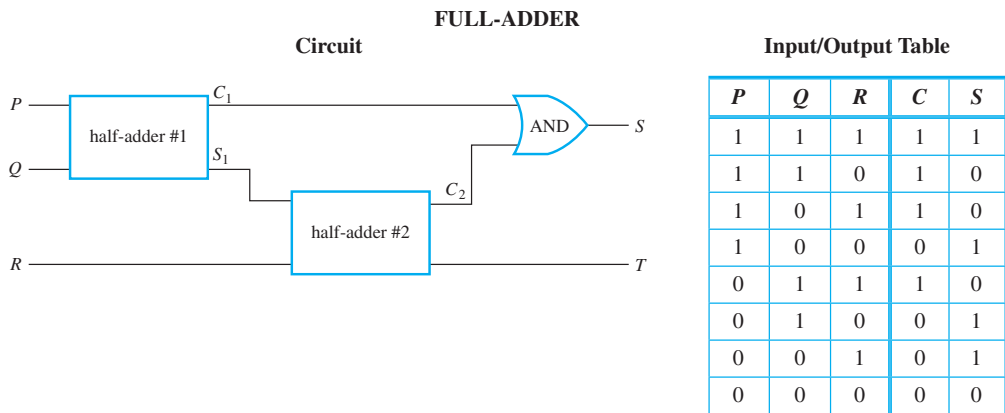


Figure 2.5.2 Circuit to Add $P + Q + R$, Where P , Q , and R Are Binary Digits

Two full-adders and one half-adder can be used together to build a circuit that will add two three-digit binary numbers PQR and STU to obtain the sum $WXYZ$. This is illustrated in Figure 2.5.3. Such a circuit is called a **parallel adder**. Parallel adders can be constructed to add binary numbers of any finite length.

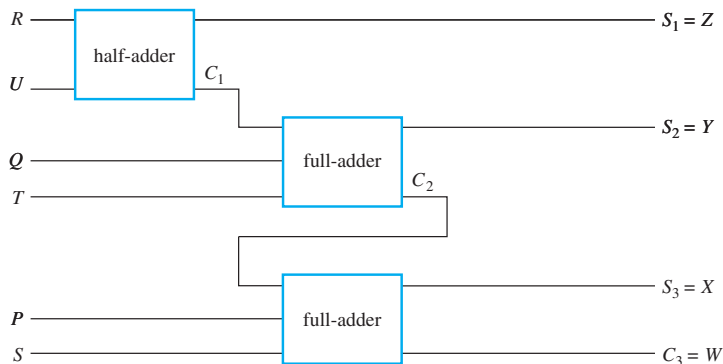


Figure 2.5.3 A Parallel Adder to Add PQR and STU to Obtain $WXYZ$

Two's Complements and the Computer Representation of Negative Integers

Typically, a fixed number of bits is used to represent integers on a computer, and these are required to represent negative as well as nonnegative integers. Sometimes a particular bit, normally the left-most, is used as a sign indicator, and the remaining bits are taken to be the absolute value of the number in binary notation. The problem with this approach is that the procedures for adding the resulting numbers are somewhat complicated and the representation of 0 is not unique. A more common approach, using *two's complements*, makes it possible to add integers quite easily and results in a unique representation for 0. The two's complement of an integer relative to a fixed bit length is defined as follows:

• Definition

Given a positive integer a , the **two's complement of a relative to a fixed bit length n** is the n -bit binary representation of

$$2^n - a.$$

Bit lengths of 16 and 32 are the most commonly used in practice. However, because the principles are the same for all bit lengths, we use a bit length of 8 for simplicity in this discussion. For instance, because

$$(2^8 - 27)_{10} = (256 - 27)_{10} = 229_{10} = (128 + 64 + 32 + 4 + 1)_{10} = 11100101_2,$$

the 8-bit two's complement of 27 is 11100101₂.

It turns out that there is a convenient way to compute two's complements that involves less arithmetic than direct application of the definition. For an 8-bit representation, it is based on three facts:

1. $2^8 - a = [(2^8 - 1) - a] + 1$.
2. The binary representation of $2^8 - 1$ is 11111111_2 .
3. Subtracting an 8-bit binary number a from 11111111_2 just switches all the 0's in a to 1's and all the 1's to 0's. (The resulting number is called the **one's complement** of the given number.)

For instance, by (2) and (3), with $a = 27$,

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} & 2^8 - 1 \\
 - & \\
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline \end{array} & 27 \\
 \hline
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ \hline \end{array} & (2^8 - 1) - 27
 \end{array}
 \quad 2.5.1$$

0's and 1's are switched

and so in binary notation the difference $(2^8 - 1) - 27$ is 11100100_2 . But by (1) with $a = 27$, $2^8 - 27 = [(2^8 - 1) - 27] + 1$, and so if we add 1 to (2.5.1), we obtain the 8-bit binary representation of $2^8 - 27$, which is the 8-bit two's complement of 27:

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ \hline \end{array} & (2^8 - 1) - 27 \\
 + & \\
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array} & 1 \\
 \hline
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array} & 2^8 - 27
 \end{array}$$

In general,

To find the 8-bit two's complement of a positive integer a that is at most 255:

- Write the 8-bit binary representation for a .
- Flip the bits (that is, switch all the 1's to 0's and all the 0's to 1's).
- Add 1 in binary notation.

Example 2.5.6 Finding a Two's Complement

Find the 8-bit two's complement of 19.

Solution Write the 8-bit binary representation for 19, switch all the 0's to 1's and all the 1's to 0's, and add 1.

$$19_{10} = (16 + 2 + 1)_{10} = 00010011_2 \xrightarrow{\text{flip the bits}} 11101100 \xrightarrow{\text{add 1}} 11101101$$

To check this result, note that

$$\begin{aligned}
 11101101_2 &= (128 + 64 + 32 + 8 + 4 + 1)_{10} = 237_{10} = (256 - 19)_{10} \\
 &= (2^8 - 19)_{10},
 \end{aligned}$$

which is the two's complement of 19. ■

Observe that because

$$2^8 - (2^8 - a) = a$$

the *two's complement of the two's complement of a number is the number itself*, and therefore,

To find the decimal representation of the integer with a given 8-bit two's complement:

- Find the two's complement of the given two's complement.
- Write the decimal equivalent of the result.

Example 2.5.7 Finding a Number with a Given Two's Complement

What is the decimal representation for the integer with two's complement 10101001?

Solution

$$10101001_2 \xrightarrow{\text{flip the bits}} 01010110 \xrightarrow{\text{add 1}} 01010111_2 = (64 + 16 + 4 + 2 + 1)_{10} = 87_{10}$$

To check this result, note that the given number is

$$10101001_2 = (128 + 32 + 8 + 1)_{10} = 169_{10} = (256 - 87)_{10} = (2^8 - 87)_{10},$$

which is the two's complement of 87. ■

8-Bit Representation of a Number

Now consider the two's complement of an integer n that satisfies the inequality $1 \leq n \leq 128$. Then

$$-1 \geq -n \geq -128 \quad \text{because multiplying by } -1 \text{ reverses the direction of the inequality}$$

and

$$2^8 - 1 \geq 2^8 - n \geq 2^8 - 128 \quad \text{by adding } 2^8 \text{ to all parts of the inequality.}$$

But $2^8 - 128 = 256 - 128 = 128 = 2^7$. Hence

$$2^7 \leq \text{the two's complement of } n < 2^8.$$

It follows that the 8-bit two's complement of an integer from 1 through 128 has a leading bit of 1. Note also that the ordinary 8-bit representation of an integer from 0 through 127 has a leading bit of 0. Consequently, eight bits can be used to represent both nonnegative and negative integers by representing each nonnegative integer up through 127 using ordinary 8-bit binary notation and representing each negative integer from -1 through -128 as the two's complement of its absolute value. That is, for any integer a from -128 through 127,

The 8-bit representation of a

$$= \begin{cases} \text{the 8-bit binary representation of } a & \text{if } a \geq 0 \\ \text{the 8-bit binary representation of } 2^8 - |a| & \text{if } a < 0 \end{cases}.$$

The representations are illustrated in Table 2.5.2.

Table 2.5.2

Integer	8-Bit Representation (ordinary 8-bit binary notation if nonnegative or 8-bit two's complement of absolute value if negative)	Decimal Form of Two's Complement for Negative Integers
127	01111111	
126	01111110	
\vdots	\vdots	
2	00000010	
1	00000001	
0	00000000	
-1	11111111	$2^8 - 1$
-2	11111110	$2^8 - 2$
-3	11111101	$2^8 - 3$
\vdots	\vdots	\vdots
-127	10000001	$2^8 - 127$
-128	10000000	$2^8 - 128$

Computer Addition with Negative Integers

Here is an example of how two's complements enable addition circuits to perform subtraction. Suppose you want to compute $72 - 54$. First note that this is the same as $72 + (-54)$, and the 8-bit binary representations of 72 and -54 are 01001000 and 11001010, respectively. So if you add the 8-bit binary representations for both numbers, you get

$$\begin{array}{r}
 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0 \\
 +\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0 \\
 \hline
 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0
 \end{array}$$

And if you truncate the leading 1, you get 00010010. This is the 8-bit binary representation for 18, which is the right answer!

The description below explains how to use this method to add any two integers between -128 and 127. It is easily generalized to apply to 16-bit and 32-bit representations in order to add integers between about -2,000,000,000 and 2,000,000,000.

To add two integers in the range -128 through 127 whose sum is also in the range -128 through 127:

- Convert both integers to their 8-bit representations (representing negative integers by using the two's complements of their absolute values).
- Add the resulting integers using ordinary binary addition.
- Truncate any leading 1 (overflow) that occurs in the 2^8 th position.
- Convert the result back to decimal form (interpreting 8-bit integers with leading 0's as nonnegative and 8-bit integers with leading 1's as negative).

To see why this result is true, consider four cases: (1) both integers are nonnegative, (2) one integer is nonnegative and the other is negative and the absolute value of the nonnegative integer is less than that of the negative one, (3) one integer is nonnegative and the other is negative and the absolute value of the negative integer is less than or equal to that of the nonnegative one, and (4) both integers are negative.

Case 1, (both integers are nonnegative): This case is easy because if two nonnegative integers from 0 through 127 are written in their 8-bit representations and if their sum is also in the range 0 through 127, then the 8-bit representation of their sum has a leading 0 and is therefore interpreted correctly as a nonnegative integer. The example below illustrates what happens when 38 and 69 are added.

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline \end{array} & 38 \\
 + & \\
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array} & 69 \\
 \hline
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline \end{array} & 107
 \end{array}$$

Cases (2) and (3) both involve adding a negative and a nonnegative integer. To be concrete, let the nonnegative integer be a and the negative integer be $-b$ and suppose both a and $-b$ are in the range -128 through 127. The crucial observation is that adding the 8-bit representations of a and $-b$ is equivalent to computing

$$a + (2^8 - b)$$

because the 8-bit representation of $-b$ is the binary representation of $2^8 - b$.

Case 2 (a is nonnegative and $-b$ is negative and $|a| < |b|$): In this case, observe that $a = |a| < |b| = b$ and

$$a + (2^8 - b) = 2^8 - (b - a),$$

and the binary representation of this number is the 8-bit representation of $-(b - a) = a + (-b)$. We must be careful to check that $2^8 - (b - a)$ is between 2^7 and 2^8 . But it is because

$$2^7 = 2^8 - 2^7 \leq 2^8 - (b - a) < 2^8 \quad \text{since } 0 < b - a \leq b \leq 128 = 2^7.$$

Hence in case $|a| < |b|$, adding the 8-bit representations of a and $-b$ gives the 8-bit representation of $a + (-b)$.

Example 2.5.8 Computing $a + (-b)$ Where $0 \leq a < b \leq 128$

Use 8-bit representations to compute $39 + (-89)$.

Solution

Step 1: Change from decimal to 8-bit representations using the two's complement to represent -89 .

Since $39_{10} = (32 + 4 + 2 + 1)_{10} = 100111_2$, the 8-bit representation of 39 is 00100111. Now the 8-bit representation of -89 is the two's complement of 89. This is obtained as follows:

$$\begin{array}{lcl}
 89_{10} = (64 + 16 + 8 + 1)_{10} = 01011001_2 & \xrightarrow{\text{flip the bits}} & 10100110 \\
 & & \xrightarrow{\text{add 1}} 10100111
 \end{array}$$

So the 8-bit representation of -89 is 10100111.

Step 2: Add the 8-bit representations in binary notation and truncate the 1 in the 2^8 th position if there is one:

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ \hline \end{array} \\
 + \\
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ \hline \end{array} \\
 \hline
 \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline \end{array}
 \end{array}$$

There is no 1 in the 2^8 th position to truncate \rightarrow

Step 3: Find the decimal equivalent of the result. Since its leading bit is 1, this number is the 8-bit representation of a negative integer.

$$\begin{array}{lcl}
 11001110 & \xrightarrow{\text{flip the bits}} & 00110001 \\
 & & \xrightarrow{\text{add 1}} \quad 00110010 \\
 & & \Leftrightarrow -(32 + 16 + 2)_{10} = -50_{10}
 \end{array}$$

Note that since $39 - 89 = -50$, this procedure gives the correct answer. ■

Case 3 (a is nonnegative and $-b$ is negative and $|b| \leq |a|$): In this case, observe that $b = |b| \leq |a| = a$ and

$$a + (2^8 - b) = 2^8 + (a - b).$$

Also

$$2^8 \leq 2^8 + (a - b) < 2^8 + 2^7 \quad \text{because } 0 \leq a - b \leq a < 128 = 2^7.$$

So the binary representation of $a + (2^8 - b) = 2^8 + (a - b)$ has a leading 1 in the ninth (2^8 th) position. This leading 1 is often called “overflow” because it does not fit in the 8-bit integer format. Now subtracting 2^8 from $2^8 + (a - b)$ is equivalent to truncating the leading 1 in the 2^8 th position of the binary representation of the number. But

$$[a + (2^8 - b)] - 2^8 = 2^8 + (a - b) - 2^8 = a - b = a + (-b).$$

Hence in case $|a| \geq |b|$, adding the 8-bit representations of a and $-b$ and truncating the leading 1 (which is sure to be present) gives the 8-bit representation of $a + (-b)$.

Example 2.5.9 Computing $a + (-b)$ Where $1 \leq b \leq a \leq 127$

Use 8-bit representations to compute $39 + (-25)$.

Solution

Step 1: Change from decimal to 8-bit representations using the two’s complement to represent -25 .

As in Example 2.5.8, the 8-bit representation of 39 is 00100111. Now the 8-bit representation of -25 is the two’s complement of 25, which is obtained as follows:

$$\begin{array}{lcl}
 25_{10} = (16 + 8 + 1)_{10} = 00011001_2 & \xrightarrow{\text{flip the bits}} & \\
 & & 11100110 \\
 & & \xrightarrow{\text{add 1}} \quad 11100111
 \end{array}$$

So the 8-bit representation of -25 is 11100111.

Step 2: Add the 8-bit representations in binary notation and truncate the 1 in the 2^8 th position if there is one:

$$\begin{array}{r}
 \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \\
 + \\
 \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \\
 \hline
 \text{Truncate} \rightarrow 1 \boxed{0} \boxed{0} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{0}
 \end{array}$$

Step 3: Find the decimal equivalent of the result:

$$00001110_2 = (8 + 4 + 2)_{10} = 14_{10}.$$

Since $39 - 25 = 14$, this is the correct answer. ■

Case 4 (both integers are negative): This case involves adding two negative integers in the range -1 through -128 whose sum is also in this range. To be specific, consider the sum $(-a) + (-b)$ where a , b , and $a + b$ are all in the range 1 through 128. In this case, the 8-bit representations of $-a$ and $-b$ are the 8-bit representations of $2^8 - a$ and $2^8 - b$. So if the 8-bit representations of $-a$ and $-b$ are added, the result is

$$(2^8 - a) + (2^8 - b) = [2^8 - (a + b)] + 2^8.$$

Recall that truncating a leading 1 in the ninth (2^8 th) position of a binary number is equivalent to subtracting 2^8 . So when the leading 1 is truncated from the 8-bit representation of $(2^8 - a) + (2^8 - b)$, the result is $2^8 - (a + b)$, which is the 8-bit representation of $-(a + b) = (-a) + (-b)$. (In exercise 37 you are asked to show that the sum $(2^8 - a) + (2^8 - b)$ has a leading 1 in the ninth (2^8 th) position.)

Example 2.5.10 Computing $(-a) + (-b)$ Where $1 \leq a, b \leq 128$, and $1 \leq a + b \leq 128$

Use 8-bit representations to compute $(-89) + (-25)$.

Solution

Step 1: Change from decimal to 8-bit representations using the two's complements to represent -89 and -25 .

The 8-bit representations of -89 and -25 were shown in Examples 2.5.8 and 2.5.9 to be 10100111 and 11100111, respectively.

Step 2: Add the 8-bit representations in binary notation and truncate the 1 in the 2^8 th position if there is one:

$$\begin{array}{r}
 \boxed{1} \boxed{0} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \\
 + \\
 \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \\
 \hline
 \text{Truncate} \rightarrow 1 \boxed{1} \boxed{0} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{0}
 \end{array}$$

Step 3: Find the decimal equivalent of the result. Because its leading bit is 1, this number is the 8-bit representation of a negative integer.

$$\begin{array}{lcl}
 10001110 & \xrightarrow{\text{flip the bits}} & 01110001 \xrightarrow{\text{add 1}} 01110010_2 \\
 & & \leftrightarrow -(64 + 32 + 16 + 2)_{10} = -114_{10}
 \end{array}$$

Since $(-89) + (-25) = -114$, that is the correct answer. ■

Hexadecimal Notation

It should now be obvious that numbers written in binary notation take up much more space than numbers written in decimal notation. Yet many aspects of computer operation can best be analyzed using binary numbers. **Hexadecimal notation** is even more compact than decimal notation, and it is much easier to convert back and forth between hexadecimal and binary notation than it is between binary and decimal notation. The word *hexadecimal* comes from the Greek root *hex-*, meaning “six,” and the Latin root *deci-*, meaning “ten.” Hence *hexadecimal* refers to “sixteen,” and hexadecimal notation is also called **base 16 notation**. Hexadecimal notation is based on the fact that any integer can be uniquely expressed as a sum of numbers of the form

$$d \cdot 16^n,$$

where each n is a nonnegative integer and each d is one of the integers from 0 to 15. In order to avoid ambiguity, each hexadecimal digit must be represented by a single symbol. The integers 10 through 15 are represented by the symbols A, B, C, D, E, and F. The sixteen hexadecimal digits are shown in Table 2.5.3, together with their decimal equivalents and, for future reference, their 4-bit binary equivalents.

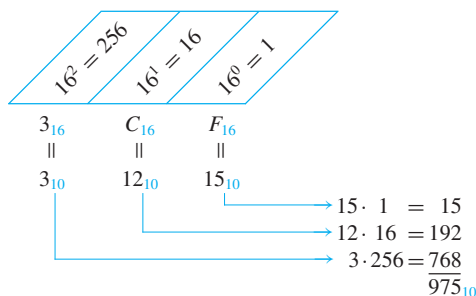
Table 2.5.3

Decimal	Hexadecimal	4-Bit Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Example 2.5.11 Converting from Hexadecimal to Decimal Notation

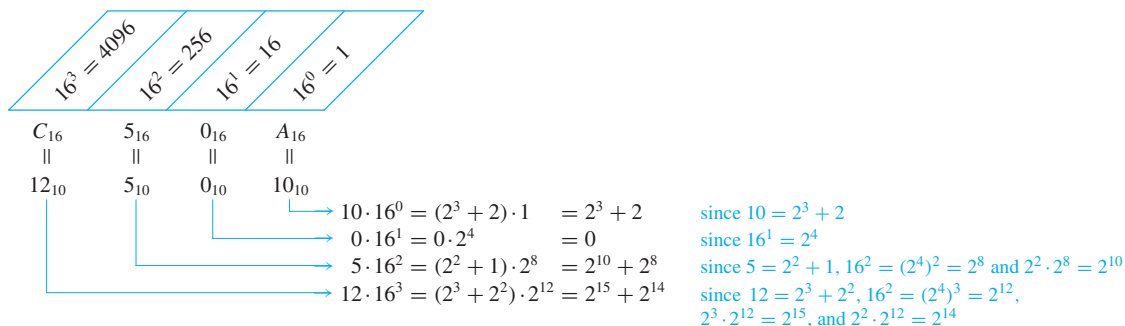
Convert $3CF_{16}$ to decimal notation.

Solution A schema similar to the one introduced in Example 2.5.2 can be used here.



So $3CF_{16} = 975_{10}$.

Now consider how to convert from hexadecimal to binary notation. In the example below the numbers are rewritten using powers of 2, and the laws of exponents are applied. The result suggests a general procedure.



But

$$\begin{aligned}
 &(2^{15} + 2^{14}) + (2^{10} + 2^8) + 0 + (2^3 + 2) \\
 &= 1100\ 0000\ 0000\ 0000_2 + 0101\ 0000\ 0000_2 \quad \text{by the rules for writing binary numbers.} \\
 &\quad + 0000\ 0000_2 + 1010_2
 \end{aligned}$$

So

$$C50A_{16} = \underbrace{1100}_{C_{16}} \underbrace{0101}_{5_{16}} \underbrace{0000}_{0_{16}} \underbrace{1010}_{A_{16}}_2 \quad \text{by the rules for adding binary numbers.}$$

The procedure illustrated in this example can be generalized. In fact, the following sequence of steps will always give the correct answer:

To convert an integer from hexadecimal to binary notation:

- Write each hexadecimal digit of the integer in 4-bit binary notation.
- Juxtapose the results.

Example 2.5.12 Converting from Hexadecimal to Binary Notation

Convert $B09F_{16}$ to binary notation.

Solution $B_{16} = 11_{10} = 1011_2$, $0_{16} = 0_{10} = 0000_2$, $9_{16} = 9_{10} = 1001_2$, and $F_{16} = 15_{10} = 1111_2$. Consequently,

B	0	9	F
\updownarrow	\updownarrow	\updownarrow	\updownarrow
1011	0000	1001	1111

and the answer is 1011000010011111_2 . ■

To convert integers written in binary notation into hexadecimal notation, reverse the steps of the previous procedure.

To convert an integer from binary to hexadecimal notation:

- Group the digits of the binary number into sets of four, starting from the right and adding leading zeros as needed.
- Convert the binary numbers in each set of four into hexadecimal digits. Juxtapose those hexadecimal digits.

Example 2.5.13 Converting from Binary to Hexadecimal Notation

Convert 100110110101001_2 to hexadecimal notation.

Solution First group the binary digits in sets of four, working from right to left and adding leading 0's if necessary.

$$0100 \quad 1101 \quad 1010 \quad 1001.$$

Convert each group of four binary digits into a hexadecimal digit.

0100	1101	1010	1001
\updownarrow	\updownarrow	\updownarrow	\updownarrow
4	D	A	9

Then juxtapose the hexadecimal digits.

$$4DA9_{16}$$

■

Example 2.5.14 Reading a Memory Dump

The smallest addressable memory unit on most computers is one byte, or eight bits. In some debugging operations a dump is made of memory contents; that is, the contents of each memory location are displayed or printed out in order. To save space and make the output easier on the eye, the hexadecimal versions of the memory contents are given, rather than the binary versions. Suppose, for example, that a segment of the memory dump looks like

$$A3 \text{ BB } 59 \text{ 2E}.$$

What is the actual content of the four memory locations?

Solution

$$A3_{16} = 10100011_2$$

$$BB_{16} = 10111011_2$$

$$59_{16} = 01011001_2$$

$$2E_{16} = 00101110_2$$

Test Yourself

- To represent a nonnegative integer in binary notation means to write it as a sum of products of the form $______$, where $______$.
- To add integers in binary notation, you use the facts that $1_2 + 1_2 = ______$ and $1_2 + 1_2 + 1_2 = ______$.
- To subtract integers in binary notation, you use the facts that $10_2 - 1_2 = ______$ and $11_2 - 1_2 = ______$.
- A half-adder is a digital logic circuit that $______$, and a full-adder is a digital logic circuit that $______$.
- The 8-bit two's complement of a positive integer a is $______$.
- To find the 8-bit two's complement of a positive integer a that is at most 255, you $______$, $______$, and $______$.
- If a is an integer with $-128 \leq a \leq 127$, the 8-bit representation of a is $______$ if $a \geq 0$ and is $______$ if $a < 0$.
- To add two integers in the range -128 through 127 whose sum is also in the range -128 through 127 , you $______$, $______$, and $______$.
- To represent a nonnegative integer in hexadecimal notation means to write it as a sum of products of the form $______$, where $______$.
- To convert a nonnegative integer from hexadecimal to binary notation, you $______$ and $______$.

Exercise Set 2.5

Represent the decimal integers in 1–6 in binary notation.

- 19
- 55
- 287
- 458
- 1609
- 1424

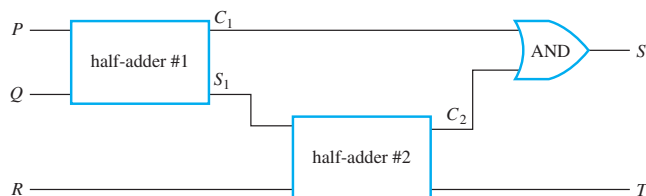
Represent the integers in 7–12 in decimal notation.

- 1110_2
- 10111_2
- 110110_2
- 1100101_2
- 1000111_2
- 1011011_2

Perform the arithmetic in 13–20 using binary notation.

- $$\begin{array}{r} 1011_2 \\ + 101_2 \\ \hline \end{array}$$
- $$\begin{array}{r} 101101_2 \\ + 11101_2 \\ \hline \end{array}$$
- $$\begin{array}{r} 10100_2 \\ - 1101_2 \\ \hline \end{array}$$
- $$\begin{array}{r} 101101_2 \\ - 10011_2 \\ \hline \end{array}$$
- $$\begin{array}{r} 1001_2 \\ + 1011_2 \\ \hline \end{array}$$
- $$\begin{array}{r} 110111011_2 \\ + 1001011010_2 \\ \hline \end{array}$$
- $$\begin{array}{r} 11010_2 \\ - 1101_2 \\ \hline \end{array}$$
- $$\begin{array}{r} 1010100_2 \\ - 10111_2 \\ \hline \end{array}$$

- Give the output signals S and T for the circuit in the right column if the input signals P , Q , and R are as specified. Note that this is *not* the circuit for a full-adder.
 - $P = 1$, $Q = 1$, $R = 1$
 - $P = 0$, $Q = 1$, $R = 0$
 - $P = 1$, $Q = 0$, $R = 1$



- Add $11111111_2 + 1_2$ and convert the result to decimal notation, to verify that $11111111_2 = (2^8 - 1)_{10}$.

Find the 8-bit two's complements for the integers in 23–26.

- 23
- 67
- 4
- 115

Find the decimal representations for the integers with the 8-bit representations given in 27–30.

- 11010011
- 10011001
- 11110010
- 10111010

Use 8-bit representations to compute the sums in 31–36.

- $57 + (-118)$
- $62 + (-18)$
- $(-6) + (-73)$
- $89 + (-55)$
- $(-15) + (-46)$
- $123 + (-94)$

- ★ 37. Show that if a , b , and $a + b$ are integers in the range 1 through 128, then

$$(2^8 - a) + (2^8 - b) = (2^8 - (a + b)) + 2^8 \geq 2^8 + 2^7.$$

Explain why it follows that if the 8-bit binary representation of the sum of the negatives of two numbers in the given range is computed, the result is a negative number.

Convert the integers in 38–40 from hexadecimal to decimal notation.

38. $A2BC_{16}$ 39. $E0D_{16}$ 40. $39EB_{16}$

Convert the integers in 41–43 from hexadecimal to binary notation.

41. $1C0ABE_{16}$ 42. $B53DF8_{16}$ 43. $4ADF83_{16}$

Convert the integers in 44–46 from binary to hexadecimal notation.

44. 00101110_2 45. 1011011111000101_2

46. 11001001011100_2

47. **Octal Notation:** In addition to binary and hexadecimal, computer scientists also use *octal notation* (base 8) to represent numbers. Octal notation is based on the fact that any integer can be uniquely represented as a sum of numbers of the form $d \cdot 8^n$, where each n is a nonnegative integer and each d is one of the integers from 0 to 7. Thus, for example, $5073_8 = 5 \cdot 8^3 + 0 \cdot 8^2 + 7 \cdot 8^1 + 3 \cdot 8^0 = 2619_{10}$.

- Convert 61502_8 to decimal notation.
- Convert 20763_8 to decimal notation.
- Describe methods for converting integers from octal to binary notation and the reverse that are similar to the methods used in Examples 2.5.12 and 2.5.13 for converting back and forth from hexadecimal to binary notation. Give examples showing that these methods result in correct answers.

Answers for Test Yourself

- $d \cdot 2^n$; $d = 0$ or $d = 1$, and n is a nonnegative integer
- $10_2; 11_2$
- $1_2; 10_2$
- outputs the sum of any two binary digits; outputs the sum of any three binary digits
- $2^8 - a$
- write the 8-bit binary representation of a ; flip the bits; add 1 in binary notation
- the 8-bit binary representation of a ; the 8-bit binary representation of $2^8 - a$
- convert both integers to their 8-bit binary representations; add the results using binary notation; truncate any leading 1; convert back to decimal form
- $d \cdot 16^n$; $d = 0, 1, 2, \dots, 9, A, B, C, D, E, F$, and n is a nonnegative integer
- write each hexadecimal digit in 4-bit binary notation; juxtapose the results