CHAPTER 1

SPEAKING MATHEMATICALLY

Copyright © Cengage Learning. All rights reserved.

1.1 Variables

Copyright © Cengage Learning. All rights reserved.

Variables (1/4)

A variable is sometimes thought of as a mathematical "John Doe" because you can use it as a placeholder when you want to talk about something but either (1) you imagine that it has one or more values but you don't know what they are, or (2) you want whatever you say about it to be equally true for all elements in a given set, and so you don't want to be restricted to considering only a particular, concrete value for it. To illustrate the first use, consider asking

> Is there a number with the following property: doubling it and adding 3 gives the same result as squaring it?

Variables (2/4)

In this sentence you can introduce a variable to replace the potentially ambiguous word “it”:

Is there a number *x* with the property that $2x + 3 = x^2$?

The advantage of using a variable is that it allows you to give a temporary name to what you are seeking so that you can perform concrete computations with it to help discover its possible values.

Variables (3/4)

To emphasize the role of the variable as a placeholder, you might write the following:

> Is there a number $\square$ with the property that $2 \cdot \square + 3 = \square^2$?

The emptiness of the box can help you imagine filling it in with a variety of different values, some of which might make the two sides equal and others of which might not.

In this sense, a variable in a computer program is similar to a mathematical variable because it creates a location in computer memory (either actual or virtual) into which values can be placed.

Variables (4/4)

To illustrate the second use of variables, consider the statement

> No matter what number might be chosen, if it is greater than 2, then its square is greater than 4.

In this case introducing a variable to give a temporary name to the (arbitrary) number you might choose enables you to maintain the generality of the statement, and replacing all instances of the word “it” by the name of the variable ensures that possible ambiguity is avoided

> No matter what number *n* might be chosen, if *n* is greater than 2, then n^2 is greater than 4.

Example 1.1.1 – *Writing Sentences Using Variables*

Use variables to rewrite the following sentences more formally.

a. Are there numbers with the property that the sum of their squares equals the square of their sum?

b. Given any real number, its square is nonnegative.

Example 1.1.1 – *Solution*

a. Are there numbers *a* and *b* with the property that $a^2 + b^2 = (a+b)^2$?

Or: Are there numbers *a* and *b* such that $a^2 + b^2 = (a+b)^2$?

Or: Do there exist any numbers *a* and *b* such that $a^2 + b^2 = (a+b)^2$?

b. Given any real number r, r^2 is nonnegative.

Or: For any real number r, $r^2 \geq 0$.

Or: For every real number r, $r^2 \geq 0$.

Some Important Kinds of Mathematical Statements

Some Important Kinds of Mathematical Statements (1/1)

Three of the most important kinds of sentences in mathematics are universal statements, conditional statements, and existential statements:

A **universal statement** says that a certain property is true for all elements in a set. (For example: *All positive numbers are greater than zero.*)

A **conditional statement** says that if one thing is true then some other thing also has to be true. (For example: *If 378 is divisible by 18, then 378 is divisible by 6.*)

Given a property that may or may not be true, an **existential statement** says that there is at least one thing for which the property is true. (For example: *There is a prime number that is even.*)

Universal Conditional Statements

Universal Conditional Statements (1/1)

Universal statements contain some variation of the words "for every" and conditional statements contain versions of the words "if-then." A **universal conditional statement** is a statement that is both universal and conditional. Here is an example:

> For every animal *a*, if *a* is a dog, then *a* is a mammal.

One of the most important facts about universal conditional statements is that they can be rewritten in ways that make them appear to be purely universal or purely conditional.

Universal Conditional Statements (2/2)

For example, the previous statement can be rewritten in a way that makes its conditional nature explicit but its universal nature implicit:

> If *a* is a dog, then *a* is a mammal.
>
> *Or:* If an animal is a dog, then the animal is a mammal.

The statement can also be expressed so as to make its universal nature explicit and its conditional nature implicit:

> For every dog *a*, *a* is a mammal.
>
> *Or:* All dogs are mammals.

Example 1.1.2 – *Rewriting a Universal Conditional Statement*

Fill in the blanks to rewrite the following statement:

For every real number *x*, if *x* is nonzero then x^2 is positive.

a. If a real number is nonzero, then its square _____.

b. For every nonzero real number *x*, _____.

c. If *x* _____, then _____.

d. The square of any nonzero real number is _____.

e. All nonzero real numbers have _____.

Example 1.1.2 – *Solution*

a. is positive

b. x^2 is positive

c. is a nonzero real number; x^2 is positive

d. is positive

e. positive squares (*or:* squares that are positive)

Universal Existential Statements

Universal Existential Statements (1/3)

A **universal existential statement** is a statement that is universal because its first part says that a certain property is true for all objects of a given type, and it is existential because its second part asserts the existence of something. For example:

Every real number has an additive inverse.

In this statement the property “has an additive inverse” applies universally to all real numbers. “Has an additive inverse” asserts the existence of something—an additive inverse—for each real number.

Universal Existential Statements (2/3)

However, the nature of the additive inverse depends on the real number; different real numbers have different additive inverses. Knowing that an additive inverse is a real number, you can rewrite this statement in several ways, some less formal and some more formal:

All real numbers have additive inverses.

Or: For every real number r, there is an additive inverse for r.

Or: For every real number r, there is a real number s such that s is an additive inverse for r.

Universal Existential Statements (3/3)

One of the most important reasons for using variables in mathematics is that it gives you the ability to refer to quantities unambiguously throughout a lengthy mathematical argument, while not restricting you to consider only specific values for them.

Example 1.1.3 – *Rewriting a Universal Existential Statement*

Fill in the blanks to rewrite the following statement: Every pot has a lid.

a. All pots _____.

b. For every pot *P*, there is _____.

c. For every pot *P*, there is a lid *L* such that _____.

Example 1.1.3 – *Solution*

a. have lids

b. a lid for P

c. L is a lid for P

Existential Universal Statements

Existential Universal Statements (1/4)

An **existential universal statement** is a statement that is existential because its first part asserts that a certain object exists and is universal because its second part says that the object satisfies a certain property for all things of a certain kind. For example:

> There is a positive integer that is less than or equal to every positive integer.

This statement is true because the number one is a positive integer, and it satisfies the property of being less than or equal to every positive integer.

Existential Universal Statements (2/4)

We can rewrite the statement in several ways, some less formal and some more formal:

Some positive integer is less than or equal to every positive integer.

Or: There is a positive integer m that is less than or equal to every positive integer.

Or: There is a positive integer m such that every positive integer is greater than or equal to m.

Or: There is a positive integer m with the property that for every positive integer n, $m \leq n$.

Example 1.1.4 – *Rewriting an Existential Universal Statement*

Fill in the blanks to rewrite the following statement in three different ways:

> There is a person in my class who is at least as old as every person in my class.

a. Some _____ is at least as old as _____.

b. There is a person *p* in my class such that *p* is _____.

c. There is a person *p* in my class with the property that for every person *q* in my class, *p* is _____.

Example 1.1.4 – *Solution*

a. person in my class; every person in my class

b. at least as old as every person in my class

c. at least as old as *q*

Existential Universal Statements (3/4)

Some of the most important mathematical concepts, such as the definition of limit of a sequence, can only be defined using phrases that are universal, existential, and conditional, and they require the use of all three phrases “for every,” “there is,” and “if-then.”

Existential Universal Statements (4/4)

For example, if a_1, a_2, a_3,… is a sequence of real numbers, saying that

the limit of a_n as n approaches infinity is L

means that

for every positive real number ε, **there is** an integer N such that **for every** integer n, **if** $n > N$ **then** $-\varepsilon < a_n - L < \varepsilon.$

CHAPTER 1

SPEAKING MATHEMATICALLY

Copyright © Cengage Learning. All rights reserved.

1.2 The Language of Sets

Copyright © Cengage Learning. All rights reserved.

The Language of Sets (1/7)

Use of the word *set* as a formal mathematical term was introduced in 1879 by Georg Cantor (1845–1918). For most mathematical purposes we can think of a set intuitively, as Cantor did, simply as a collection of elements.

Set-Roster Notation

If S is a set, the notation $\boldsymbol{x \in S}$ means that x is an element of S. The notation $\boldsymbol{x \notin S}$ means that x is not an element of S. A set may be specified using the **set-roster notation** by writing all of its elements between braces. For example, $\{\mathbf{1, 2, 3}\}$ denotes the set whose elements are 1, 2, and 3. A variation of the notation is sometimes used to describe a very large set, as when we write $\{\mathbf{1, 2, 3, \ldots, 100}\}$ to refer to the set of all integers from 1 to 100. A similar notation can also describe an infinite set, as when we write $\{\mathbf{1, 2, 3, \ldots}\}$ to refer to the set of all positive integers. (The symbol $\ldots$ is called an **ellipsis** and is read "and so forth.")

The Language of Sets (2/7)

The **axiom of extension** says that a set is completely determined by what its elements are—not the order in which they might be listed or the fact that some elements might be listed more than once.

Example 1.2.1 – *Using the Set-Roster Notation*

a. Let $A = \{1, 2, 3\}$, $B = \{3, 1, 2\}$, and $C = \{1, 1, 2, 3, 3, 3\}$. What are the elements of A, B, and C? How are A, B, and C related?

b. Is $\{0\} = 0$?

c. How many elements are in the set $\{1, \{1\}\}$?

d. For each nonnegative integer n, let $\cup_n = \{n, -n\}$. Find $\cup_1$, $\cup_2$, and $\cup_0$.

Example 1.2.1 – *Solution (1/2)*

a. A, B, and C have exactly the same three elements: 1, 2, and 3. Therefore, A, B, and C are simply different ways to represent the same set.

b. $\{0\} \neq 0$ because $\{0\}$ is a set with one element, namely 0, whereas 0 is just the symbol that represents the number zero.

Example 1.2.1 – *Solution (2/2)* continued

c. The set {1, {1}} has two elements: 1 and the set whose only element is 1.

d. $\cup_1 = \{1, -1\}$, $\cup_2 = \{2, -2\}$, $\cup_0 = \{0, -0\} = \{0, 0\} = \{0\}$.

The Language of Sets (3/7)

Certain sets of numbers are so frequently referred to that they are given special symbolic names. These are summarized in the following table.

Symbol	Set
R	the set of all real numbers
Z	the set of all integers
Q	the set of all rational numbers, or quotients of integers

The Language of Sets (4/7)

Addition of a superscript + or − or the letters *nonneg* indicates that only the positive or negative or nonnegative elements of the set, respectively, are to be included. Thus $\mathbf{R}^{+}$ denotes the set of positive real numbers, and $\mathbf{Z}^{nonneg}$ refers to the set of nonnegative integers: 0, 1, 2, 3, 4, and so forth.

Some authors refer to the set of nonnegative integers as the set of **natural numbers** and denote it as **N**.

The Language of Sets (5/7)

The set of real numbers is usually pictured as the set of all points on a line. The number 0 corresponds to a middle point, called the *origin*.

A unit of distance is marked off, and each point to the right of the origin corresponds to a positive real number found by computing its distance from the origin.

The Language of Sets (6/7)

Each point to the left of the origin corresponds to a negative real number, which is denoted by computing its distance from the origin and putting a minus sign in front of the resulting number. The set of real numbers is therefore divided into three parts: the set of positive real numbers, the set of negative real numbers, and the number 0. *Note that 0 is neither positive nor negative*. Labels are given for a few real numbers corresponding to points on the line shown below.

The Language of Sets (7/7)

Set-Builder Notation

Let S denote a set and let $P(x)$ be a property that elements of S may or may not satisfy. We may define a new set to be **the set of all elements x in S such that $P(x)$ is true**. We denote this set as follows:

$$\{x \in S \mid P(x)\}$$

the set of all such that

Example 1.2.2 – *Using the Set-Builder Notation*

Given that **R** denotes the set of all real numbers, **Z** the set of all integers, and $\mathbf{Z}^+$ the set of all positive integers, describe each of the following sets.

a. $\{x \in \mathbf{R} \mid -2 < x < 5\}$

b. $\{x \in \mathbf{Z} \mid -2 < x < 5\}$

c. $\{x \in \mathbf{Z}^+ \mid -2 < x < 5\}$

Example 1.2.2 – *Solution*

a. $\{x \in \mathbf{R} \mid -2 < x < 5\}$ is the open interval of real numbers (strictly) between −2 and 5. It is pictured as follows:

b. $\{x \in \mathbf{Z} \mid -2 < x < 5\}$ is the set of all integers (strictly) between −2 and 5. It is equal to the set {−1, 0, 1, 2, 3, 4}.

c. Since all the integers in $\mathbf{Z}^+$ are positive,
$\{x \in \mathbf{Z}^+ \mid -2 < x < 5\} = \{1, 2, 3, 4\}$.

Subsets

Subsets (1/2)

A basic relation between sets is that of subset.

Definition

If A and B are sets, then A is called a **subset** of B, written $\boldsymbol{A \subseteq B}$, if, and only if, every element of A is also an element of B.

Symbolically:

$$A \subseteq B \quad \text{means that} \quad \text{for every element } x, \text{ if } x \in A \text{ then } x \in B.$$

The phrases A *is contained in* B and B *contains* A are alternative ways of saying that A is a subset of B.

Subsets (2/2)

It follows from the definition of subset that for a set *A* not to be a subset of a set *B* means that there is at least one element of *A* that is not an element of *B*. Symbolically:

$A \not\subseteq B$ means that there is at least one element x such that $x \in A$ and $x \notin B$.

Definition

Let A and B be sets. A is a **proper subset** of B if, and only if, every element of A is in B but there is at least one element of B that is not in A.

Example 1.2.3 – *Subsets*

Let $A = \mathbf{Z}^+$, $B = \{n \in \mathbf{Z} \mid 0 \leq n \leq 100\}$, and C = {100, 200, 300, 400, 500}. Evaluate the truth and falsity of each of the following statements.

a. $B \subseteq A$

b. C is a proper subset of A

c. C and B have at least one element in common

d. $C \subseteq B$

e. $C \subseteq C$

Example 1.2.3 – *Solution (1/2)*

a. False. Zero is not a positive integer. Thus zero is in B but zero is not in A, and so $B \not\subseteq A$.

b. True. Each element in C is a positive integer and, hence, is in A, but there are elements in A that are not in C. For instance, 1 is in A and not in C.

c. True. For example, 100 is in both C and B.

Example 1.2.3 – *Solution (2/2)*

continued

d. False. For example, 200 is in *C* but not in *B*.

e. True. Every element in *C* is in *C*. In general, the definition of subset implies that all sets are subsets of themselves.

Example 1.2.4 – *Distinction between* $\in$ *and* $\subseteq$

Which of the following are true statements?

a. $2 \in \{1, 2, 3\}$

b. $\{2\} \in \{1, 2, 3\}$

c. $2 \subseteq \{1, 2, 3\}$

d. $\{2\} \subseteq \{1, 2, 3\}$

e. $\{2\} \subseteq \{\{1\}, \{2\}\}$

f. $\{2\} \in \{\{1\}, \{2\}\}$

Example 1.2.4 – *Solution (1/2)*

Only (a), (d), and (f) are true.

For (b) to be true, the set {1, 2, 3} would have to contain the element {2}. But the only elements of {1, 2, 3} are 1, 2, and 3, and 2 is not equal to {2}. Hence (b) is false.

For (c) to be true, the number 2 would have to be a set and every element in the set 2 would have to be an element of {1, 2, 3}. This is not the case, so (c) is false.

Example 1.2.4 – *Solution (2/2)*

continued

For (e) to be true, every element in the set containing only the number 2 would have to be an element of the set whose elements are {1} and {2}. But 2 is not equal to either {1} or {2}, and so (e) is false.

Cartesian Products

Cartesian Products (1/4)

Notation

Given elements a and b, the symbol (a, b) denotes the **ordered pair** consisting of a and b together with the specification that a is the first element of the pair and b is the second element. Two ordered pairs (a, b) and (c, d) are equal if, and only if, $a = c$ and $b = d$. Symbolically:

$$(a, b) = (c, d) \quad \text{means that} \quad a = c \text{ and } b = d.$$

Example 1.2.5 – *Ordered Pairs*

a. Is (1, 2) = (2, 1)?

b. Is $\left(3, \frac{5}{10}\right) = \left(\sqrt{9}, \frac{1}{2}\right)$?

c. What is the first element of (1, 1)?

Example 1.2.5 – *Solution (1/2)*

a. No. By definition of equality of ordered pairs,

$$(1, 2) = (2, 1) \text{ if, and only if, } 1 = 2 \text{ and } 2 = 1.$$

But $1 \neq 2$, and so the ordered pairs are not equal.

b. Yes. By definition of equality of ordered pairs,

$$\left(3, \frac{5}{10}\right) = \left(\sqrt{9}, \frac{1}{2}\right) \text{ if, and only if, } 3 = \sqrt{9} \text{ and } \frac{5}{10} = \frac{1}{2}.$$

Because these equations are both true, the ordered pairs are equal.

Example 1.2.5 – *Solution (2/2)*

continued

c. In the ordered pair (1, 1), the first and the second elements are both 1.

Cartesian Products (2/4)

The notation for an *ordered n-tuple* generalizes the notation for an ordered pair to a set with any finite number of elements. It also takes both order and multiplicity into account.

Definition

Let n be a positive integer and let $x_1, x_2, \ldots, x_n$ be (not necessarily distinct) elements. The **ordered n-tuple, $(x_1, x_2, \ldots, x_n)$**, consists of $x_1, x_2, \ldots, x_n$ together with the ordering: first x_1, then x_2, and so forth up to x_n. An ordered 2-tuple is called an **ordered pair**, and an ordered 3-tuple is called an **ordered triple**.

Two ordered n-tuples $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ are **equal** if, and only if, $x_1 = y_1, x_2 = y_2, \ldots,$ and $x_n = y_n$.

Symbolically:

$$(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n) \quad \Leftrightarrow \quad x_1 = y_1, x_2 = y_2, \ldots, x_n = y_n.$$

Example 1.2.6 – *Ordered n-tuples*

a. Is (1, 2, 3, 4) = (1, 2, 4, 3)?

b. Is $\left(3, (-2)^2, \frac{1}{2}\right) = \left(\sqrt{9}, 4, \frac{3}{6}\right)$?

Example 1.2.6 – *Solution (1/2)*

a. No. By definition of equality of ordered 4-tuples,

$$(1, 2, 3, 4) = (1, 2, 4, 3) \Leftrightarrow 1 = 1, 2 = 2, 3 = 4, \text{ and } 4 = 3$$

But $3 \neq 4$, and so the ordered 4-tuples are not equal.

Example 1.2.6 – *Solution (2/2)*

continued

b. Yes. By definition of equality of ordered triples,

$$\left(3, (-2)^2, \frac{1}{2}\right) = \left(\sqrt{9}, 4, \frac{3}{6}\right) \quad \Leftrightarrow \quad 3 = \sqrt{9} \text{ and } (-2)^2 = 4 \text{ and } \frac{1}{2} = \frac{3}{6}.$$

Because these equations are all true, the two ordered triples are equal.

Cartesian Products (3/4)

Definition

Given sets $A_1, A_2, \ldots, A_n$, the **Cartesian product** of $A_1, A_2, \ldots, A_n$, denoted $\boldsymbol{A_1 \times A_2 \times \cdots \times A_n}$, is the set of all ordered n-tuples $(a_1, a_2, \ldots, a_n)$ where $a_1 \in A_1$, $a_2 \in A_2, \ldots, a_n \in A_n$.

Symbolically:

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n\}.$$

In particular,

$$A_1 \times A_2 = \{(a_1, a_2) \mid a_1 \in A_1 \text{ and } a_2 \in A_2\}$$

is the Cartesian product of A_1 and A_2.

Example 1.2.7 – *Cartesian Products*

Let $A = \{x, y\}$, $B = \{1, 2, 3\}$, and $C = \{a, b\}$.

a. Find $A \times B$.

b. Find $B \times A$.

c. Find $A \times A$.

d. How many elements are in $A \times B$, $B \times A$, and $A \times A$?

e. Find $(A \times B) \times C$

f. Find $A \times B \times C$

g. Let $\mathbf{R}$ denote the set of all real numbers. Describe $\mathbf{R} \times \mathbf{R}$.

Example 1.2.7 – *Solution (1/5)*

a. $A \times B = \{(x, 1), (y, 1), (x, 2), (y, 2), (x, 3), (y, 3)\}$

b. $B \times A = \{(1, x), (1, y), (2, x), (2, y), (3, x), (3, y)\}$

c. $A \times A = \{(x, x), (x, y), (y, x), (y, y)\}$

d. $A \times B$ has 6 elements. Note that this is the number of elements in *A* times the number of elements in *B*. $B \times A$ has 6 elements, the number of elements in *B* times the number of elements in *A*. $A \times A$ has 4 elements, the number of elements in *A* times the number of elements in *A*.

Example 1.2.7 – *Solution (2/5)* continued

e. The Cartesian product of *A* and *B* is a set, so it may be used as one of the sets making up another Cartesian product. This is the case for $(A \times B) \times C$.

$$(A \times B) \times C = \{(u, v) | u \in A \times B \text{ and } v \in C\}$$ by definition of Cartesian product

$$= \{((x, 1), a), ((x, 2), a), ((x, 3), a), ((y, 1), a), ((y, 2), a), ((y, 3), a), ((x, 1), b), ((x, 2), b), ((x, 3), b), ((y, 1), b), ((y, 2), b), ((y, 3), b)\}$$

Example 1.2.7 – *Solution (3/5)*

continued

f. The Cartesian product $A \times B \times C$ is superficially similar to but is not quite the same mathematical object as $(A \times B) \times C$. $(A \times B) \times C$ is a set of ordered pairs of which one element is itself an ordered pair, whereas $A \times B \times C$ is a set of ordered triples. By definition of Cartesian product,

$$\begin{aligned} A \times B \times C &= \{(u, v, w) \mid u \in A, v \in B, \text{ and } w \in C\} \\ &= \{(x, 1, a), (x, 2, a), (x, 3, a), (y, 1, a), (y, 2, a), (y, 3, a), (x, 1, b), \\ &\quad (x, 2, b), (x, 3, b), (y, 1, b), (y, 2, b), (y, 3, b)\}. \end{aligned}$$

Example 1.2.7 – *Solution (4/5)*

continued

g. $\mathbf{R} \times \mathbf{R}$ is the set of all ordered pairs (x, y) where both x and y are real numbers. If horizontal and vertical axes are drawn on a plane and a unit length is marked off, then each ordered pair in $\mathbf{R} \times \mathbf{R}$ corresponds to a unique point in the plane, with the first and second elements of the pair indicating, respectively, the horizontal and vertical positions of the point.

Example 1.2.7 – *Solution (5/5)*

continued

The term **Cartesian plane** is often used to refer to a plane with this coordinate system, as illustrated in Figure 1.2.1.

A Cartesian Plane

Figure 1.2.1

Cartesian Products (4/4)

Another notation, which is important in both mathematics and computer science, denotes objects called *strings*.

Definition

Let n be a positive integer. Given a finite set A, a **string of length n over A** is an ordered n-tuple of elements of A written without parentheses or commas. The elements of A are called the **characters** of the string. The **null string** over A is defined to be the "string" with no characters. It is often denoted λ and is said to have length 0. If $A = \{0, 1\}$, then a string over A is called a **bit string**.

Example 1.2.8 – *Strings*

Let $A = \{a, b\}$. List all the strings of length 3 over A with at least two characters that are the same.

Example 1.2.8 – *Solution*

aab, *aba*, *baa*, *aaa*, *bba*, *bab*, *abb*, *bbb*

In computer programming it is important to distinguish among different kinds of data structures and to respect the notations that are used for them. Similarly in mathematics, it is important to distinguish among, say, $\{a, b, c\}$, $\{\{a, b\}, c\}$, (a, b, c), $(a, (b, c))$, *abc*, and so forth, because these are all significantly different objects.

CHAPTER 1

SPEAKING MATHEMATICALLY

Copyright © Cengage Learning. All rights reserved.

1.3 The Language of Relations and Functions

Copyright © Cengage Learning. All rights reserved.

The Language of Relations and Functions

Definition

Let A and B be sets. A **relation R from A to B** is a subset of $A \times B$. Given an ordered pair (x, y) in $A \times B$, **x is related to y by R**, written $x\,R\,y$, if, and only if, (x, y) is in R. The set A is called the **domain** of R and the set B is called its **co-domain**.

The notation for a relation R may be written symbolically as follows:

$$\boldsymbol{x\,R\,y} \quad \text{means that} \quad \boldsymbol{(x, y) \in R.}$$

The notation $x \not R\, y$ means that x is not related to y by R:

$$\boldsymbol{x \not R\, y} \quad \text{means that} \quad \boldsymbol{(x, y) \notin R.}$$

Example 1.3.1 – *A Relation as a Subset*

Let $A = \{1, 2\}$ and $B = \{1, 2, 3\}$ and define a relation R from A to B as follows: Given any $(x, y) \in A \times B$,

$$(x, y) \in R \quad \text{means that} \quad \frac{x - y}{2} \text{ is an integer.}$$

a. State explicitly which ordered pairs are in $A \times B$ and which are in R.

b. Is $1 \; R \; 3$? Is $2 \; R \; 3$? Is $2 \; R \; 2$?

c. What are the domain and co-domain of R?

Example 1.3.1 – *Solution (1/2)*

a. $A \times B = \{(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3)\}$. To determine explicitly the composition of R, examine each ordered pair in $A \times B$ to see whether its elements satisfy the defining condition for R.

$(1, 1) \in R$ because $\frac{1-1}{2} = \frac{0}{2} = 0,$ which is an integer.

$(1, 2) \notin R$ because $\frac{1-2}{2} = \frac{-1}{2},$ which is not an integer.

$(1, 3) \in R$ because $\frac{1-3}{2} = \frac{-2}{2} = -1,$ which is an integer.

$(2, 1) \notin R$ because $\frac{2-1}{2} = \frac{1}{2},$ which is not an integer.

Example 1.3.1 – *Solution (2/2)*

continued

(2, 2) $\in$ *R* because $\frac{2-2}{2} = \frac{0}{2} = 0$, which is an integer.

(2, 3) $\notin$ *R* because $\frac{2-3}{2} = \frac{-1}{2}$, which is not an integer.

Thus

$$R = \{(1, 1), (1, 3), (2, 2)\}$$

b. Yes, 1 *R* 3 because (1, 3) $\in$ *R*.

No, 2 $\not R$ 3 because (2, 3) $\notin$ *R*.

Yes, 2 *R* 2 because (2, 2) $\in$ *R*.

c. The domain of *R* is {1, 2} and the co-domain is {1, 2, 3}.

Example 1.3.2 – *The Circle Relation*

Define a relation C from $\mathbf{R}$ to $\mathbf{R}$ as follows: For any $(x, y) \in \mathbf{R} \times \mathbf{R}$,

$(x, y) \in C$ means that $x^2 + y^2 = 1.$

a. Is $(1, 0) \in C$? Is $(0, 0) \in C$? Is $\left(-\frac{1}{2}, \frac{\sqrt{3}}{2}\right) \in C$? Is $-2\ C\ 0$? Is $0\ C\ (-1)$? Is $1\ C\ 1$?

b. What are the domain and co-domain of C?

c. Draw a graph for C by plotting the points of C in the Cartesian plane.

Example 1.3.2 – *Solution (1/2)*

a. Yes, (1, 0) $\in$ *C* because $1^2 + 0^2 = 1$.

No, (0, 0) $\notin$ *C* because $0^2 + 0^2 = 0 \neq 1$.

Yes, $\left(-\frac{1}{2}, \frac{\sqrt{3}}{2}\right) \in C$ because $\left(-\frac{1}{2}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2 = \frac{1}{4} + \frac{3}{4} = 1$.

No, $-2 \not C\, 0$ because $(-2)^2 + 0^2 = 4 \neq 1$.

Yes, 0 *C* (−1) because $0^2 + (-1)^2 = 1$.

No, $1 \not C\, 1$ because $1^2 + 1^2 = 2 \neq 1$.

Example 1.3.2 – *Solution (2/2)* continued

b. The domain and co-domain of *C* are both **R**, the set of all real numbers.

c.

Arrow Diagram of a Relation

Arrow Diagram of a Relation

Suppose R is a relation from a set A to a set B. The **arrow diagram for R** is obtained as follows:

1. Represent the elements of A as points in one region and the elements of B as points in another region.

2. For each x in A and y in B, draw an arrow from x to y if, and only if, x is related to y by R. Symbolically:

   Draw an arrow from x to y

   if, and only if, $x \, R \, y$

   if, and only if, $(x, y) \in R$.

Example 1.3.3 – *Arrow Diagrams of Relations*

Let $A = \{1, 2, 3\}$ and $B = \{1, 2, 3\}$ and define relations S and T from A to B as follows:

For every $(x, y) \in A \times B$,

$(x, y) \in S$ means that $x < y$ *S* is a "less than" relation.

$T = \{(2, 1), (2, 5)\}$.

Draw arrow diagrams for S and T.

Example 1.3.3 – *Solution*

These example relations illustrate that it is possible to have several arrows coming out of the same element of *A* pointing in different directions. Also, it is quite possible to have an element of *A* that does not have an arrow coming out of it.

Functions

Functions (1/2)

Definition

A function F from a set A to a set B is a relation with domain A and co-domain B that satisfies the following two properties:

1. For every element x in A, there is an element y in B such that $(x, y) \in F$.
2. For all elements x in A and y and z in B,

$$\text{if} \quad (x, y) \in F \text{ and } (x, z) \in F, \quad \text{then} \quad y = z.$$

Functions (2/2)

Properties (1) and (2) can be stated less formally as follows: A relation F from A to B is a function if, and only if:

1. Every element of A is the first element of an ordered pair of F.
2. No two distinct ordered pairs in F have the same first element.

Function Notation

If A and B are sets and F is a function from A to B, then given any element x in A, the unique element in B that is related to x by F is denoted $\boldsymbol{F(x)}$, which is read "***F* of *x***."

Example 1.3.4 – *Functions and Relations on Finite Sets*

Let $A = \{2, 4, 6\}$ and $B = \{1, 3, 5\}$. Which of the relations R, S, and T defined below are functions from A to B?

a. $R = \{(2, 5), (4, 1), (4, 3), (6, 5)\}$

b. For every $(x, y) \in A \times B$, $(x, y) \in S$ means that $y = x + 1$.

c. T is defined by the arrow diagram

Example 1.3.4 – *Solution (1/3)*

a. *R* is not a function because it does not satisfy property (2). The ordered pairs (4, 1) and (4, 3) have the same first element but different second elements. You can see this graphically if you draw the arrow diagram for *R*. There are two arrows coming out of 4: One points to 1 and the other points to 3.

Example 1.3.4 – *Solution (2/3)*

b. S is not a function because it does not satisfy property (1). It is not true that every element of A is the first element of an ordered pair in S. For example, $6 \in A$ but there is no y in B such that $y = 6 + 1 = 7$. You can also see this graphically by drawing the arrow diagram for S.

Example 1.3.4 – *Solution (3/3)*

c. T is a function: Each element in {2, 4, 6} is related to some element in {1, 3, 5}, and no element in {2, 4, 6} is related to more than one element in {1, 3, 5}. When these properties are stated in terms of the arrow diagram, they become (1) there is an arrow coming out of each element of the domain, and (2) no element of the domain has more than one arrow coming out of it.

So you can write $T(2) = 5$, $T(4) = 1$, and $T(6) = 1$.

Example 1.3.5 – *Functions and Relations on Sets of Strings*

Let $A = \{a, b\}$ and let S be the set of all strings over A.

a. Define a relation L from S to $\mathbf{Z}^{nonneg}$ as follows: For every string s in S and for every nonnegative integer n,

$$(s, n) \in L \text{ means that the length of } s \text{ is } n.$$

Observe that L is a function because every string in S has one and only one length. Find $L(abaaba)$ and $L(bbb)$.

Example 1.3.5 – *Functions and Relations on Sets of Strings (1/2)*

b. Define a relation C from S to S as follows: For all strings s and t in S,

$$(s, t) \in C \text{ means that } t = as,$$

where as is the string obtained by appending a on the left of the characters in s. (C is called **concatenation** by a on the left.)

Example 1.3.5 – *Functions and Relations on Sets of Strings (2/2)*

Observe that C is a function because every string in S consists entirely of a's and b's and adding an additional a on the left creates a new strong that also consists of a's and b's and thus is also in S. Find $C(abaaba)$ and $C(bbb)$.

Example 1.3.5 – *Solution*

a. $L(abaaba) = 6$ and $L(bbb) = 3$

b. $C(abaaba) = aabaaba$ and $C(bbb) = abbb$

Function Machines

Function Machines (1/3)

Another useful way to think of a function is as a machine. Suppose f is a function from X to Y and an input x of X is given. Imagine f to be a machine that processes x in a certain way to produce the output $f(x)$. This is illustrated in Figure 1.3.1.

Figure 1.3.1

Example 1.3.6 – *Functions Defined by Formulas (1/4)*

The **squaring function** *f* from **R** to **R** is defined by the formula $f(x) = x^2$ for every real number *x*. This means that no matter what real number input is substituted for *x*, the output of *f* will be the square of that number. This idea can be represented by writing $f(\square) = \square^2$.

In other words, *f* sends each real number *x* to x^2, or, symbolically, $f: x \longrightarrow x^2$.

Example 1.3.6 – *Functions Defined by Formulas (2/4)*

The **successor function** g from **Z** to **Z** is defined by the formula $g(n) = n + 1$.

Thus, no matter what integer is substituted for n, the output of g will be that number plus 1: $g(\square) = \square + 1.$

In other words, g sends each integer n to $n + 1$, or, symbolically, $g: n \rightarrow n + 1$.

Example 1.3.6 – *Functions Defined by Formulas (3/4)*

An example of a **constant function** is the function *h* from **Q** to **Z** defined by the formula $h(r) = 2$ for all rational numbers *r*. This function sends each rational number *r* to 2. In other words, no matter what the input, the output is always 2: $h(\square) = 2$ or $h\colon r \longrightarrow 2$.

Example 1.3.6 – *Functions Defined by Formulas (4/4)*

The functions *f*, *g*, and *h* are represented by the function machines in Figure 1.3.2.

Figure 1.3.2

Function Machines (2/3)

A function is an entity in its own right. It can be thought of as a certain relationship between sets or as an input/output machine that operates according to a certain rule.

This is the reason why a function is generally denoted by a single symbol or string of symbols, such as *f*, *G*, of log, or sin.

Function Machines (3/3)

A relation is a subset of a Cartesian product and a function is a special kind of relation. Specifically, if *f* and *g* are functions from a set *A* to a set *B*, then

$$f = \{(x, y) \in A \times B \mid y = f(x)\} \quad \text{and} \quad g = \{(x, y) \in A \times B \mid y = g(x)\}.$$

It follows that

$\boldsymbol{f}$ **equals** $\boldsymbol{g}$, written $\boldsymbol{f} = \boldsymbol{g}$, if, and only if, $\boldsymbol{f(x)} = \boldsymbol{g(x)}$ for all x in A.

Example 1.3.7 – *Equality of Functions*

Define functions *f* and *g* from **R** to **R** by the following formulas:

$$f(x) = |x| \quad \text{for every } x \in \mathbf{R}.$$
$$g(x) = \sqrt{x^2} \quad \text{for every } x \in \mathbf{R}.$$

Does *f* = *g*?

Example 1.3.7 – *Solution*

Yes. Because the absolute value of any real number equals the square root of its square, $|x| = \sqrt{x^2}$ for all $x \in \mathbf{R}$. Hence $f = g$.

CHAPTER 1

SPEAKING MATHEMATICALLY

Copyright © Cengage Learning. All rights reserved.

1.4 The Language of Graphs

Copyright © Cengage Learning. All rights reserved.

The Language of Graphs (1/8)

Imagine an organization that wants to set up teams of three to work on some projects.

In order to maximize the number of people on each team who had previous experience working together successfully, the director asked the members to provide names of their previous partners.

The Language of Graphs (2/8)

This information is displayed below both in a table and in a diagram.

Name	Previous Partners
Ana	Dan, Flo
Bev	Cai, Flo, Hal
Cai	Bev, Flo
Dan	Ana, Ed
Ed	Dan, Hal
Flo	Cai, Bev, Ana
Gia	Hal
Hal	Gia, Ed, Bev, Ira
Ira	Hal

The Language of Graphs (3/8)

From the diagram, it is easy to see that Bev, Cai, and Flo are a group of three previous partners, and so it would be reasonable for them to form one of these teams. The drawing below shows the result when these three names are removed from the diagram.

The Language of Graphs (4/8)

This drawing shows that placing Hal on the same team as Ed would leave Gia and Ira on a team where they would not have a previous partner.

However, if Hal is placed on a team with Gia and Ira, then the remaining team would consist of Ana, Dan, and Ed, and everyone on both teams would be working with a previous partner.

The Language of Graphs (5/8)

Drawings such as these are illustrations of a structure known as a *graph*. The dots are called *vertices* (plural of *vertex*) and the line segments joining vertices are called *edges*.

As you can see from the first drawing, it is possible for two edges to cross at a point that is not a vertex. Note also that the type of graph described here is quite different from the "graph of an equation" or the "graph of a function."

The Language of Graphs (6/8)

In general, a graph consists of a set of vertices and a set of edges connecting various pairs of vertices. The edges may be straight or curved and should either connect one vertex to another or a vertex to itself, as shown below.

The Language of Graphs (7/8)

In this drawing, the vertices are labeled with *v*'s and the edges with *e*'s. When an edge connects a vertex to itself (as e_5 does), it is called a *loop*.

When two edges connect the same pair of vertices (as e_2 and e_3 do), they are said to be *parallel*. It is quite possible for a vertex to be unconnected by an edge to any other vertex in the graph (as v_5 is), and in that case the vertex is said to be *isolated*.

The Language of Graphs (8/8)

The formal definition of a graph follows.

Definition

A **graph** G consists of two finite sets: a nonempty set $V(G)$ of **vertices** and a set $E(G)$ of **edges**, where each edge is associated with a set consisting of either one or two vertices called its **endpoints**. The correspondence from edges to endpoints is called the **edge-endpoint function**.

An edge with just one endpoint is called a **loop**, and two or more distinct edges with the same set of endpoints are said to be **parallel**. An edge is said to **connect** its endpoints; two vertices that are connected by an edge are called **adjacent**; and a vertex that is an endpoint of a loop is said to be **adjacent to itself**.

An edge is said to be **incident on** each of its endpoints, and two edges incident on the same endpoint are called **adjacent**. A vertex on which no edges are incident is called **isolated**.

Example 1.4.1 – *Terminology (1/2)*

Consider the following graph:

a. Write the vertex set and the edge set, and give a table showing the edge-endpoint function.

Example 1.4.1 – *Terminology (2/2)*

b. Find all edges that are incident on v_1, all vertices that are adjacent to v_1, all edges that are adjacent to e_1, all loops, all parallel edges, all vertices that are adjacent to themselves, and all isolated vertices.

Example 1.4.1 – *Solution (1/2)*

a. vertex set = $\{v_1, v_2, v_3, v_4, v_5, v_6\}$

edge set = $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$

edge-endpoint function:

Edge	Endpoints
e_1	$\{v_1, v_2\}$
e_2	$\{v_1, v_3\}$
e_3	$\{v_1, v_3\}$
e_4	$\{v_2, v_3\}$
e_5	$\{v_5, v_6\}$
e_6	$\{v_5\}$
e_7	$\{v_6\}$

Example 1.4.1 – *Solution (2/2)*

b. e_1, e_2, and e_3 are incident on v_1.

v_2 and v_3 are adjacent to v_1.

e_2, e_3, and e_4 are adjacent to e_1.

e_6 and e_7 are loops.

e_2 and e_3 are parallel.

v_5 and v_6 are adjacent to themselves.

v_4 is an isolated vertex.

Example 1.4.2 – *Drawing More Than One Picture for a Graph (1/2)*

Consider the graph specified as follows:

vertex set = $\{v_1, v_2, v_3, v_4\}$

edge set = $\{e_1, e_2, e_3, e_4\}$

edge-endpoint function:

Edge	Endpoints
e_1	$\{v_1, v_3\}$
e_2	$\{v_2, v_4\}$
e_3	$\{v_2, v_4\}$
e_4	$\{v_3\}$

Example 1.4.2 – *Drawing More Than One Picture for a Graph (2/2)*

Both drawings (a) and (b) shown below are pictorial representations of this graph.

(a) (b)

Example 1.4.3 – *Labeling Drawings to Show They Represent the Same Graph*

Consider the two drawings shown in Figure 1.4.1. Label vertices and edges in such a way that both drawings represent the same graph.

Figure 1.4.1

Example 1.4.3 – *Solution (1/3)*

Imagine putting one end of a piece of string at the top vertex of Figure 1.4.1(a) (call this vertex v_1), then laying the string to the next adjacent vertex on the lower right (call this vertex v_2), then laying it to the next adjacent vertex on the upper left (v_3), and so forth, returning finally to the top vertex v_1. Call the first edge e_1, the second e_2, and so forth, as shown below.

Example 1.4.3 – *Solution (2/3)*

Now imagine picking up the piece of string, together with its labels, and repositioning it as follows:

Example 1.4.3 – *Solution (3/3)*

This is the same as Figure 1.4.1(b), so both drawings represent the graph with vertex set $\{v_1, v_2, v_3, v_4, v_5\}$, edge set $\{e_1, e_2, e_3, e_4, e_5\}$, and edge-endpoint function as follows:

Edge	Endpoints
e_1	$\{v_1, v_2\}$
e_2	$\{v_2, v_3\}$
e_3	$\{v_3, v_4\}$
e_4	$\{v_4, v_5\}$
e_5	$\{v_5, v_1\}$

Examples of Graphs

Example 1.4.4 – *Using a Graph to Represent a Network (1/2)*

Telephone, electric power, gas pipeline, and air transport systems can all be represented by graphs, as can computer networks—from small local area networks to the global Internet system that connects millions of computers worldwide.

Questions that arise in the design of such systems involve choosing connecting edges to minimize cost, optimize a certain type of service, and so forth.

Example 1.4.4 – *Using a Graph to Represent a Network (2/2)*

A typical network, called a *hub-and-spoke model*, is shown below.

Examples of Graphs (1/2)

A *directed graph* is like an (undirected) graph except that each edge is associated with an *ordered pair* of vertices rather than a *set* of vertices. Thus each edge of a directed graph can be drawn as an arrow going from the first vertex to the second vertex of the ordered pair.

Definition

A **directed graph**, or **digraph**, consists of two finite sets: a nonempty set $V(G)$ of vertices and a set $D(G)$ of directed edges, where each is associated with an ordered pair of vertices called its **endpoints**. If edge e is associated with the pair (v, w) of vertices, then e is said to be the (**directed**) **edge** from v to w.

Example 1.4.6 – *Using a Graph to Represent Knowledge (1/3)*

In many applications of artificial intelligence, a knowledge base of information is collected and represented inside a computer. Because of the way the knowledge is represented and because of the properties that govern the artificial intelligence program, the computer is not limited to retrieving data in the same form as it was entered; it can also derive new facts from the knowledge base by using certain built-in rules of inference. For example, from the knowledge that the *Los Angeles Times* is a big-city daily and that a big-city daily contains national news, an artificial intelligence program could infer that the *Los Angeles Times* contains national news.

Example 1.4.6 – *Using a Graph to Represent Knowledge (2/3)*

The directed graph shown in Figure 1.4.2 is a pictorial representation for a simplified knowledge base about periodical publications.

Figure 1.4.2

Example 1.4.6 – *Using a Graph to Represent Knowledge (3/3)*

According to this knowledge base, what paper finish does the *New York Times* use?

Example 1.4.6 – *Solution*

The arrow going from *New York Times* to big-city daily (labeled “instanceof”) shows that the *New York Times* is a big-city daily. The arrow going from big-city daily to newspaper (labeled “is-a”) shows that a big-city daily is a newspaper.

The arrow going from newspaper to matte (labeled “paper-finish”) indicates that the paper finish on a newspaper is matte. Hence it can be inferred that the paper finish on the *New York Times* is matte.

Examples of Graphs (2/2)

Definition

Let G be a graph and v a vertex of G. The **degree of v**, denoted **deg(v)**, equals the number of edges that are incident on v, with an edge that is a loop counted twice.

Example 1.4.8 – *Degree of a Vertex*

Find the degree of each vertex of the graph *G* shown below.

Example 1.4.8 – *Solution*

$\deg(v_1) = 0$ since no edge is incident on v_1 (v_1 is isolated).

$\deg(v_2) = 2$ since both e_1 and e_2 are incident on v_2.

$\deg(v_3) = 4$ since e_1 and e_2 are incident on v_3 and the loop e_3 is also incident on v_3 (and contributes 2 to the degree of v_3).

Example 1.4.9 – *Using a Graph to Color a Map*

Imagine that the diagram shown below is a map with countries labeled *A*–*J*. Show that you can color the map so that no two adjacent countries have the same color.

Example 1.4.9 – *Solution (1/10)*

Notice that coloring the map does not depend on the sizes or shapes of the countries, but only on which countries are adjacent to which. So, to figure out a coloring, you can draw a graph, as shown below, where vertices represent countries and where edges are drawn between pairs of vertices that represent adjacent countries.

Example 1.4.9 – *Solution (2/10)*

Coloring the vertices of the graph will translate to coloring the countries on the map.

As you assign colors to vertices, a relatively efficient strategy is, at each stage, to focus on an uncolored vertex that has maximum degree, in other words that is connected to a maximum number of other uncolored vertices.

Example 1.4.9 – *Solution (3/10)*

If there is more than one such vertex, it does not matter which you choose because there are often several acceptable colorings for a given graph.

Example 1.4.9 – *Solution (4/10)*

For this graph, both *C* and *H* have maximum degree so you can choose one, say, *C*, and color it, say, blue.

Now since *A*, *F*, *I*, and *J* are not connected to *C*, some of them may also be colored blue, and, because *J* is connected to a maximum number of others, you could start by coloring it blue.

Example 1.4.9 – *Solution (5/10)*

Then *F* is the only remaining vertex not connected to either *C* or *J*, so you can also color *F* blue. The drawing below shows the graph with vertices *C*, *J*, and *F* colored blue.

Example 1.4.9 – *Solution (6/10)*

Since the vertices adjacent to *C*, *J*, and *F* cannot be colored blue, you can simplify the job of choosing additional colors by removing *C*, *J*, and *F* and the edges connecting them to adjacent vertices. The result is shown in Figure 1.4.4a.

Figure 1.4.4(a)

Example 1.4.9 – *Solution (7/10)*

In the simplified graph again choose a vertex that has a maximum degree, namely *H*, and give it a second color, say, gray.

Since *A*, *D*, and *E* are not connected to *H*, some of them may also be colored gray, and, because *E* is connected to a maximum number of these vertices, you could start by coloring *E* gray.

Example 1.4.9 – *Solution (8/10)*

Then *A* is not connected to *E*, and so you can also color *A* gray. This is shown in Figure 1.4.4b.

Figure 1.4.4(b)

Example 1.4.9 – *Solution (9/10)*

The drawing below shows the original graph with vertices *C*, *J*, and *F* colored blue, vertices *H*, *A*, and *E*, colored gray, and the remaining vertices colored black. You can check that no two adjacent vertices have the same color.

Example 1.4.9 – *Solution (10/10)*

Translating the graph coloring back to the original map gives the following picture in which no two adjacent countries have the same color.

CHAPTER 2

THE LOGIC OF COMPOUND STATEMENTS

Copyright © Cengage Learning. All rights reserved.

2.1 Logical Form and Logical Equivalence

Copyright © Cengage Learning. All rights reserved.

Logical Form and Logical Equivalence (1/4)

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*.

Logical Form and Logical Equivalence (2/4)

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.

Logical Form and Logical Equivalence (3/4)

Consider the following two arguments. They have very different content but their logical form is the same. To help make this clear, we use letters like *p*, *q*, and *r* to represent component sentences; we let the expression "not *p*" refer to the sentence "It is not the case that *p*"; and we let the symbol $\therefore$ stand for the word "therefore."

Argument 1

If $\overbrace{\text{the bell rings}}^{p}$ or $\overbrace{\text{the flag drops}}^{q}$, then $\overbrace{\text{the race is over}}^{r}$.

$\therefore$ If $\overbrace{\text{the race is not over}}^{\text{not } r}$, then $\overbrace{\text{the bell hasn't rung}}^{\text{not } p}$ and $\overbrace{\text{the flag hasn't dropped}}^{\text{not } q}$.

Logical Form and Logical Equivalence (4/4)

Argument 2

$$\text{If } \overbrace{x = 2}^{p} \text{ or } \overbrace{x = -2}^{q}, \text{ then } \overbrace{x^2 = 4}^{r}.$$

$$\therefore \text{ If } \overbrace{x^2 \neq 4}^{\text{not } r}, \text{ then } \overbrace{x \neq 2}^{\text{not } p} \text{ and } \overbrace{x \neq -2}^{\text{not } q}.$$

The common form of the arguments is

If *p* or *q*, then *r*.

∴ 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.

a. 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.

b. If logic is easy or ___(1)___, then ___(2)___. I will study hard. Therefore, I will get an A in this course.

Example 2.1.1 – *Solution*

a. I (will) study hard.

b. I will get an A in this course.

Common form: If p or q, then r.

q.

Therefore, r.

Statements

Statements (1/2)

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

$$x^2 + 2 = 11$$

depends on the value of x. For some values of x, it is true ($x = 3$ and $x = -3$), whereas for other values it is false.

Statements (2/2)

Similarly, the truth or falsity of

$$x + y > 0$$

depends on the values of x and y. For instance, when $x = -1$ and $y = 2$ it is true, whereas when $x = -1$ and $y = 1$ it is false.

Compound Statements

Compound Statements (1/3)

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*." In some computer languages the symbol $\neg$ is used in place of $\sim$.

Compound Statements (2/3)

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.

Compound Statements (3/3)

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.

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.”

a. It is not hot but it is sunny.

b. It is neither hot nor sunny.

Example 2.1.2 – *Solution*

a. The given sentence is equivalent to “It is not hot and it is sunny,” which can be written symbolically as $\sim h \wedge s$.

b. 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 $\sim h \wedge \sim s$.

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. $0 < x \leq 3$

Example 2.1.3 – *Solution*

a. $q \vee r$

b. $p \wedge q$

c. $p \wedge (q \vee r)$

Truth Values

Truth Values (1/5)

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*.

p	$\sim p$
T	F
F	T

Truth Table for $\sim p$

Truth Values (2/5)

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.

Truth Values (3/5)

The table is obtained by considering the four possible combinations of truth values for *p* and *q.*

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

Truth Table for $p \wedge q$

Truth Values (4/5)

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 because an *and* statement is true only when both components are true.

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.

Truth Values (5/5)

Here is the truth table for disjunction:

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

Truth Table for $p \vee q$

Evaluating the Truth of More General Compound Statements

Evaluating the Truth of More General Compound Statements (1/2)

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*).

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.

Evaluating the Truth of More General Compound Statements (2/2)

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

Example 2.1.4 – *Solution (1/3)*

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.

Example 2.1.4 – *Solution (2/3)*

Finally, fill in the $(p \vee q) \wedge \sim (p \wedge q)$ column by considering the truth values for an *and* statement together with the truth values for $p \vee q$ and $\sim (p \wedge q)$.

Since an *and* statement is true only when both components are true and since rows 2 and 3 are the only two rows where both $p \vee q$ and $\sim (p \wedge q)$ are true, put T in rows 2 and 3 and F in the remaining rows.

Example 2.1.4 – *Solution (3/3)*

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

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

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

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

Example 2.1.5 – *Solution (1/3)*

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.

Example 2.1.5 – *Solution (2/3)*

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.

Example 2.1.5 – *Solution (3/3)*

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

Logical Equivalence

Logical Equivalence (1/3)

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.

Logical Equivalence (2/3)

Testing Whether Two Statement Forms *P* and *Q* Are Logically Equivalent

1. Construct a truth table with one column for the truth values of P and another column for the truth values of Q.
2. 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.
   a. 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.
   b. 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.

Example 2.1.6 – *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

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.

Example 2.1.7 – *Solution (1/2)*

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	$\neq$	F
F	T	T	F	F	T	$\neq$	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

Example 2.1.7 – *Solution (2/2)*

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,

$\sim p \wedge \sim q$ is "$0 \nless 1$ and $1 \nless 0$,"

which is false.

Example 2.1.7 – *Solution*

continued

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.

Logical Equivalence (3/3)

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:

a. John is 6 feet tall and he weighs at least 200 pounds.

b. The bus was late or Tom's watch was slow.

Example 2.1.9 – *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."

Example 2.1.10 – *Inequalities and De Morgan's Laws*

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

Example 2.1.10 – *Solution (1/2)*

The given statement is equivalent to

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

By De Morgan's laws, the negation is

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

which is equivalent to

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

Example 2.1.10 – *Solution (2/2)*

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

Tautologies and Contradictions

Tautologies and Contradictions (1/1)

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 **contradication** 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**.

Example 2.1.12 – *Tautologies and Contradictions*

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

Example 2.1.12 – *Solution*

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

all T's, so $p \vee \sim p$ is a tautology

all F's, so $p \wedge \sim p$ is a 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 $p \wedge \mathbf{c} \equiv \mathbf{c}$.

Example 2.1.13 – *Solution*

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

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

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

Summary of Logical Equivalences

Summary of Logical Equivalences (1/1)

Theorem 2.1.1 Logical Equivalences

Given any statement variables p, q, and r, a tautology $\mathbf{t}$ and a contradiction $\mathbf{c}$, the following logical equivalences hold.

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

Example 2.1.14 – *Simplifying Statement Forms*

Use Theorem 2.1.1 to verify the logical equivalence

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

Example 2.1.14 – *Solution (1/2)*

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.

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

Example 2.1.14 – *Solution (2/2)*

$\equiv$	$p \vee (q \wedge \sim q)$	by the commutative law for $\wedge$
$\equiv$	$p \vee \mathbf{c}$	by the negation law
$\equiv$	p	by the identity law

CHAPTER 2

THE LOGIC OF COMPOUND STATEMENTS

Copyright © Cengage Learning. All rights reserved.

2.2 Conditional Statements

Copyright © Cengage Learning. All rights reserved.

Conditional Statements (1/2)

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**.

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.

Conditional Statements (2/2)

In expressions that include $\rightarrow$ as well as other logical operators such as $\wedge$, $\vee$, and $\neg$, the **order of operations** is that $\rightarrow$ is performed last.

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

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

Example 2.2.2 – *Solution (1/2)*

By the order of operations given, the following two expressions are equivalent:

$$p \vee \sim q \rightarrow \sim p \text{ and } (p \vee (\sim q)) \rightarrow (\sim p),$$

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 $\vee$.

Example 2.2.2 – *Solution (2/2)*

Finally, fill in the $p \vee \sim q \rightarrow \sim p$ column using the definition of $\rightarrow$.

		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

Logical Equivalences Involving $\rightarrow$

Example 2.2.3 – *Division into Cases: Showing That* $p \vee q \rightarrow r \equiv (p \rightarrow r) \wedge (q \rightarrow r)$

Use truth tables to show the logical equivalence of the statement forms $p \vee q \rightarrow r$ and $(p \rightarrow r) \wedge (q \rightarrow r)$. Annotate the table with a sentence of explanation.

Example 2.2.3 – *Solution (1/2)*

First fill in the eight possible combinations of truth values for p, q, and r. Then fill in the columns for $p \vee q$, $p \rightarrow r$, and $q \rightarrow r$ using the definitions of *or* and *if-then*. For instance, the $p \rightarrow r$ column has F's in the second and fourth rows because these are the rows in which p is true and r is false.

Next fill in the $p \vee q \rightarrow r$ column using the definition of *if-then*. The rows in which the hypothesis $p \vee q$ is true and the conclusion r is false are the second, fourth, and sixth. So F's go in these rows and T's in all the others.

Example 2.2.3 – *Solution (2/2)*

The complete table shows that $p \vee q \rightarrow r$ and $(p \rightarrow r) \wedge (q \rightarrow r)$ have the same truth values for each combination of truth values of p, q, and r. Hence the two statement forms are logically equivalent.

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

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

Representation of If-Then as Or

Example 2.2.4 – *Application of the Equivalence between* $\sim p \vee q$ and $p \rightarrow q$

Rewrite the following statement in if-then form.

Either you get to work on time or you are fired.

Example 2.2.4 – *Solution (1/2)*

Let $\sim p$ be

You get to work on time.

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.

Example 2.2.4 – *Solution (2/2)*

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

The Negation of a Conditional Statement (1/2)

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:

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

The Negation of a Conditional Statement (2/2)

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

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

Example 2.2.5 – *Negations of If-Then Statements*

Write negations for each of the following statements:

a. If my car is in the repair shop, then I cannot get to class.

b. If Sara lives in Athens, then she lives in Greece.

Example 2.2.5 – *Solution*

a. My car is in the repair shop and I can get to class.

b. Sara lives in Athens and she does not live in Greece. (Sara might live in Athens, Georgia; Athens, Ohio; or Athens, Wisconsin.)

The Contrapositive of a Conditional Statement

The Contrapositive of a Conditional Statement (1/2)

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

$$\text{If } \sim q \text{ then } \sim p.$$

Symbolically,

$$\text{The contrapositive of } p \rightarrow q \text{ is } \sim q \rightarrow \sim p.$$

The Contrapositive of a Conditional Statement (2/2)

The fact is that

A conditional statement is logically equivalent to its contrapositive.

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.

Example 2.2.6 – *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.

The Converse and Inverse of a Conditional Statement

The Converse and Inverse of a Conditional Statement (1/2)

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,

$$\text{The converse of } p \rightarrow q \text{ is } q \rightarrow p,$$

and

$$\text{The inverse of } p \rightarrow q \text{ is } \sim p \rightarrow \sim q.$$

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.

Example 2.2.7 – *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.

The Converse and Inverse of a Conditional Statement (2/2)

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.

Only If and the Biconditional

Only If and the Biconditional (1/5)

Definition

If 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.

Example 2.2.8 – *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.

Only If and the Biconditional (2/5)

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**.

Only If and the Biconditional (3/5)

The biconditional has the following truth table:

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

Truth Table for $p \leftrightarrow q$

Only If and the Biconditional (4/5)

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.

Only If and the Biconditional (5/5)

The full hierarchy of operations for the five logical operators is shown below.

Order of Operations for Logical Operators

1. $\sim$	Evaluate negations first.
2. $\wedge, \vee$	Evaluate $\wedge$ and $\vee$ second. When both are present, parentheses may be needed.
3. $\rightarrow, \leftrightarrow$	Evaluate $\rightarrow$ and $\leftrightarrow$ third. When both are present, parentheses may be needed.

Example 2.2.9 – *If and Only If*

Rewrite the following statement as a conjunction of two if-then statements:

This computer program is correct if, and only if, it produces correct answers for all possible sets of input data.

Example 2.2.9 – *Solution*

If this program is correct, then it produces the correct answers for all possible sets of input data; and if this program produces the correct answers for all possible sets of input data, then it is correct.

Necessary and Sufficient Conditions

Necessary and Sufficient Conditions (1/1)

Definition

If r and s are statements:

r is a **sufficient condition** for s	means	"if r then s."
r is a **necessary condition** for s	means	"if not r then not s."

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.

Example 2.2.11 – *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.

Example 2.2.12 – *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.

CHAPTER 2

THE LOGIC OF COMPOUND STATEMENTS

Copyright © Cengage Learning. All rights reserved.

2.3 Valid and Invalid Arguments

Copyright © Cengage Learning. All rights reserved.

Valid and Invalid Arguments (1/2)

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.

Valid and Invalid Arguments (2/2)

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 (1/2)*

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.

Example 2.3.1 – *Determining Validity or Invalidity (2/2)*

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.

$$p \rightarrow q \vee \sim r$$

$$q \rightarrow p \wedge r$$

$$\therefore p \rightarrow r$$

Example 2.3.1 – *Solution (1/2)*

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.

Example 2.3.1 – *Solution (2/2)*

						premises		conclusion
p	q	r	$\sim r$	$q \vee \sim r$	$p \wedge r$	$p \to q \vee \sim r$	$q \to p \wedge r$	$p \to r$
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

Modus Ponens and Modus Tollens (1/2)

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

$\therefore q$

Modus Ponens and Modus Tollens (2/2)

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.

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.

$\therefore$ _____.

Example 2.3.2 – *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

Additional Valid Argument Forms: Rules of Inference (1/1)

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 (1/2)*

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*.

Example 2.3.3 – *Generalization (2/2)*

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.

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 (1/2)*

The following argument forms are valid:

a. $p \wedge q$

$\therefore p$

b. $p \wedge q$

$\therefore q$

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.

Example 2.3.4 – *Specialization (2/2)*

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.

Example 2.3.5 – *Elimination (1/2)*

The following argument forms are valid:

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

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

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,

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

Example 2.3.5 – *Elimination (2/2)*

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

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

Example 2.3.6 – *Transitivity (1/2)*

The following argument form is valid:

$$p \rightarrow q$$
$$q \rightarrow r$$
$$\therefore p \rightarrow r$$

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.

Example 2.3.6 – *Transitivity (2/2)*

In the example below suppose n is a particular integer.

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

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

$\therefore$ If n is divisible by 18, then the sum of the digits of n is divisible by 9.

Example 2.3.7 – *Proof by Division into Cases (1/2)*

The following argument forms are valid:

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

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 that is not zero. The trichotomy property of the real numbers says that any real number is positive, negative, or zero.

Example 2.3.7 – *Proof by Division into Cases (2/2)*

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 $x^2 > 0.$

$\therefore x^2 > 0.$

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 (1/2)*

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

a. If I was reading my class notes in the kitchen, then my glasses are on the kitchen table.

b. If my glasses are on the kitchen table, then I saw them at breakfast.

c. I did not see my glasses at breakfast.

Example 2.3.8 – *Application: A More Complex Deduction (2/2)*

d. I was reading my class notes in the living room or I was reading my class notes in the kitchen

e. If I was reading my class notes in the living room then my glasses are on the coffee table.

Where are the glasses?

Example 2.3.8 – *Solution (1/3)*

Let *RK* = I was reading my class notes in the kitchen.

GK = My glasses are on the kitchen table.

SB = I saw my glasses at breakfast.

RL = I was reading my class notes 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)

Example 2.3.8 – *Solution (2/3)*

	$GK \rightarrow SB$	by (b)
	$\therefore RK \rightarrow SB$	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)
	$\therefore$	by elimination
	RL	

Example 2.3.8 – *Solution (3/3)*

4.	$RL \rightarrow GC$	by (e)
	RL	by the conclusion of (3)
	$\therefore$	by modus ponens
	GC	

Thus the glasses are on the coffee table.

Fallacies

Fallacies (1/3)

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.

Fallacies (2/3)

We 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.

Example 2.3.9 – *Solution (1/2)*

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.

Example 2.3.9 – *Solution (2/2)*

The general form of the previous argument is as follows:

$$p \rightarrow q$$
$$q$$
$$\therefore p$$

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.

Example 2.3.10 – *Inverse Error*

Consider the following argument:

If these two vertices are adjacent, then they do not have the same color.

These two vertices are not adjacent.

$\therefore$ These two vertices have the same color.

Note that this argument has the following form:

$$p \rightarrow q$$

$$\sim p$$

$$\therefore \sim q$$

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 Canada is north of the United States, then temperatures in Canada never rise above freezing.

Canada is north of the United States.

$\therefore$ Temperatures in Canada never rise above freezing.

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.

Fallacies (3/3)

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.

Contradictions and Valid Arguments

Contradictions and Valid Arguments (1/1)

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 **c** is a contradiction

$\therefore\ p$

Example 2.3.13 – *Solution*

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

			premises	conclusion
p	$\sim p$	c	$\sim p \rightarrow c$	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.

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*?

Example 2.3.14 – *Solution (1/2)*

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

Example 2.3.14 – *Solution (2/2)*

$\therefore$ *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.

$\therefore$ What *A* says is false.

$\therefore$ *B* is not a knight.

$\therefore$ *B* is also a knave.

by elimination

Summary of Rules of Inference

Summary of Rules of Inference (1/1)

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

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

Valid Argument Forms

Table 2.3.1

CHAPTER 2

THE LOGIC OF COMPOUND STATEMENTS

Copyright © Cengage Learning. All rights reserved.

2.4 Application: Digital Logic Circuits

Copyright © Cengage Learning. All rights reserved.

Black Boxes and Gates

Black Boxes and Gates (1/8)

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.

Black Boxes and Gates (2/8)

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.

Black Boxes and Gates (3/8)

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.

Black Boxes and Gates (4/8)

One possible correspondence of input to output signals is as follows:

Input			Output
P	*Q*	*R*	*S*
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

An Input/Output Table

Black Boxes and Gates (5/8)

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

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.

Black Boxes and Gates (6/8)

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.

Black Boxes and Gates (7/8)

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.

Type of Gate	Symbolic Representation	Action
NOT		Input: *P*; Output: *R* 1 → 0 0 → 1
AND		Input: *P*, *Q*; Output: *R* 1 1 → 1 1 0 → 0 0 1 → 0 0 0 → 0
OR		Input: *P*, *Q*; Output: *R* 1 1 → 1 1 0 → 1 0 1 → 1 0 0 → 0

Figure 2.4.3

Black Boxes and Gates (8/8)

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.

The Input/Output Table for a Circuit

The Input/Output Table for a Circuit (1/1)

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

Example 2.4.1 – *Solution (1/2)*

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.

Example 2.4.1 – *Solution (2/2)*

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.

Example 2.4.2 – *Constructing the Input/Output Table for a Circuit*

Construct the input/output table for the following circuit.

Example 2.4.2 – *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

The Boolean Expression Corresponding to a Circuit

The Boolean Expression Corresponding to a Circuit (1/3)

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.

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**.

Example 2.4.3 – *Finding a Boolean Expression for a Circuit*

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

(a)

(b)

Example 2.4.3 – *Solution (1/2)*

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.

Example 2.4.3 – *Solution (2/2)*

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

The Boolean Expression Corresponding to a Circuit (2/3)

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.

The Boolean Expression Corresponding to a Circuit (3/3)

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

Input/Output Table for a Recognizer

The Circuit Corresponding to a Boolean Expression

The Circuit Corresponding to a Boolean Expression (1/1)

The following example shows how to construct a circuit corresponding to a Boolean expression.

The strategy is to work from the outermost part of the Boolean expression to the innermost part, adding logic gates that correspond to the operations in the expression as you move from right to left in the circuit diagram.

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

Example 2.4.4 – *Solution (1/4)*

a. Write the input variables in a column on the left side of the diagram. 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.

Example 2.4.4 – *Solution (2/4)*

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.

Example 2.4.4 – *Solution (3/4)*

b. To start constructing this circuit, put one AND-gate at the extreme right to correspond to the $\wedge$, which is the final operation between $((P \wedge Q) \wedge (R \wedge S))$ and T.

To the left of that gate put the AND-gate corresponding to the $\wedge$ between $P \wedge Q$ and $R \wedge S$.

To the left of that gate put the two AND-gates corresponding to the $\wedge$'s between P and Q and between R and S.

Example 2.4.4 – *Solution (4/4)*

The circuit is shown in Figure 2.4.4.

Figure 2.4.4

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

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

Example 2.4.5 – *Solution (1/4)*

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 rows 1, 3, and 4.

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 row 1 is $\boldsymbol{P} \wedge \boldsymbol{Q} \wedge \boldsymbol{R}$ 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.

Example 2.4.5 – *Solution (2/4)*

The expression for row 3 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 row 4 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

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

Example 2.4.5 – *Solution (3/4)*

The circuit corresponding to this expression has the diagram shown in Figure 2.4.7.

Figure 2.4.7

Example 2.4.5 – *Solution (4/4)*

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**.

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

Simplifying Combinational Circuits

Simplifying Combinational Circuits (1/3)

Consider the two combinational circuits shown in Figure 2.4.8.

Figure 2.4.8

Simplifying Combinational Circuits (2/3)

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

Input		Output
P	Q	R
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.

Simplifying Combinational Circuits (3/3)

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.

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.

Figure 2.4.8

Example 2.4.6 – *Solution (1/2)*

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,

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

Example 2.4.6 – *Solution (2/2)*

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.

NAND and NOR Gates

NAND and NOR Gates (1/3)

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.

NAND and NOR Gates (2/3)

The logical symbols corresponding to these gates are **|** (for NAND) and ↓ (for NOR), where **|** is called a **Sheffer stroke** and ↓ is called a **Peirce arrow**.

Thus

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

NAND and NOR Gates (3/3)

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

Type of Gate	Symbolic Representation	Action
NAND		
NOR		

NAND:

Input		Output
P	Q	$R = P \mid Q$
1	1	0
1	0	1
0	1	1
0	0	1

NOR:

Input		Output
P	Q	$R = P \downarrow Q$
1	1	0
1	0	0
0	1	0
0	0	1

Example 2.4.7 – *Rewriting Expressions Using the Sheffer Stroke*

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

a. $\sim P \equiv P \mid P$ and b. $P \vee Q \equiv (P \mid P) \mid (Q \mid Q)$.

Example 2.4.7 – *Solution*

a. $\sim P \equiv \sim(P \wedge P)$ by the idempotent law for $\vee$

$\equiv P|P$ by definition of $|$.

b. $P \vee Q \equiv \sim(\sim(P \vee Q))$ by the double negative law

$\equiv \sim(\sim P \wedge \sim Q)$ by De Morgan's laws

$\equiv \sim((P|P) \wedge (Q|Q))$ by part (a)

$\equiv (P|P)|(Q|Q)$ by definition of $|$.

CHAPTER 2

THE LOGIC OF COMPOUND STATEMENTS

Copyright © Cengage Learning. All rights reserved.

2.5 Application: Number Systems and Circuits for Addition

Copyright © Cengage Learning. All rights reserved.

Binary Representation of Numbers

Binary Representation of Numbers (1/2)

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.”)

Example 2.5.1 – *Binary Notation for Integers from 1 to 9*

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

Example 2.5.1 – *Solution*

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

Binary Representation of Numbers (2/2)

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

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

Powers of 2

Table 2.5.1

Example 2.5.2 – *Converting a Binary to a Decimal Number*

Represent 110101_2 in decimal notation.

Example 2.5.2 – *Solution*

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

Alternatively, the schema below may be used.

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

Represent 209 in binary notation.

Example 2.5.3 – *Solution (1/3)*

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.

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

Powers of 2

Table 2.5.1

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

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

Example 2.5.3 – *Solution (2/3)* continued

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

$$209_{10} = 128 + 64 + 16 + 1$$

$$= 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0.$$

Example 2.5.3 – *Solution (3/3)*

continued

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

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

Binary Addition and Subtraction

Binary Addition and Subtraction (1/1)

The computational methods of binary arithmetic are analogous to those of decimal arithmetic. In binary arithmetic the number 2 (which equals 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.

Example 2.5.4 – *Solution (1/2)*

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").

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

Example 2.5.4 – *Solution (2/2)*

continued

Thus the addition can be performed as follows:

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

Example 2.5.5 – *Subtraction in Binary Notation*

Subtract 1011_2 from 11000_2 using binary notation.

Example 2.5.5 – *Solution (1/2)*

In decimal subtraction the fact that $10_{10} - 1_{10} = 9_{10}$ is used to borrow across several columns.

For example, consider the following:

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

Example 2.5.5 – *Solution (2/2)*

continued

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:

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

Circuits for Computer Addition

Circuits for Computer Addition (1/9)

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 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**).

Circuits for Computer Addition (2/9)

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:* $(P \vee Q) \wedge \sim(P \wedge Q)$.

Circuits for Computer Addition (3/9)

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**.

HALF-ADDER

Circuit

Input/OutputTable

P	*Q*	Carry	Sum
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

Circuit to Add *P* + *Q*, Where *P* and *Q* Are Binary Digits

Figure 2.5.1

Circuits for Computer Addition (4/9)

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.

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

Circuits for Computer Addition (5/9)

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*.

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

Circuits for Computer Addition (6/9)

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:

$$\begin{array}{r} 1_2 \\ +\ \ 0_2 \\ +\ \ 1_2 \\ \hline 10_2 \end{array} \quad 1_2 + 0_2 = 01_2 \quad 1_2 + 1_2 = 10_2$$

Circuits for Computer Addition (7/9)

FULL-ADDER

Circuit

Input/Output Table

P	*Q*	*R*	*C*	*S*
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

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

Figure 2.5.2

Circuits for Computer Addition (8/9)

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*.

Circuits for Computer Addition (9/9)

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.

A Parallel Adder to Add *PQR* and *STU* to Obtain *WXYZ*

Figure 2.5.3

Two’s Complements and the Computer Representation of Signed Integers

Two’s Complements and the Computer Representation of Signed Integers (1/9)

Typically a fixed number of bits is used to represent integers on a computer. One way to do this is to select a particular bit, normally the left-most, to indicate the sign of the integer, and to use the remaining bits for its absolute value 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 is to use “two’s complements,” which makes it possible to add integers quite easily and results in a unique representation for 0.

Two's Complements and the Computer Representation of Signed Integers (2/9)

Bit lengths of 64 and (sometimes) 32 are most often used in practice, but, for simplicity and because the principles are the same for all bit lengths, this discussion will focus on a bit length of 8.

Definition

The 8-bit two's complement for an integer a between -128 and 127 is the 8-bit binary representation for $\begin{cases} a & \text{if } a \geq 0 \\ 2^8 - |a| & \text{if } a < 0. \end{cases}$

Two's Complements and the Computer Representation of Signed Integers (3/9)

Thus the 8-bit representation for a nonnegative integer is the same as its 8-bit binary representation. As a concrete example for the negative integer −46, observe that

$$(2^8 - |-46|)_{10} = (256 - 46)_{10} = 210_{10} = (128 + 64 + 16 + 2)_{10} = 11010010_2,.$$

and so the 8-bit two's complement for −46 is 11010010.

Two's Complements and the Computer Representation of Signed Integers (4/9)

For negative integers, however, there is a more convenient way to compute two's complements, which involves less arithmetic than applying the definition directly.

The 8-Bit Two's Complement for a Negative Integer

The 8-bit two's complement for a negative integer a that is at least -128 can be obtained as follows:

- Write the 8-bit binary representation for $|a|$.
- Switch all the 1's to 0's and all the 0's to 1's. (This is called flipping, or complementing, the bits.)
- Add 1 in binary notation.

Example 2.5.6 – *Finding a Two's Complement*

Use the method described above to find the 8-bit two's complement for −46.

Example 2.5.6 – *Solution*

Write the 8-bit binary representation for $|-46|\ (=46)$, switch all the 1's to 0's and all the 0's to 1's, and then add 1.

$$|-46|_{10} = 46_{10} = (32+8+4+2)_{10}$$

$$= 00101110_2 \xrightarrow{\text{flip the bits}} 11010001 \xrightarrow{\text{add 1}} 11010010.$$

Note that this is the same result as was obtained directly from the definition.

Two's Complements and the Computer Representation of Signed Integers (5/9)

The fact that the method for finding 8-bit two's complements works in general depends on the following facts:

1. The binary representation of $2^8 - 1$ is 11111111_2.
2. Subtracting an 8-bit binary number a from 11111111_2 switches all the 1's to 0's and all the 0's to 1's.
3. $2^8 - |a| = [(2^8 - 1) - |a|] + 1$ for any number a.

Here is how the facts are used when $a = -46$:

	1	1	1	1	1	1	1	1	$\leftrightarrow 2^8 - 1$
	0	0	1	0	1	1	1	0	$\leftrightarrow \lvert -46 \rvert$
0's and 1's are switched ↗ ↘									
	1	1	0	1	0	0	0	1	$\leftrightarrow (2^8 - 1) - \lvert -46 \rvert$
1 is added +	0	0	0	0	0	0	0	1	$\leftrightarrow +1$
	1	1	0	1	0	0	1	0	$\leftrightarrow 2^8 - \lvert -46 \rvert$

Two's Complements and the Computer Representation of Signed Integers (7/9)

Because 127 is the largest integer represented in the 8-bit two's complement system and because $127_{10} = 01111111_2$, all the 8-bit two's complements for nonnegative integers have a leading bit of 0.

Moreover, because the bits are switched, the leading bit for all the negative integers is 1.

Two's Complements and the Computer Representation of Signed Integers (8/9)

Table 2.5.2 illustrates the 8-bit two's complement representations for the integers from −128 through 127.

Integer	8-Bit Two's Complement	Decimal Form of Two's Complement for Negative Integers
127	01111111	
126	01111110	
⋮	⋮	
2	00000010	
1	00000001	
0	00000000	
−1	11111111	$2^8 - 1$
−2	11111110	$2^8 - 2$
−3	11111101	$2^8 - 3$
⋮	⋮	⋮
−127	10000001	$2^8 - 127$
−128	10000000	$2^8 - 128$

Table 2.5.2

Two's Complements and the Computer Representation of Signed Integers (9/9)

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

- Apply the two's complement procedure to 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?

Example 2.5.7 – *Solution*

Since the left-most digit is 1, the integer is negative. Applying the two's complement procedure gives the following result:

$$1010\,1001 \xrightarrow{\text{flip the bits}} 0101\,0110 \xrightarrow{\text{add 1}} 0101\,0111_2$$

$$= (64 + 16 + 4 + 2 + 1)_{10} = 87_{10} = |-87|_{10}.$$

So the answer is −87. You can check its correctness by deriving the two's complement of −87 directly from the definition:

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

Addition and Subtraction with Integers in Two’s Complement Form

Addition and Subtraction with Integers in Two's Complement Form (1/4)

The main advantage of a two's complement representation for integers is that the same computer circuits used to add nonnegative integers in binary notation can be used for both additions and subtractions of integers in a two's complement system of numeration.

First note that because of the algebraic identity

$$a - b = a + (-b) \text{ for all real numbers,}$$

any subtraction problem can be changed into an addition one.

Addition and Subtraction with Integers in Two's Complement Form (2/4)

For example, suppose you want to compute 78 − 46. This equals 78 + (−46), which should give an answer of 32.

To see what happens when you add the numbers in their two's complement forms, observe that the 8-bit two's complement for 78 is the same as the ordinary binary representation for 78, which is 01001110 because 78 = 64 + 8 + 4 + 2, and, as previously shown, the 8-bit two's complement for −46 is 11010010.

Addition and Subtraction with Integers in Two's Complement Form (3/4)

Adding the numbers using binary addition gives the following:

		0	1	0	0	1	1	1	0	$\leftrightarrow 78$
	+	1	1	0	1	0	0	1	0	$\leftrightarrow -46$
	1	0	0	1	0	0	0	0	0	$\leftrightarrow 32?$

The result has a carry bit of 1 in the ninth, or 2^8th, position, but if you discard it, you obtain 00100000, which *is* the correct answer in 8-bit two's complement form because, since $32 = 2^8$,

$$32_{10} = 00100000_2.$$

Addition and Subtraction with Integers in Two's Complement Form (4/4)

General Procedure for Using 8-Bit Two's Complements to Add Two Integers

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 two's complement representations.
- Add the resulting integers using ordinary binary addition, discarding any carry bit of 1 that may occur in the 2^8th position.
- Convert the result back to decimal form.

Hexadecimal Notation

Hexadecimal Notation (1/5)

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 (2/5)

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.

Hexadecimal Notation (3/5)

The 16 hexadecimal digits are shown in Table 2.5.3, together with their decimal equivalents and, for future reference, their 4-bit binary equivalents.

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

Table 2.5.3

Example 2.5.8 – *Converting from Hexadecimal to Decimal Notation*

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

Example 2.5.8 – *Solution*

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

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

Hexadecimal Notation (4/5)

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 2.5.9 – *Converting from Hexadecimal to Binary Notation*

Convert $B09F_{16}$ to binary notation.

Example 2.5.9 – *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,

$$\begin{array}{cccc} \mathrm{B} & 0 & 9 & \mathrm{F} \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ 1011 & 0000 & 1001 & 1111 \end{array}$$

and the answer is 1011000010011111_2.

Hexadecimal Notation (5/5)

To convert integers written in binary notation into hexadecimal notation, reverse the steps of the previous procedure. Note that the commonly used computer representation for integers uses 32 bits. When these numbers are written in hexadecimal notation only eight characters are needed.

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 2.5.10 – *Converting from Binary to Hexadecimal Notation*

Convert 100110110101001_2 to hexadecimal notation.

Example 2.5.10 – *Solution (1/2)*

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

0100 1101 1010 1001.

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

0100	1101	1010	1001
↕	↕	↕	↕
4	D	A	9

Example 2.5.10 – *Solution (2/2)* continued

Then juxtapose the hexadecimal digits.

$$4DA9_{16}$$

Example 2.5.11 – *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 BB 59 2E.

What is the actual content of the four memory locations?

Example 2.5.11 – *Solution*

$A3_{16} = 10100011_2$

$BB_{16} = 10111011_2$

$59_{16} = 01011001_2$

$2E_{16} = 00101110_2$

CHAPTER 3

THE LOGIC OF QUANTIFIED STATEMENTS

Copyright © Cengage Learning. All rights reserved.

3.1 Predicates and Quantified Statements I

Copyright © Cengage Learning. All rights reserved.

Predicates and Quantified Statements I(1/2)

Definition

A **predicate** is a sentence that contains a finite number of variables and becomes a statement when specific values are substituted for the variables. The **domain** of a predicate variable is the set of all values that may be substituted in place of the variable.

Example 3.1.1 – *Finding Truth Values of a Predicate*

Let $P(x)$ be the predicate "$x^2 > x$" with domain the set **R** of all real numbers. Write $P(2)$, $P\left(\frac{1}{2}\right)$, and $P\left(-\frac{1}{2}\right)$, and indicate which of these statements are true and which are false.

Example 3.1.1 – *Solution*

$$P(2): \quad 2^2 > 2, \quad \text{or} \quad 4 > 2. \quad \text{True.}$$

$$P\left(\frac{1}{2}\right): \quad \left(\frac{1}{2}\right)^2 > \frac{1}{2}, \quad \text{or} \quad \frac{1}{4} > \frac{1}{2}. \quad \text{False.}$$

$$P\left(-\frac{1}{2}\right): \quad \left(-\frac{1}{2}\right)^2 > -\frac{1}{2} \quad \text{or} \quad \frac{1}{4} > -\frac{1}{2}. \quad \text{True.}$$

Predicates and Quantified Statements I(2/2)

Definition

If $P(x)$ is a predicate and x has domain D, the **truth set** of $P(x)$ is the set of all elements of D that make $P(x)$ true when they are substituted for x. The truth set of $P(x)$ is denoted

$$\{x \in D \mid P(x)\}.$$

Example 3.1.2 – *Finding the Truth Set of a Predicate*

Let $Q(n)$ be the predicate "n is a factor of 8." Find the truth set of $Q(n)$ if

a. the domain of n is $\mathbf{Z}^+$, the set of all positive integers

b. the domain of n is $\mathbf{Z}$, the set of all integers.

Example 3.1.2 – *Solution*

a. The truth set is {1, 2, 4, 8} because these are exactly the positive integers that divide 8 evenly.

b. The truth set is {1, 2, 4, 8, −1, −2, −4, −8} because the negative integers −1, −2, −4, and −8 also divide into 8 without leaving a remainder.

The Universal Quantifier: ∀

The Universal Quantifier: ∀ (1/3)

One sure way to change predicates into statements is to assign specific values to all their variables.

For example, if *x* represents the number 35, the sentence "*x* is (evenly) divisible by 5" is a true statement since $35 = 5 \cdot 7$. Another way to obtain statements from predicates is to add **quantifiers**.

Quantifiers are words that refer to quantities such as "some" or "all" and tell for how many elements a given predicate is true.

The Universal Quantifier: ∀ (2/3)

The symbol ∀ is called the **universal quantifier**. Depending on the context, it is read as “for every,” “for each,” “for any,” “given any,” or “for all.”

For example, another way to express the sentence “Every human being is mortal” or “All human beings are mortal” is to write

∀ human beings x, x is mortal,

which you would read as “For every human being x, x is mortal.”

The Universal Quantifier: ∀ (3/3)

If you let H be the set of all human beings, then you can symbolize the statement more formally by writing

$$\forall x \in H,\ x \text{ is mortal.}$$

Definition

Let $Q(x)$ be a predicate and D the domain of x. A **universal statement** is a statement of the form "$\forall x \in D,\ Q(x)$." It is defined to be true if, and only if, $Q(x)$ is true for each individual x in D. It is defined to be false if, and only if, $Q(x)$ is false for at least one x in D. A value for x for which $Q(x)$ is false is called a **counterexample** to the universal statement.

Example 3.1.3 – *Truth and Falsity of Universal Statements*

a. Let $D = \{1, 2, 3, 4, 5\}$, and consider the statement

$$\forall x \in D, x^2 \geq x.$$

Write one way to read this statement out loud, and show that it is true.

b. Consider the statement

$$\forall x \in \mathbf{R}, x^2 \geq x.$$

Find a counterexample to show that this statement is false.

Example 3.1.3 – *Solution (1/1)*

a. "For every *x* in the set *D*, x^2 is greater than or equal to *x*." The inequalities below show th "$x^2 \geq x$" is true for each individual *x* in *D*.

$$1^2 \geq 1, \qquad 2^2 \geq 2, \qquad 3^2 \geq 3, \qquad 4^2 \geq 4, \qquad 5^2 \geq 5.$$

Hence "$\forall x \in D, x^2 \geq x$" is true.

Example 3.1.3 – *Solution (1/2)*

continued

b. *Counterexample:* The statement claims that $x^2 \geq x$ for every real number *x*. But when $x = \frac{1}{2}$, for example,

$$\left(\frac{1}{2}\right)^2 = \frac{1}{4} \ngeq \frac{1}{2}.$$

Hence "$\forall x \in \mathbf{R}, x^2 \geq x$" is false.

The Existential Quantifier: ∃

The Existential Quantifier: ∃ (1/2)

The symbol ∃ denotes “there exists” and is called the **existential quantifier**. For example, the sentence “There is a student in Math 140” can be written as

∃ a person *p* such that *p* is a student in Math 140,

or, more formally,

$\exists\, p \in P$ such that *p* is a student in Math 140,

where *P* is the set of all people.

The Existential Quantifier: ∃ (2/2)

The domain of the predicate variable is generally indicated either between the ∃ symbol and the variable name or immediately following the variable name, and the words *such that* are inserted just before the predicate.

Definition

Let $Q(x)$ be a predicate and D the domain of x. An **existential statement** is a statement of the form "$\exists x \in D$ such that $Q(x)$." It is defined to be true if, and only if, $Q(x)$ is true for at least one x in D. It is false if, and only if, $Q(x)$ is false for all x in D.

Example 3.1.4 – *Truth and Falsity of Existential Statements*

a. Consider the statement

$$\exists m \in \mathbf{Z}^+ \text{ such that } m^2 = m.$$

Write one way to read this statement out loud, and show that it is true.

b. Let $E = \{5, 6, 7, 8\}$ and consider the statement

$$\exists m \in E \text{ such that } m^2 = m.$$

Show that this statement is false.

Example 3.1.4 – *Solution*

a. "There is at least one positive integer *m* such that $m^2 = m$." Observe that $1^2 = 1$. Thus "$m^2 = m$" is true for a positive integer *m*, and so "$\exists m \in \mathbf{Z}^+$ such that $m^2 = m$" is true.

b. Note that $m^2 = m$ is not true for any integers *m* from 5 through 8:

$$5^2 = 25 \neq 5, \quad 6^2 = 36 \neq 6, \quad 7^2 = 49 \neq 7, \quad 8^2 = 64 \neq 8.$$

Thus "$\exists m \in E$ such that $m^2 = m$" is false.

Formal vs. Informal Language

Formal vs. Informal Language (1/1)

It is important to be able to translate from formal to informal language when trying to make sense of mathematical concepts that are new to you. It is equally important to be able to translate from informal to formal language when thinking out a complicated problem.

Example 3.1.5 – *Translating from Formal to Informal Language*

Rewrite the following formal statements in a variety of equivalent but more informal ways. Do not use the symbol ∀ or ∃.

a. $\forall x \in \mathbf{R}, x^2 \geq 0.$

b. $\forall x \in \mathbf{R}, x^2 \neq -1.$

c. $\exists m \in \mathbf{Z}^+$ such that $m^2 = m.$

Example 3.1.5 – *Solution (1/2)*

a. Every real number has a nonnegative square.

Or: All real numbers have nonnegative squares.

Or: Any real number has a nonnegative square.

Or: The square of each real number is nonnegative.

b. All real numbers have squares that do not equal −1.

Or: No real numbers have squares equal to −1.

(The words *none are* or *no* … *are* are equivalent to the words *all are not*.)

Example 3.1.5 – *Solution (2/2)* continued

c. There is a positive integer whose square is equal to itself.

Or: We can find at least one positive integer equal to its own square.

Or: Some positive integer equals its own square.

Or: Some positive integers equal their own squares.

Example 3.1.6 – *Trailing Quantifiers*

Rewrite the following statements so that the quantifier trails the rest of the sentence.

a. For any integer n, $2n$ is even.

b. There exists at least one real number x such that $x^2 \leq 0$.

Example 3.1.6 – *Solution*

a. 2*n* is even for any integer *n*.

b. $x^2 \leq 0$ for some real number *x*.

Or: $x^2 \leq 0$ for at least one real number *x*.

Example 3.1.7 – *Translating from Informal to Formal Language*

Rewrite each of the following statements formally. Use quantifiers and variables.

a. All triangles have three sides.

b. No dogs have wings.

c. Some programs are structured.

Example 3.1.7 – *Solution*

a. $\forall$ triangle t, t has three sides.

Or: $\forall t \in T$, t has three sides (where T is the set of all triangles).

b. $\forall$ dog d, d does not have wings.

Or: $\forall d \in D$, d does not have wings (where D is the set of all dogs).

c. $\exists$ a program p such that p is structured.

Or: $\exists p \in P$ such that p is structured (where P is the set of all programs).

Universal Conditional Statements

Universal Conditional Statements (1/1)

A reasonable argument can be made that the most important form of statement in mathematics is the **universal conditional statement**:

$$\forall x, \text{ if } P(x) \text{ then } Q(x).$$

Example 3.1.8 – *Writing Universal Conditional Statements Informally*

Rewrite the following statement informally, without quantifiers or variables.

$$\forall x \in \mathbf{R}, \text{ if } x > 2 \text{ then } x^2 > 4.$$

Example 3.1.8 – *Solution*

If a real number is greater than 2, then its square is greater than 4.

Or: Whenever a real number is greater than 2, its square is greater than 4.

Or: The square of any real number greater than 2 is greater than 4.

Or: The squares of all real numbers greater than 2 are greater than 4.

Example 3.1.9 – *Writing Universal Conditional Statements Formally*

Rewrite each of the following statements in the form

$$\forall _____, \text{ if } _____ \text{ then } _____.$$

a. If a real number is an integer, then it is a rational number.

b. All bytes have eight bits.

c. No fire trucks are green.

Example 3.1.9 – *Solution*

a. $\forall$ real number x, if x is an integer, then x is a rational number.

Or: $\forall x \in \mathbf{R}$, if $x \in \mathbf{Z}$ then $x \in \mathbf{Q}$.

b. $\forall x$, if x is a byte, then x has eight bits.

c. $\forall x$, if x is a fire truck, then x is not green.

It is common, as in (b) and (c) above, to omit explicit identification of the domain of predicate variables in universal conditional statements.

Equivalent Forms of Universal and Existential Statements

Equivalent Forms of Universal and Existential Statements (1/2)

Observe that the two statements "∀ real number x, if x is an integer then x is rational" and "∀ integer x, x is rational" mean the same thing because the set of integers is a subset of the set of real numbers.

Both have informal translations "All integers are rational." In fact, a statement of the form $\forall x \in U$, if $P(x)$ then $Q(x)$

can always be rewritten in the form $\forall x \in D$, $Q(x)$

by narrowing U to be the subset D consisting of all values of the variable x that make $P(x)$ true.

Equivalent Forms of Universal and Existential Statements (2/2)

Conversely, a statement of the form

$$\forall x \in D, Q(x)$$

can be rewritten as

$\forall x$, if x is in D then $Q(x)$.

Example 3.1.10 – *Equivalent Forms for Universal Statements*

Rewrite the following statement in the two forms

"$\forall x$, if _____ then _____" and "$\forall$ _____ x, _____":

All squares are rectangles.

Example 3.1.10 – *Solution*

$\forall x$, if x is a square then x is a rectangle.

$\forall$ square x, x is a rectangle.

Example 3.1.11 – *Equivalent Forms for Existential Statements*

A **prime number** is an integer greater than 1 whose only positive integer factors are itself and 1.

Consider the statement "There is an integer that is both prime and even."

Let Prime(n) be "n is prime" and Even(n) be "n is even." Use the notation Prime(n) and Even(n) to rewrite this statement in the following two forms:

a. $\exists n$ such that _____ $\wedge$ _____.

b. $\exists$ _____ n such that _____.

Example 3.1.11 – *Solution*

a. $\exists n$ such that Prime(n) $\wedge$ Even(n).

b. Two answers: $\exists$ a prime number n such that Even(n).

$\exists$ an even number n such that Prime(n).

Bound Variables and Scope

Bound Variables and Scope (1/2)

For example, consider the following statements:

(1) For every integer *x*, $x^2 \geq 0.$

(2) There exists a real number *x* such that $x^3 = 8.$

Statements (1) and (2) both call the variable *x*, but the *x* in Statement (1) serves a different function from the *x* in Statement (2).

Bound Variables and Scope (2/2)

We say that the variable *x* is **bound** by the quantifier that controls it and that its **scope** begins when the quantifier introduces it and ends at the end of the quantified statement.

Implicit Quantification

Implicit Quantification (1/2)

Consider the statement

If a number is an integer, then it is a rational number.

As shown earlier, this statement is equivalent to a universal statement. However, it does not contain the telltale word *all* or *every* or *any* or *each*.

The only clue to indicate its universal quantification comes from the presence of the indefinite article *a*. This is an example of *implicit* universal quantification.

Implicit Quantification (2/2)

Existential quantification can also be implicit. For instance, the statement "The number 24 can be written as a sum of two even integers" can be expressed formally as "∃ even integers m and n such that $24 = m + n$."

Notation

Let $P(x)$ and $Q(x)$ be predicates and suppose the common domain of x is D.

- The notation $\boldsymbol{P(x) \Rightarrow Q(x)}$ means that every element in the truth set of $P(x)$ is in the truth set of $Q(x)$, or, equivalently, $\forall x, P(x) \rightarrow Q(x)$.
- The notation $\boldsymbol{P(x) \Leftrightarrow Q(x)}$ means that $P(x)$ and $Q(x)$ have identical truth sets, or, equivalently, $\forall x, P(x) \leftrightarrow Q(x)$.

Example 3.1.12 – *Using* $\Rightarrow$ *and* $\Leftrightarrow$

Let

$Q(n)$ be "n is a factor of 8,"
$R(n)$ be "n is a factor of 4,"
$S(n)$ be "$n < 5$ and $n \neq 3$,"

and suppose the domain of n is $\mathbf{Z}^+$, the set of positive integers.

Use the $\Rightarrow$ and $\Leftrightarrow$ symbols to indicate true relationships among $Q(n)$, $R(n)$, and $S(n)$.

Example 3.1.12 – *Solution (1/2)*

1. As noted in Example 3.1.2, the truth set of $Q(n)$ is $\{1, 2, 4, 8\}$ when the domain of n is $\mathbf{Z}^+$. By similar reasoning the truth set of $R(n)$ is $\{1, 2, 4\}$.

   Thus it is true that every element in the truth set of $R(n)$ is in the truth set of $Q(n)$, or, equivalently, $\forall n$ in $\mathbf{Z}^+$, $R(n) \rightarrow Q(n)$. So $R(n) \Rightarrow Q(n)$, or, equivalently

   n is a factor of 4 $\Rightarrow$ n is a factor of 8.

Example 3.1.12 – *Solution (2/2)* continued

2. The truth set of $S(n)$ is $\{1, 2, 4\}$, which is identical to the truth set of $R(n)$, or, equivalently, $\forall n$ in $\mathbf{Z}^+$, $R(n) \leftrightarrow S(n)$.

   So $R(n) \Leftrightarrow S(n)$, or, equivalently,

   $$n \text{ is a factor of } 4 \quad \Leftrightarrow \quad n < 5 \text{ and } n \neq 3.$$

   Moreover, since every element in the truth set of $S(n)$ is in the truth set of $Q(n)$, or, equivalently, $\forall n$ in $\mathbf{Z}^+$,

   $S(n) \rightarrow Q(n)$, then $S(n) \Rightarrow Q(n)$, or, equivalently,

   $$n < 5 \text{ and } n \neq 3 \quad \Rightarrow \quad n \text{ is a factor of } 8.$$

Tarski’s World

Tarski’s World (1/1)

Tarski’s World is a computer program developed by information scientists Jon Barwise and John Etchemendy to help teach the principles of logic.

Example 3.1.13 – *Investigating Tarski's World (1/3)*

The program for Tarski's World provides pictures of blocks of various sizes, shapes, and colors, which are located on a grid.

Shown in Figure 3.1.1 is a picture of an arrangement of objects in a two-dimensional Tarski world.

Figure 3.1.1

Example 3.1.13 – *Investigating Tarski's World (2/3)*

The configuration can be described using logical operators and—for the two-dimensional version—notation such as Triangle(x), meaning "x is a triangle," Blue(y), meaning "y is blue," and RightOf(x, y), meaning "x is to the right of y (but possibly in a different row)."

Individual objects can be given names such as a, b, or c.

Determine the truth or falsity of each of the following statements. The domain for all variables is the set of objects in the Tarski world shown in Figure 3.1.1.

Example 3.1.13 – *Investigating Tarski's World (3/3)*

a. $\forall t$, Triangle(t) $\rightarrow$ Blue(t).

b. $\forall x$, Blue(x) $\rightarrow$ Triangle(x).

c. $\exists y$ such that Square(y) $\wedge$ RightOf(d, y).

d. $\exists z$ such that Square(z) $\wedge$ Gray(z).

Example 3.1.13 – *Solution*

a. This statement is true: Every triangle is blue.

b. This statement is false. As a counterexample, note that *e* is blue and it is not a triangle.

c. This statement is true because *e* and *h* are both square and *d* is to their right.

d. This statement is false: All the squares are either blue or black.

CHAPTER 3

THE LOGIC OF QUANTIFIED STATEMENTS

Copyright © Cengage Learning. All rights reserved.

3.2 Predicates and Quantified Statements II

Copyright © Cengage Learning. All rights reserved.

Negations of Quantified Statements

Negations of Quantified Statements (1/5)

The general form of the negation of a universal statement follows immediately from the definitions of negation and of the truth values for universal and existential statements.

Theorem 3.2.1 Negation of a Universal Statement

The negation of a statement of the form

$$\forall x \text{ in } D,\ Q(x)$$

is logically equivalent to a statement of the form

$$\exists x \text{ in } D \text{ such that } \sim Q(x).$$

Symbolically,

$$\sim(\forall x \in D,\ Q(x)) \equiv \exists x \in D \text{ such that } \sim Q(x).$$

Negations of Quantified Statements (2/5)

Thus

> **The negation of a universal statement ("all are") is logically equivalent to an existential statement ("some are not" or "there is at least one that is not").**

Note that when we speak of **logical equivalence for quantified statements**, we mean that the statements always have identical truth values no matter what predicates are substituted for the predicate symbols and no matter what sets are used for the domains of the predicate variables.

Negations of Quantified Statements (3/5)

Theorem 3.2.2 Negation of an Existential Statement

The negation of a statement of the form

$$\exists x \text{ in } D \text{ such that } Q(x)$$

is logically equivalent to a statement of the form

$$\forall x \text{ in } D, \sim Q(x).$$

Symbolically,

$$\sim(\exists x \in D \text{ such that } Q(x)) \equiv \forall x \in D, \sim Q(x).$$

Negations of Quantified Statements (4/5)

Thus

The negation of an existential statement (“some are”) is logically equivalent to a universal statement (“none are” or “all are not”).

Example 3.2.1 – *Negating Quantified Statements*

Write formal negations for the following statements:

a. ∀ primes p, p is odd.

b. ∃ a triangle T such that the sum of the angles of T equals 200°.

Example 3.2.1 – *Solution*

a. By applying the rule for the negation of a ∀ statement, you can see that the answer is

∃ a prime p such that p is not odd.

b. By applying the rule for the negation of a ∃ statement, you can see that the answer is

∀ triangles T, the sum of the angles of T does not equal 200°.

Negations of Quantified Statements (5/5)

You need to exercise special care to avoid mistakes when writing negations of statements that are given informally. One way to avoid error is to rewrite the statement formally and take the negation using the formal rule.

Example 3.2.2 – *More Negations*

Rewrite the following statements formally. Then write formal and informal negations.

a. No politicians are honest.

b. The number 1,357 is not divisible by any integer between 1 and 37.

Example 3.2.2 – *Solution (1/2)*

a. *Formal version:* ∀ politicians x, x is not honest.

Formal negation: ∃ a politician x such that x is honest.

Informal negation: Some politicians are honest.

b. This statement has a trailing quantifier. Written formally it becomes:

∀ integer n between 1 and 37, 1,357 is not divisible by n.

Its negation is therefore

∃ an integer n between 1 and 37 such that 1,357 is divisible by n.

Example 3.2.2 – *Solution (2/2)*

An informal version of the negation is

> The number 1,357 is divisible by some integer between 1 and 37.

Negations of Universal Conditional Statements

Negations of Universal Conditional Statements (1/2)

Negations of universal conditional statements are of special importance in mathematics. The form of such negations can be derived from facts that have already been established.

By definition of the negation of a *for all* statement,

$$\sim(\forall x, P(x) \rightarrow Q(x)) \equiv \exists x \text{ such that } \sim(P(x) \rightarrow Q(x)). \qquad 3.2.1$$

But the negation of an if-then statement is logically equivalent to an *and* statement. More precisely,

$$\sim(P(x) \rightarrow Q(x)) \equiv P(x) \wedge \sim Q(x). \qquad 3.2.2$$

Negations of Universal Conditional Statements (2/2)

Substituting (3.2.2) into (3.2.1) gives

$$\sim(\forall x, P(x) \rightarrow Q(x)) \equiv \exists x \text{ such that } (P(x) \wedge \sim Q(x)).$$

Written somewhat less symbolically, this becomes

Negation of a Universal Conditional Statement

$$\sim(\forall x, \text{ if } P(x) \text{ then } Q(x)) \equiv \exists x \text{ such that } P(x) \text{ and } \sim Q(x).$$

Example 3.2.4 – *Negating Universal Conditional Statements*

Write a formal negation for statement (a) and an informal negation for statement (b).

a. ∀ person *p*, if *p* is blond then *p* has blue eyes.

b. If a computer program has more than 100,000 lines, then it contains a bug.

Example 3.2.4 – *Solution*

a. ∃ a person *p* such that *p* is blond and *p* does not have blue eyes.

b. There is at least one computer program that has more than 100,000 lines and does not contain a bug.

The Relation among $\forall$, $\exists$, $\wedge$, and $\vee$

The Relation among ∀, ∃, ∧ , and ∨ (1/4)

The negation of a *for all* statement is a *there exists* statement, and the negation of a *there exists* statement is a *for all* statement.

These facts are analogous to De Morgan's laws, which state that the negation of an *and* statement is an *or* statement and that the negation of an *or* statement is an *and* statement. This similarity is not accidental.

In a sense, universal statements are generalizations of *and* statements, and existential ments are generalizations of *or* statements.

The Relation among ∀, ∃, ∧ , and ∨ (2/4)

If $Q(x)$ is a predicate and the domain D of x is the set $\{x_1, x_2, \ldots, x_n\}$, then the statements

$$\forall x \in D, Q(x) \quad \text{and} \quad Q(x_1) \wedge Q(x_2) \wedge \cdots \wedge Q(x_n)$$

are logically equivalent. For example, let $Q(x)$ be "$x \cdot x = x$" and suppose $D = \{0, 1\}$. Then

$$\forall x \in D, Q(x)$$

can be rewritten as

$$\forall \text{ binary digits } x, x \cdot x = x.$$

The Relation among ∀, ∃, ∧ , and ∨ (3/4)

This is equivalent to

$$0 \cdot 0 = 0 \text{ and } 1 \cdot 1 = 1,$$

which can be rewritten in symbols as

$$Q(0) \wedge Q(1).$$

Similarly, if $Q(x)$ is a predicate and $D = \{x_1, x_2, \ldots, x_n\}$,

then the statements

$$\exists x \in D \text{ such that } Q(x) \quad \text{and} \quad Q(x_1) \vee Q(x_2) \vee \cdots \vee Q(x_n)$$

are logically equivalent. For example, let $Q(x)$ be "$x + x = x$" and suppose $D = \{0, 1\}$.

The Relation among ∀, ∃, ∧, and ∨ (4/4)

Then

$$\exists x \in D \text{ such that } Q(x)$$

can be rewritten as

$$\exists \text{ a binary digit } x \text{ such that } x + x = x.$$

This is equivalent to

$$0 + 0 = 0 \text{ or } 1 + 1 = 1,$$

which can be rewritten in symbols as

$$Q(0) \vee Q(1).$$

Vacuous Truth of Universal Statements

Vacuous Truth of Universal Statements (1/4)

Suppose a bowl sits on a table and next to the bowl is a pile of five blue and five gray balls, any of which may be placed in the bowl. If three blue balls and one gray ball are placed in the bowl, as shown in Figure 3.2.1(a), the statement “All the balls in the bowl are blue” would be false (since one of the balls in the bowl is gray).

Figure 3.2.1(a)

Vacuous Truth of Universal Statements (2/4)

Now suppose that no balls at all are placed in the bowl, as shown in Figure 3.2.1(b).

Figure 3.2.1(b)

Vacuous Truth of Universal Statements (3/4)

Consider the statement

All the balls in the bowl are blue.

Is this statement true or false? The statement is false if, and only if, its negation is true. And its negation is

There exists a ball in the bowl that is not blue.

But the only way this negation can be true is for there actually to be a nonblue ball in the bowl. And there is not! Hence the negation is false, and so the statement is true "by default."

Vacuous Truth of Universal Statements (4/4)

In general, a statement of the form

$$\forall x \text{ in } D, \text{ if } P(x) \text{ then } Q(x)$$

is called **vacuously true** or **true by default** if, and only if, $P(x)$ is false for every x in D.

Variants of Universal Conditional Statements

Variants of Universal Conditional Statements (1/1)

A conditional statement has a contrapositive, a converse, and an inverse. The definitions of these terms can be extended to universal conditional statements.

Definition

Consider a statement of the form $\forall x \in D$, if $P(x)$ then $Q(x)$.

1. Its **contrapositive** is the statement $\forall x \in D$, if $\sim Q(x)$ then $\sim P(x)$.
2. Its **converse** is the statement $\forall x \in D$, if $Q(x)$ then $P(x)$.
3. Its **inverse** is the statement $\forall x \in D$, if $\sim P(x)$ then $\sim Q(x)$.

Example 3.2.5 – *Contrapositive, Converse, and Inverse of a Universal Conditional Statement*

Write a formal and an informal contrapositive, converse, and inverse for the following statement:

> If a real number is greater than 2, then its square is greater than 4.

Example 3.2.5 – *Solution (1/2)*

The formal version of this statement is $\forall x \in \mathbf{R}$, if $x > 2$ then $x^2 > 4$.

Contrapositive: $\forall x \in \mathbf{R}$, if $x^2 \leq 4$ then $x \leq 2$.

Or: If the square of a real number is less than or equal to 4, then the number is less than or equal to 2.

Converse: $\forall x \in \mathbf{R}$, if $x^2 > 4$ then $x > 2$.

Or: If the square of a real number is greater than 4, then the number is greater than 2.

Example 3.2.5 – *Solution (2/2)*

continued

Inverse: $\forall x \in \mathbf{R}$, if $x \leq 2$ then $x^2 \leq 4$.

Or: If a real number is less than or equal to 2, then the square of the number is less than or equal to 4.

Note that in solving this example, we have used the equivalence of "$x \not> a$" and "$x \leq a$" for all real numbers x and a.

Necessary and Sufficient Conditions, Only If

Necessary and Sufficient Conditions, Only If (1/1)

The definitions of *necessary*, *sufficient*, and *only if* can also be extended to apply to universal conditional statements.

Definition

- "$\forall x$, $r(x)$ is a **sufficient condition** for $s(x)$" means "$\forall x$, if $r(x)$ then $s(x)$."
- "$\forall x$, $r(x)$ is a **necessary condition** for $s(x)$" means "$\forall x$, if $\sim r(x)$ then $\sim s(x)$" or, equivalently, "$\forall x$, if $s(x)$ then $r(x)$."
- "$\forall x$, $r(x)$ **only if** $s(x)$" means "$\forall x$, if $\sim s(x)$ then $\sim r(x)$" or, equivalently, "$\forall x$, if $r(x)$ then $s(x)$."

Example 3.2.6 – *Necessary and Sufficient Conditions*

Rewrite each of the following as a universal conditional statement, quantified either explicitly or implicitly. Do not use the word *necessary* or *sufficient*.

a. Squareness is a sufficient condition for rectangularity.

b. Being at least 35 years old is a necessary condition for being president of the United States.

Example 3.2.6 – *Solution (1/2)*

a. A formal version of the statement is

$\forall x$, if x is a square, then x is a rectangle.

Or, with implicit universal quantification:

If a figure is a square, then it is a rectangle.

b. Using formal language, you could write the answer as

$\forall$ person x, if x is younger than 35, then x cannot be president of the United States.

Example 3.2.6 – *Solution (2/2)* continued

Or, by the equivalence between a statement and its contrapositive:

∀ person *x*, if *x* is president of the United States, then *x* is at least 35 years old.

Example 3.2.7 – *Only If*

Rewrite the following as a universal conditional statement:

A product of two numbers is 0 only if one of the numbers is 0.

Example 3.2.7 – *Solution (1/2)*

Using informal language, you could write the answer as

> If it is not the case that one of two numbers is 0, then the product of the numbers is not 0.

In other words,

> If neither of two numbers is 0, then the product of the numbers is not 0.

Example 3.2.7 – *Solution (2/2)*

continued

Or, by the equivalence between a statement and its contrapositive:

> If a product of two numbers is 0, then one of the numbers is 0.

CHAPTER 3

THE LOGIC OF QUANTIFIED STATEMENTS

Copyright © Cengage Learning. All rights reserved.

3.3 Statements with Multiple Quantifiers

Copyright © Cengage Learning. All rights reserved.

Statements with Multiple Quantifiers (1/4)

***When a statement contains more than one kind of quantifier, we imagine the actions suggested by the quantifiers as being performed in the order in which the quantifiers occur*.** For instance, consider a statement of the form

$\forall x$ in set D, $\exists y$ in set E such that x and y satisfy property $P(x, y)$.

Statements with Multiple Quantifiers (2/4)

To show that such a statement is true, you must be able to meet the following challenge:

- Imagine that someone is allowed to choose any element whatsoever from the set D, and imagine that the person gives you that element. Call it x.

- The challenge for you is to find an element y in E so that the person's x and your y, taken together, satisfy property $P(x, y)$.

Statements with Multiple Quantifiers (3/4)

Because you do not have to specify the y until after the other person has specified the x, you are allowed to find a different value of y for each different x you are given.

Example 3.3.1 – *Truth of a ∀∃ Statement in a Tarski World*

Consider the Tarski world shown in Figure 3.3.1.

Show that the following statement is true in this world:

> For every triangle *x*, there is a square *y* such that *x* and *y* have the same color.

Figure 3.3.1

Example 3.3.1 – *Solution*

The statement says that no matter which triangle someone gives you, you will be able to find a square of the same color. There are only three triangles, *d*, *f*, and *i*.

The following table shows that for each of these triangles a square of the same color can be found.

Given $x =$	choose $y =$	and check that y is the same color as x.
d	e	yes ✓
f or i	h or g	yes ✓

Example 3.3.2 – *Truth of a ∃∀ Statement in a Tarski World*

Consider the Tarski world shown in Figure 3.3.1.

Show that the following statement is true:

> There is a triangle *x* such that for every circle *y*, *x* is to the right of *y*.

Figure 3.3.1

Example 3.3.2 – *Solution*

The statement says that you can find a triangle that is to the right of all the circles.

Actually, either *d* or *i* would work for all of the three circles, *a*, *b*, and *c*, as you can see in the following table.

Choose x =	Then: given y =	check that x is to the right of y.
d or i	a	yes ✓
	b	yes ✓
	c	yes ✓

Statements with Multiple Quantifiers (4/4)

Interpreting Statements with Two Different Quantifiers

If you want to establish the truth of a statement of the form

$$\forall x \text{ in } D, \exists y \text{ in } E \text{ such that } P(x, y)$$

your challenge is to allow someone else to pick whatever element x in D they wish and then you must find an element y in E that "works" for that particular x.

If you want to establish the truth of a statement of the form

$$\exists x \text{ in } D \text{ such that } \forall y \text{ in } E, P(x, y)$$

your job is to find one particular x in D that will "work" no matter what y in E anyone might choose to challenge you with.

Example 3.3.3 – *Interpreting Statements with More Than One Quantifier (1/2)*

A college cafeteria line has four stations: salads, main courses, desserts, and beverages. The salad station offers a choice of green salad or fruit salad; the main course station offers spaghetti or fish; the dessert station offers pie or cake; and the beverage station offers milk, soda, or coffee. Three students, Uta, Tim, and Yuen, go through the line and make the following choices:

Uta: green salad, spaghetti, pie, milk

Tim: fruit salad, fish, pie, cake, milk, coffee

Yuen: spaghetti, fish, pie, soda

Example 3.3.3 – *Interpreting Statements with More Than One Quantifier (2/2)*

These choices are illustrated in Figure 3.3.2.

Figure 3.3.2

Example 3.3.3 – *Interpreting Statements with More Than One Quantifier* continued

Write each of following statements informally and find its truth value.

a. ∃ an item *I* such that ∀ student *S*, *S* chose *I*.

b. ∃ a student *S* such that ∀ item *I*, *S* chose *I*.

c. ∃ a student *S* such that ∀ station *Z*, ∃ an item *I* in *Z* such that *S* chose *I*.

d. ∀ student *S* and ∀ station *Z*, ∃ an item *I* in *Z* such that *S* chose *I*.

Example 3.3.3 – *Solution*

a. There is an item that was chosen by every student. This is true; every student chose pie.

b. There is a student who chose every available item. This is false; no student chose all nine items.

c. There is a student who chose at least one item from every station. This is true; both Uta and Tim chose at least one item from every station.

d. Every student chose at least one item from every station. This is false; Yuen did not choose a salad.

Translating from Informal to Formal Language

Translating from Informal to Formal Language (1/1)

Most problems are stated in informal language, but solving them often requires translating them into more formal terms.

Example 3.3.4 – *Translating Statements with Multiple Quantifiers from Informal to Formal Language*

The **reciprocal** of a real number a is a real number b such that $ab = 1$. The following two statements are true. Rewrite them formally using quantifiers and variables.

a. Every nonzero real number has a reciprocal.

b. There is a real number with no reciprocal.

Example 3.3.4 – *Solution*

a. $\forall$ nonzero real number u, $\exists$ a real number v such that $uv = 1$.

b. $\exists$ a real number c such that $\forall$ real number d, $cd \neq 1$.

Example 3.3.7 – *The Definition of Limit of a Sequence (1/3)*

The definition of limit of a sequence, studied in calculus, uses both quantifiers ∀ and ∃ and also if-then. We say that the limit of the sequence a_n as n goes to infinity equals L and write

$$\lim_{n \to \infty} a_n = L$$

if, and only if, the values of a_n become *arbitrarily* close to L as n gets larger and larger without bound.

Example 3.3.7 – *The Definition of Limit of a Sequence (2/3)*

More precisely, this means that given any positive number ε, we can find an integer N such that whenever n is larger than N, the number a_n sits between $L - \varepsilon$ and $L + \varepsilon$ on the number line.

Example 3.3.7 – *The Definition of Limit of a Sequence (3/3)*

Symbolically:

$\forall \varepsilon > 0$, $\exists$ an integer N such that $\forall$ integer n,
if $n > N$ then $L - \varepsilon < a_n < L + \varepsilon$.

Considering the logical complexity of this definition, it is no wonder that many students find it hard to understand.

Ambiguous Language

Ambiguous Language (1/1)

The drawing in Figure 3.3.3 is a famous example of visual ambiguity. When you look at it for a while, you will probably see either a silhouette of a young woman wearing a large hat or an elderly woman with a large nose.

Figure 3.3.3

Negations of Statements with More Than One Quantifier

Negations of Statements with More Than One Quantifier (1/2)

You can use the same rules to negate statements with several quantifiers that you used to negate simpler quantified statements. We know that

$$\sim(\forall x \text{ in } D, P(x)) \equiv \exists x \text{ in } D \text{ such that } \sim P(x).$$

and

$$\sim(\exists x \text{ in } D \text{ such that } P(x)) \equiv \forall x \text{ in } D, \sim P(x).$$

Thus

$$\begin{aligned}\sim(\forall x \text{ in } D, \exists y \text{ in } E \text{ such that } P(x, y)) &\equiv \exists x \text{ in } D \text{ such that } \sim(\exists y \text{ in } E \text{ such that } P(x, y)) \\ &\equiv \exists x \text{ in } D \text{ such that } \forall y \text{ in } E, \sim P(x, y)\end{aligned}$$

Negations of Statements with More Than One Quantifier (2/2)

Similarly,

$$\sim(\exists x \text{ in } D \text{ such that } \forall y \text{ in } E, P(x, y)) \equiv \forall x \text{ in } D, \sim(\forall y \text{ in } E, P(x, y))$$
$$\equiv \forall x \text{ in } D, \exists y \text{ in } E \text{ such that } \sim P(x, y)$$

These facts are summarized as follows:

Negations of Statements with Two Different Quantifiers

$$\sim(\forall x \text{ in } D, \exists y \text{ in } E \text{ such that } P(x, y)) \equiv \exists x \text{ in } D \text{ such that } \forall y \text{ in } E, \sim P(x, y)$$
$$\sim(\exists x \text{ in } D \text{ such that } \forall y \text{ in } E, P(x, y)) \equiv \forall x \text{ in } D, \exists y \text{ in } E \text{ such that } \sim P(x, y)$$

Example 3.3.8 – *Negating Statements in a Tarski World (1/2)*

Consider the Tarski world shown in Figure 3.3.1.

Figure 3.3.1

Example 3.3.8 – *Negating Statements in a Tarski World (2/2)*

Write a negation for each of the following statements, and determine which is true, the given statement or its negation.

a. For every square *x*, there is a circle *y* such that *x* and *y* have the same color.

b. There is a triangle *x* such that for every square *y*, *x* is to the right of *y*.

Example 3.3.8 – *Solution (1/2)*

a. *First version of negation:* ∃ a square *x* such that ~ (∃ a circle *y* such that *x* and *y* have the same color).

Final version of negation: ∃ a square *x* such that ∀ circle *y*, *x* and *y* do not have the same color.

The negation is true. Square *e* is black and no circle in this Tarski world is black, so there is a square that does not have the same color as any circle.

Example 3.3.8 – *Solution (2/2)* continued

b. *First version of negation:* ∀ triangle x, ~ (∀ square y, x is to the right of y).

Final version of negation: ∀ triangle x, ∃ a square y such that x is not to the right of y.

The negation is true because no matter what triangle is chosen, it is not to the right of square g or square j, which are the only squares in this Tarski world.

Order of Quantifiers

Order of Quantifiers (1/3)

Consider the following two statements:

∀ person *x*, ∃ a person *y* such that *x* loves *y*.

∃ a person *y* such that ∀ person *x*, *x* loves *y*.

Note that except for the order of the quantifiers, these statements are identical.

However, the first means that given any person, it is possible to find someone whom that person loves, whereas the second means that there is one amazing individual who is loved by all people.

Order of Quantifiers (2/3)

(Reread the statements carefully to verify these interpretations!)

The two sentences illustrate an extremely important property about statements with two different quantifiers.

> In a statement containing both $\forall$ and $\exists$, changing the order of the quantifiers can significantly change the meaning of the statement.

Interestingly, however, if one quantifier immediately follows another quantifier *of the same type*, then the order of the quantifiers does not affect the meaning.

Order of Quantifiers (3/3)

Consider the commutative property of addition of real numbers, for example:

$\forall$ real number x and $\forall$ real number y, $x + y = y + x$.

This means the same as

$\forall$ real number y and $\forall$ real number x, $x + y = y + x$.

Thus the property can be expressed a little less formally as

$\forall$ real numbers x and y, $x + y = y + x$.

Example 3.3.9 – *Quantifier Order in a Tarski World (1/2)*

Look again at the Tarski world of Figure 3.3.1.

Figure 3.3.1

Example 3.3.9 – *Quantifier Order in a Tarski World (2/2)*

Do the following two statements have the same truth value?

a. For every square *x* there is a triangle *y* such that *x* and *y* have different colors.

b. There exists a triangle *y* such that for every square *x*, *x* and *y* have different colors.

Example 3.3.9 – *Solution (1/2)*

Statement (a) says that if someone gives you one of the squares from the Tarski world, you can find a triangle that has a different color. This is true.

If someone gives you square *g* or *h* (which are gray), you can use triangle *d* (which is black); if someone gives you square *e* (which is black), you can use either triangle *f* or *i* (which are gray); and if someone gives you square *j* (which is blue), you can use triangle *d* (which is black) or triangle *f* or *I* (which are gray).

Example 3.3.9 – *Solution (2/2)*

Statement (b) says that there is one particular triangle in the Tarski world that has a different color from every one of the squares in the world. This is false.

Two of the triangles are gray, but they cannot be used to show the truth of the statement because the Tarski world contains gray squares.

The only other triangle is black, but it cannot be used either because there is a black square in the Tarski world. Thus one of the statements is true and the other is false, and so they have opposite truth values.

Formal Logical Notation

Formal Logical Notation (1/2)

In some areas of computer science, logical statements are expressed in purely symbolic notation.

The notation involves using predicates to describe all properties of variables and omitting the words *such that* in existential statements. The formalism also depends on the following facts:

"$\forall x$ in D, $P(x)$" can be written as "$\forall x\ (x \text{ in } D \rightarrow P(x))$,"

and

"$\exists x$ in D such that $P(x)$" can be written as "$\exists x\ (x \text{ in } D \wedge P(x))$."

Example 3.3.10 – *Formalizing Statements in a Tarski World (1/2)*

Consider once more the Tarski world of Figure 3.3.1:

Figure 3.3.1

Example 3.3.10 – *Formalizing Statements in a Tarski World (2/2)*

Let Triangle(x), Circle(x), and Square(x) mean "x is a triangle," "x is a circle," and "x is a square"; let Blue(x), Gray(x), and Black(x) mean "x is blue," "x is gray," and "x is black"; let RightOf(x, y), Above(x, y), and SameColorAs(x, y) mean "x is to the right of y," "x is above y," and "x has the same color as y"; and use the notation $x = y$ to denote the predicate "x is equal to y."

Let the common domain D of all variables be the set of all the objects in the Tarski world.

Example 3.3.10 – *Formalizing Statements in a Tarski World* continued

Use formal logical notation to write each of the following statements, and write a formal negation for each statement.

a. For every circle x, x is above f.

b. There is a square x such that x is black.

c. For every circle x, there is a square y such that x and y have the same color.

d. There is a square x such that for every triangle y, x is to the right of y.

Example 3.3.10 – *Solution (1/5)*

a. *Statement:* $\forall x(\text{Circle}(x) \rightarrow \text{Above}(x, f))$

Negation: $\sim(\forall x\ (\text{Circle}(x) \rightarrow \text{Above}(x, f)))$

$$\equiv \exists x \sim(\text{Circle}(x) \rightarrow \text{Above}(x, f))$$

by the law for negating a ∀ statement

$$\equiv \exists x(\text{Circle}(x) \wedge \sim\text{Above}(x, f))$$

by the law of negating an if-then statement

b. *Statement:* $\exists x(\text{Square}(x) \wedge \text{Black}(x))$

Negation: $\sim(\exists x(\text{Square}(x) \wedge \text{Black}(x)))$

Example 3.3.10 – *Solution (2/5)* continued

$$\equiv \forall x \sim(\text{Square}(x) \wedge \text{Black}(x))$$

by the law for negating a ∃ statement

$$\equiv \forall x(\sim\text{Square}(x) \vee \sim\text{Black}(x))$$

by De Morgan's law

c. *Statement:* $\forall x(\text{Circle}(x) \rightarrow \exists y(\text{Square}(y) \wedge \text{SameColor}(x, y)))$

Negation: $\sim(\forall x(\text{Circle}(x) \rightarrow \exists y(\text{Square}(y) \wedge \text{SameColor}(x, y))))$

$$\equiv \exists x \sim(\text{Circle}(x) \rightarrow \exists y(\text{Square}(y) \wedge \text{SameColor}(x, y)))$$

by the law for negating a ∀ statement

Example 3.3.10 – *Solution (3/5)* continued

$$\equiv \exists x(\text{Circle}(x) \wedge \sim(\exists y(\text{Square}(y) \wedge \text{SameColor}(x, y))))$$

by the law for negating an if-then statement

$$\equiv \exists x(\text{Circle}(x) \wedge \forall y(\sim(\text{Square}(y) \wedge \text{SameColor}(x, y))))$$

by the law for negating a ∃ statement

$$\equiv \exists x(\text{Circle}(x) \wedge \forall y(\sim\text{Square}(y) \vee \sim\text{SameColor}(x, y)))$$

by De Morgan's law

Example 3.3.10 – *Solution (4/5)* continued

d. *Statement:* $\exists x(\text{Square}(x) \wedge \forall y(\text{Triangle}(y) \rightarrow \text{RightOf}(x, y)))$

Negation: $\sim(\exists x(\text{Square}(x) \wedge \forall y(\text{Triangle}(y) \rightarrow \text{RightOf}(x, y))))$

$$\equiv \forall x \sim(\text{Square}(x) \wedge \forall y(\text{Triangle}(x) \rightarrow \text{RightOf}(x, y)))$$

by the law for negating a ∃ statement

$$\equiv \forall x(\sim\text{Square}(x) \vee \sim(\forall y(\text{Triangle}(y) \rightarrow \text{RightOf}(x, y))))$$

by De Morgan's law

$$\equiv \forall x(\sim\text{Square}(x) \vee \exists y(\sim(\text{Triangle}(y) \rightarrow \text{RightOf}(x, y))))$$

by the law for negating a ∀ statement

Example 3.3.10 – *Solution (5/5)* continued

$$\equiv \forall x(\sim \text{Square}(x) \vee \exists y(\text{Triangle}(y) \wedge \sim \text{RightOf}(x, y)))$$

by the law for negating an if-then statement

Formal Logical Notation (2/2)

The disadvantage of the fully formal notation is that because it is complex and somewhat remote from intuitive understanding, when we use it, we may make errors that go unrecognized.

The advantage, however, is that operations, such as taking negations, can be made completely mechanical and programmed on a computer.

Taken together, the symbols for quantifiers, variables, predicates, and logical connectives make up what is known as the **language of first-order logic**.

Prolog

Prolog (1/1)

A simple Prolog program consists of a set of statements describing some situation together with questions about the situation.

Built into the language are search and inference techniques needed to answer the questions by deriving the answers from the given statements.

This frees the programmer from the necessity of having to write separate programs to answer each type of question.

Example 3.3.11 – *A Prolog Program (1/5)*

Consider the following picture, which shows colored blocks stacked on a table.

Example 3.3.11 – *A Prolog Program (2/5)*

The following are statements in Prolog that describe this picture and ask two questions about it.

isabove(g, b_1) color(g, gray) color(b_3, blue)

isabove(b_1, w_1) color(b_1, blue) color(w_1, white)

isabove(w_2, b_2) color(b_2, blue) color(w_2, white)

isabove(b_2, b_3) isabove(X, Z) if isabove(X, Y) and isabove(Y, Z)

1. ?color(b_1, blue) 2. ?isabove(X, w_1)

Example 3.3.11 – *A Prolog Program (3/5)*

The statements “isabove(g, b_1)” and “color(g, gray)” are to be interpreted as “g is above b_1” and “g is colored gray.”

The statement “isabove(X, Z) if isabove(X, Y) and isabove(Y, Z)” is to be interpreted as “For all X, Y, and Z, if X is above Y and Y is above Z, then X is above Z.” Statement 1

?color(b_1, blue)

asks whether block b_1 is colored blue. Prolog answers this by writing

Yes.

Example 3.3.11 – *A Prolog Program (4/5)*

Statement 2

$$?\text{isabove}(X, w_1)$$

asks for which blocks X the predicate "X is above w_1" is true. Prolog answers by giving a list of all such blocks. In this case, the answer is

$$X = b_1,\ X = g.$$

Note that Prolog can find the solution $X = b_1$ by merely searching the original set of given facts.

Example 3.3.11 – *A Prolog Program (5/5)*

However, Prolog must *infer* the solution $X = g$ from the following statements:

isabove(g, b_1),

isabove(b_1, w_1),

isabove(X, Z) if isabove(X, Y) and isabove(Y, Z).

Write the answers Prolog would give if the following questions were added to the program above.

a. ?isabove(b_2, w_1) b. ?color(w_1, X) c. ?color(X, blue)

Example 3.3.11 – *Solution*

a. The question means "Is b_2 above w_1?"; so the answer is "No."

b. The question means "For what colors X is the predicate 'w_1 is colored X ' true?"; so the answer is "X = white."

c. The question means "For what blocks is the predicate 'X is colored blue' true?"; so the answer is "$X = b_1$," "$X = b_2$," and "$X = b_3$."

CHAPTER 3

THE LOGIC OF QUANTIFIED STATEMENTS

Copyright © Cengage Learning. All rights reserved.

3.4 Arguments with Quantified Statements

Copyright © Cengage Learning. All rights reserved.

Arguments with Quantified Statements (1/5)

The rule of *universal instantiation* (in-stan-she-AY-shun) says the following:

Universal Instantiation

If a property is true of *everything* in a set, then it is true of *any particular* thing in the set.

Use of the words *universal instantiation* indicates that the truth of a property in a particular case follows as a special instance of its more general or universal truth.

Arguments with Quantified Statements (2/5)

The validity of this argument form follows immediately from the definition of truth values for a universal statement. One of the most famous examples of universal instantiation is the following:

All men are mortal.

Socrates is a man.

$\therefore$ Socrates is mortal.

Arguments with Quantified Statements (3/5)

As an example of universal instantiation, suppose you are doing a problem that requires you to simplify

$$r^{k+1} \cdot r,$$

where *r* is a particular real number and *k* is a particular integer. You know from your study of algebra that the following universal statements are true:

1. For every real number *x* and for all integers *m* and *n*, $x^m \cdot x^n = x^{m+n}$.
2. For every real number *x*, $x^1 = x$.

Arguments with Quantified Statements (4/5)

So you proceed as follows:

$$
\begin{aligned}
r^{k+1} \cdot r &= r^{k+1} \cdot r^1 && \text{Step 1} \\
&= r^{(k+1)+1} && \text{Step 2} \\
&= r^{k+2} && \text{by basic algebra.}
\end{aligned}
$$

Here is the reasoning behind steps 1 and 2.

Step 1: For every real number *x*, $x^1 = x$. universal truth

r is a particular real number. particular instance

$\therefore r^1 = r.$ conclusion

Arguments with Quantified Statements (5/5)

Step 2: For every real number *x* and for all integers *m* and *n*, $x^m \cdot x^n = x^{m+n}$. universal truth

r is a particular real number and *k* + 1 and 1 are particular integers. particular instance

$\therefore r^{k+1} \cdot r^1 = r^{(k+1)+1}$. conclusion

Both arguments are examples of universal instantiation.

Universal Modus Ponens

Universal Modus Ponens (1/1)

The rule of universal instantiation can be combined with modus ponens to obtain the valid form of argument called *universal modus ponens*.

Universal Modus Ponens

Formal Version	*Informal Version*
$\forall x$, if $P(x)$ then $Q(x)$.	If x makes $P(x)$ true, then x makes $Q(x)$ true.
$P(a)$ for a particular a.	a makes $P(x)$ true.
$\therefore Q(a)$.	$\therefore a$ makes $Q(x)$ true.

Example 3.4.1 – *Recognizing Universal Modus Ponens*

Rewrite the following argument using quantifiers, variables, and predicate symbols. Is this argument valid? Why?

If an integer is even, then its square is even.

k is a particular integer that is even.

$\therefore k^2$ is even.

Example 3.4.1 – *Solution (1/2)*

The major premise of this argument can be rewritten as

$\forall x$, if x is an even integer then x^2 is even.

Let $E(x)$ be "x is an even integer," let $S(x)$ be "x^2 is even," and let k stand for a particular integer that is even. Then the argument has the following form:

$\forall x$, if $E(x)$ then $S(x)$.

$E(k)$, for a particular k.

$\therefore S(k)$.

Example 3.4.1 – *Solution (2/2)* continued

This argument has the form of universal modus ponens and is therefore valid.

Example 3.4.2 – *Drawing Conclusions Using Universal Modus Ponens*

Write the conclusion that can be inferred using universal modus ponens.

If *T* is any right triangle with hypotenuse *c* and legs *a* and *b*, then $c^2 = a^2 + b^2$.

The triangle shown at the right is a right triangle with both legs equal to 1 and hypotenuse *c*.

$\therefore$ _____

Pythagorean Theorem

Example 3.4.2 – *Solution*

$$c^2 = 1^2 + 1^2 = 2$$

Note that if you take the nonnegative square root of both sides of this equation, you obtain $c = \sqrt{2}$.

This shows that there is a line segment whose length is $\sqrt{2}$.

Use of Universal Modus Ponens in a Proof

Use of Universal Modus Ponens in a Proof (1/4)

Here is a proof that the sum of any two even integers is even. It makes use of the definition of even integer, namely, that an integer is *even* if, and only if, it equals twice some integer. (Or, more formally: ∀ integers x, x is even if, and only if, ∃ an integer—say, k—such that $x = 2k$.) Suppose m and n are particular but arbitrarily chosen even integers. Then $m = 2r$ for some integer r,(1) and $n = 2s$ for some integer s.(2) Hence

$M + n = 2r + 2s$ by substitution

$= 2(r + s)$(3) by factoring out the 2.

Use of Universal Modus Ponens in a Proof (2/4)

Now $r + s$ is an integer, (4) and so $2(r + s)$ is even. (5)
Thus $m + n$ is even.

The following expansion of the proof shows how each of the numbered steps is justified by arguments that are valid by universal modus ponens.

(1) If an integer is even, then it equals twice some integer.
m is a particular even integer.
$\therefore$ m equals twice some integer, say, r.

Use of Universal Modus Ponens in a Proof (3/4)

(2) If an integer is even, then it equals twice some integer.

n is a particular even integer.

$\therefore$ n equals twice some integer, say, s.

(3) If a quantity is an integer, then it is a real number.

r and s are particular integers.

$\therefore$ r and s are real numbers.

For all a, b, and c, if a, b, and c are real numbers, then $ab + ac = a(b + c)$.

2, r, and s are particular real numbers.

$\therefore$ $2r + 2s = 2(r + s)$.

Use of Universal Modus Ponens in a Proof (4/4)

(4) For all u and v, if u and v are integers, then $u + v$ is an integer.

r and s are two particular integers.

$\therefore$ $r + s$ is an integer.

(5) If a number equals twice some integer, then that number is even.

$2(r + s)$ equals twice the integer $r + s$.

$\therefore$ $2(r + s)$ is even.

Universal Modus Tollens

Universal Modus Tollens (1/1)

Another crucially important rule of inference is *universal modus tollens*. Its validity results from combining universal instantiation with modus tollens. Universal modus tollens is the heart of proof of contradiction, which is one of the most important methods of mathematical argument.

Universal Modus Tollens

Formal Version	*Informal Version*
$\forall x$, if $P(x)$ then $Q(x)$.	If x makes $P(x)$ true, then x makes $Q(x)$ true.
$\sim Q(a)$, for a particular a.	a does not make $Q(x)$ true.
$\therefore \sim P(a)$.	$\therefore$ a does not make $P(x)$ true.

Example 3.4.3 – *Recognizing the Form of Universal Modus Tollens*

Rewrite the following argument using quantifiers, variables, and predicate symbols. Write the major premise in conditional form. Is this argument valid? Why?

All human beings are mortal.

Zeus is not mortal.

$\therefore$ Zeus is not human.

Example 3.4.3 – *Solution*

The major premise can be rewritten as

$\forall x$, if x is human then x is mortal.

Let $H(x)$ be "x is human," let $M(x)$ be "x is mortal," and let Z stand for Zeus. The argument becomes

$\forall x$, if $H(x)$ then $M(x)$

$$\sim M(Z)$$

$$\therefore \sim H(Z).$$

This argument has the form of universal modus tollens and is therefore valid.

Example 3.4.4 – *Drawing Conclusions Using Universal Modus Tollens*

Write the conclusion that can be inferred using universal modus tollens.

All professors are absent-minded.

Tom Hutchins is not absent-minded.

$\therefore$ _____.

Example 3.4.4 – *Solution*

continued

Tom Hutchins is not a professor.

Proving Validity of Arguments with Quantified Statements

Proving Validity of Arguments with Quantified Statements (1/1)

Definition

To say that an *argument form* is **valid** means the following: No matter what particular predicates are substituted for the predicate symbols in its premises, if the resulting premise statements are all true, then the conclusion is also true. An *argument* is called **valid** if, and only if, its form is valid. It is called **sound** if, and only if, its form is valid and its premises are true.

Using Diagrams to Test for Validity

Using Diagrams to Test for Validity (1/4)

Consider the statement

All integers are rational numbers.

Or, formally,

$\forall$ integer n, n is a rational number.

Picture the set of all integers and the set of all rational numbers as disks. The truth of the given statement is represented by placing the integers disk entirely inside the rationals disk, as shown in Figure 3.4.1.

Figure 3.4.1

Using Diagrams to Test for Validity (2/4)

Because the two statements "$\forall x \in D$, $Q(x)$" and "$\forall x$, if x is in D then $Q(x)$" are logically equivalent, both can be represented by diagrams like the foregoing.

Example 3.4.5 – *Using a Diagram to Show Validity*

Use diagrams to show the validity of the following syllogism:

All human beings are mortal.

Zeus is not mortal.

$\therefore$ Zeus is not a human being.

Example 3.4.5 – *Solution (1/2)*

The major premise is pictured on the left in Figure 3.4.2 by placing a disk labeled “human beings” inside a disk labeled “mortals.” The minor premise is pictured on the right in Figure 3.4.2 by placing a dot labeled “Zeus” outside the disk labeled “mortals.”

Figure 3.4.2

Example 3.4.5 – *Solution (2/2)* continued

The two diagrams fit together in only one way, as shown in Figure 3.4.3.

Figure 3.4.3

Since the Zeus dot is outside the mortals disk, it is necessarily outside the human beings disk. Thus the truth of the conclusion follows necessarily from the truth of the premises. It is impossible for the premises of this argument to be true and the conclusion false; hence the argument is valid.

Example 3.4.6 – *Using Diagrams to Show Invalidity*

Use a diagram to show the invalidity of the following argument:

All human beings are mortal.

Felix is mortal.

∴ Felix is a human being.

Example 3.4.6 – *Solution (1/3)*

The major and minor premises are represented diagrammatically in Figure 3.4.4.

Figure 3.4.4

All that is known is that the Felix dot is located *somewhere* inside the mortals disk.

Example 3.4.6 – *Solution (2/3)* continued

Where it is located with respect to the human beings disk cannot be determined. Either one of the situations shown in Figure 3.4.5 might be the case.

Figure 3.4.5

The conclusion “Felix is a human being” is true in the first case but not in the second (Felix might, for example, be a cat).

Example 3.4.6 – *Solution (3/3)* continued

Because the conclusion does not necessarily follow from the premises, the argument is invalid.

Using Diagrams to Test for Validity (3/4)

Converse Error (Quantified Form)

Formal Version	*Informal Version*
$\forall x$, if $P(x)$ then $Q(x)$.	If x makes $P(x)$ true, then x makes $Q(x)$ true.
$Q(a)$ for a particular a.	a makes $Q(x)$ true.
$\therefore P(a)$. ← invalid conclusion	$\therefore a$ makes $P(x)$ true. ← invalid conclusion

The following form of argument would be valid if a conditional statement were logically equivalent to its inverse. But it is not, and the argument form is invalid. We say that it exhibits the inverse error.

Using Diagrams to Test for Validity (4/4)

Inverse Error (Quantified Form)

Formal Version	*Informal Version*
$\forall x$, if $P(x)$ then $Q(x)$.	If x makes $P(x)$ true, then x makes $Q(x)$ true.
$\sim P(a)$, for a particular a.	a does not make $P(x)$ true.
$\therefore \sim Q(a)$. ← invalid conclusion	$\therefore$ a does not make $Q(x)$ true. ← invalid conclusion

Example 3.4.7 – *An Argument with "No"*

Use diagrams to test the following argument for validity:

No polynomial functions have horizontal asymptotes.
This function has a horizontal asymptote.
$\therefore$ This function is not a polynomial function.

Example 3.4.7 – *Solution (1/2)*

A good way to represent the major premise diagrammatically is shown in Figure 3.4.6, two disks—a disk for polynomial functions and a disk for functions with horizontal asymptotes—that do not overlap at all.

The minor premise is represented by placing a dot labeled “this function” inside the disk for functions with horizontal asymptotes.

Figure 3.4.6

Example 3.4.7 – *Solution (2/2)* continued

The diagram shows that “this function” must lie outside the polynomial functions disk, and so the truth of the conclusion necessarily follows from the truth of the premises. Hence the argument is valid.

Creating Additional Forms of Argument

Creating Additional Forms of Argument (1/1)

Universal modus ponens and modus tollens were obtained by combining universal instantiation with modus ponens and modus tollens. In the same way, additional forms of arguments involving universally quantified statements can be obtained by combining universal instantiation with other of the valid argument forms.

Universal Transitivity

Formal Version	*Informal Version*
$\forall x\, P(x) \rightarrow Q(x)$.	Any x that makes $P(x)$ true makes $Q(x)$ true.
$\forall x\, Q(x) \rightarrow R(x)$.	Any x that makes $Q(x)$ true makes $R(x)$ true.
$\therefore \forall x\, P(x) \rightarrow R(x)$.	$\therefore$ Any x that makes $P(x)$ true makes $R(x)$ true.

Example 3.4.8 – *Evaluating an Argument for Tarski's World (1/2)*

The following argument refers to the kind of arrangement of objects of various types and colors described in Tarski world image shown below.

Example 3.4.8 – *Evaluating an Argument for Tarski's World (2/2)*

Reorder and rewrite the premises to show that the conclusion follows as a valid consequence from the premises.

1. All the triangles are blue.
2. If an object is to the right of all the squares, then it is above all the circles.
3. If an object is not to the right of all the squares, then it is not blue.

∴ All the triangles are above all the circles.

Example 3.4.8 – *Solution (1/5)*

It is helpful to begin by rewriting the premises and the conclusion in if-then form:

1. $\forall x$, if x is a triangle, then x is blue.
2. $\forall x$, if x is to the right of all the squares, then x is above all the circles.
3. $\forall x$, if x is not to the right of all the squares, then x is not blue.

$\therefore$ $\forall x$, if x is a triangle, then x is above all the circles.

Example 3.4.8 – *Solution (2/5)* continued

The goal is to reorder the premises so that the conclusion of each is the same as the hypothesis of the next.

Also, the hypothesis of the argument's conclusion should be the same as the hypothesis of the first premise, and the conclusion of the argument's conclusion should be the same as the conclusion of the last premise.

To achieve this goal, it may be necessary to rewrite some of the statements in contrapositive form.

Example 3.4.8 – *Solution (3/5)* continued

In this example you can see that the first premise should remain where it is, but the second and third premises should be interchanged.

Then the hypothesis of the argument is the same as the hypothesis of the first premise, and the conclusion of the argument's conclusion is the same as the conclusion of the third premise.

But the hypotheses and conclusions of the premises do not quite line up. This is remedied by rewriting the third premise in contrapositive form.

Example 3.4.8 – *Solution (4/5)* continued

1. ∀*x*, if *x* is a triangle, then *x* is blue.
3. ∀*x*, if *x* is blue, then *x* is to the right of all the squares.
2. ∀*x*, if *x* is to the right of all the squares, then *x* is above all the circles.

Putting 1 and 3 together and using universal transitivity gives that

4. ∀*x*, if *x* is a triangle, then *x* is to the right of all the squares.

Example 3.4.8 – *Solution (5/5)* continued

And putting 4 together with 2 and using universal transitivity gives that

$\therefore$ $\forall x$, if x is a triangle, then x is above all the circles,

which is the conclusion of the argument.

Remark on the Converse and Inverse Errors

Remark on the Converse and Inverse Errors (1/6)

One reason why so many people make converse and inverse errors is that the forms of the resulting arguments would be valid if the major premise were a biconditional rather than a simple conditional.

Consider, for example, the following argument:

All the town criminals frequent the Den of Iniquity bar.

John frequents the Den of Iniquity bar.

∴ John is one of the town criminals.

Remark on the Converse and Inverse Errors (2/6)

The conclusion of this argument is invalid—it results from making the converse error. Therefore, it may be false even when the premises of the argument are true. This type of argument attempts unfairly to establish guilt by association.

The closer, however, the major premise comes to being a biconditional, the more likely the conclusion is to be true. If hardly anyone but criminals frequent the bar and John also frequents the bar, then it is likely (though not certain) that John is a criminal.

Remark on the Converse and Inverse Errors (3/6)

On the basis of the given premises, it might be sensible to be suspicious of John, but it would be wrong to convict him.

A variation of the converse error is a very useful reasoning tool, provided that it is used with caution. It is the type of reasoning that is used by doctors to make medical diagnoses and by auto mechanics to repair cars.

Remark on the Converse and Inverse Errors (4/6)

It is the type of reasoning used to generate explanations for phenomena. It goes like this:

If a statement of the form

For every x, if $P(x)$ then $Q(x)$

is true, and if

$Q(a)$ is true, for a particular a,

then check out the statement $P(a)$; it just might be true.

Remark on the Converse and Inverse Errors (5/6)

For instance, suppose a doctor knows that

> For every *x*, if *x* has pneumonia, then *x* has a fever and chills, coughs deeply, and feels exceptionally tired and miserable.

And suppose the doctor also knows that

> John has a fever and chills, coughs deeply, and feels exceptionally tired and miserable.

Remark on the Converse and Inverse Errors (6/6)

On the basis of these data, the doctor concludes that a diagnosis of pneumonia is a strong possibility, though not a certainty.

The doctor will probably attempt to gain further support for this diagnosis through laboratory testing that is specifically designed to detect pneumonia.

Note that the closer a set of symptoms comes to being a necessary and sufficient condition for an illness, the more nearly certain the doctor can be of his or her diagnosis.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.1 Direct Proof and Counterexample I: Introduction

Copyright © Cengage Learning. All rights reserved.

Direct Proof and Counterexample I: Introduction (1/2)

Both discovery and proof are integral parts of problem solving. When you think you have discovered that a certain statement is true, try to figure out why it is true. If you succeed, you will know that your discovery is genuine.

Even if you fail, the process of trying will give you insight into the nature of the problem and may lead you to discover that the statement is false. For complex problems, the interplay between discovery and proof is not reserved to the end of the problem-solving process but, rather, is an important part of each step.

Direct Proof and Counterexample I: Introduction (2/2)

Assumptions

- In this text we assume a familiarity with the laws of basic algebra.
- We also use the three properties of equality: For all objects A, B, and C, (1) $A = A$, (2) if $A = B$, then $B = A$, and (3) if $A = B$ and $B = C$, then $A = C$.
- And we use the principle of substitution: For all objects A and B, if $A = B$, then we may substitute B wherever we have A.
- In addition, we assume that there is no integer between 0 and 1 and that the set of all integers is closed under addition, subtraction, and multiplication. This means that sums, differences, and products of integers are integers.

Even, Odd, Prime, and Composite Integers

Even, Odd, Prime, and Composite Integers (1/2)

Definitions

An integer n is **even** if, and only if, n equals twice some integer. An integer n is **odd** if, and only if, n equals twice some integer plus 1.

Symbolically, for any integer, n

$$n \text{ is even} \quad \Leftrightarrow \quad n = 2k \text{ for some integer } k$$

$$n \text{ is odd} \quad \Leftrightarrow \quad n = 2k + 1 \text{ for some integer } k$$

Example 4.1.1 – *Even and Odd Integers*

Use the definitions of *even* and *odd* to justify your answers to the following questions.

a. Is 0 even?

b. Is −301 odd?

c. If *a* and *b* are integers, is $6a^2b$ even?

d. If *a* and *b* are integers, is $10a + 8b + 1$ odd?

e. Is every integer either even or odd?

Example 4.1.1 – *Solution (1/2)*

a. Yes, 0 is even because $0 = 2 \cdot 0$.

b. Yes, -301 is odd because $-301 = 2(-151) + 1$ and -151 is an integer.

c. Yes, $6a^2b$ is even because $6a^2b = 2(3a^2b)$ and $3a^2b$ is an integer since it is a product of integers.

d. Yes, $10a + 8b + 1$ is odd because

$10a + 8b + 1 = 2(5a + 4b) + 1$ and $5a + 4b$ is an integer since it is a sum of products of integers.

Example 4.1.1 – *Solution (2/2)* continued

e. Yes, every integer is either even or odd. However, the reason for this fact is not immediately apparent. It can be deduced using the method of proof by contradiction. It is also a consequence of the quotient-remainder theorem.

Even, Odd, Prime, and Composite Integers (2/2)

Definition

An integer n is **prime** if, and only if, $n > 1$ and for all positive integers r and s, if $n = rs$, then either r or s equals n. An integer n is **composite** if, and only if, $n > 1$ and $n = rs$ for some integers r and s with $1 < r < n$ and $1 < s < n$.

In symbols: For each integer n with $n > 1$,

$$n \text{ is prime} \quad \Leftrightarrow \quad \forall \text{ positive integers } r \text{ and } s, \text{ if } n = rs \text{ then either } r = 1 \text{ and } s = n \text{ or } r = n \text{ and } s = 1.$$

$$n \text{ is composite} \quad \Leftrightarrow \quad \exists \text{ positive integers } r \text{ and } s \text{ such that } n = rs \text{ and } 1 < r < n \text{ and } 1 < s < n.$$

Example 4.1.2 – *Prime and Composite Numbers*

a. Is 1 prime?

b. Is every integer greater than 1 either prime or composite?

c. Write the first six prime numbers.

d. Write the first six composite numbers.

Example 4.1.2 – *Solution (1/3)*

a. No. A prime number is required to be greater than 1.

b. Yes. Let n be any integer that is greater than 1. Consider all pairs of positive integers r and s such that $n = rs$. There exist at least two such pairs, namely, $r = n$ and $s = 1$ and $r = 1$ and $s = n$. Moreover, since $n = rs$, all such pairs satisfy the inequalities $1 \leq r \leq n$ and $1 \leq s \leq n$.

Example 4.1.2 – *Solution (2/3)* continued

If n is prime, then these two pairs are the only ways to write n as rs. Otherwise, there exists a pair of positive integers r and s such that $n = rs$ and neither r nor s equals either 1 or n. Therefore, in this case $1 < r < n$ and $1 < s < n$, and hence n is composite.

c. 2, 3, 5, 7, 11, 13

d. 4, 6, 8, 9, 10, 12

Example 4.1.2 – *Solution (3/3)* continued

c. 2, 3, 5, 7, 11, 13

d. 4, 6, 8, 9, 10, 12

Proving Existential Statements

Proving Existential Statements (1/3)

According to the definition, a statement in the form

$$\exists x \in D \text{ such that } Q(x)$$

is true if, and only if,

$$Q(x) \text{ is true for at least one } x \text{ in } D.$$

One way to prove this is to find an x in D that makes $Q(x)$ true. Another way is to give a set of directions for finding such an x. Both of these methods are called **constructive proofs of existence**. The logical principle underlying such a proof is called **existential generalization**.

Proving Existential Statements (2/3)

It says that if you know a certain property is true for a particular object, then you may conclude that “there exists an object for which the property is true.”

Example 4.1.3 – *Constructive Proofs of Existence*

a. Prove: ∃ an even integer n that can be written in two ways as a sum of two prime numbers.

b. Suppose that r and s are integers. Prove: ∃ an integer k such that $22r + 18s = 2k$.

Example 4.1.3 – *Solution*

a. Let $n = 10$. Then $10 = 5 + 5 = 3 + 7$ and 3, 5, and 7 are all prime numbers. Thus $\exists$ an even integer—namely, 10—that can be written in two ways as a sum of two prime numbers.

b. Let $k = 11r + 9s$. Then k is an integer because it is a sum of products of integers, and by substitution, and the distributive law of algebra,

$$2k = 2(11r + 9s) = 22r + 18s.$$

Thus $\exists$ an integer, namely k, such that $22r + 18s = 2k$.

Proving Existential Statements (3/3)

A **nonconstructive proof of existence** involves showing either (a) that the existence of a value of x that makes $Q(x)$ true is guaranteed by an axiom or a previously proved theorem or (b) that the assumption that there is no such x leads to a contradiction.

The disadvantage of a nonconstructive proof is that it may give virtually no clue about where or how x may be found. The widespread use of digital computers in recent years has led to some dissatisfaction with this aspect of nonconstructive proofs and to increased efforts to produce constructive proofs containing directions for computer calculation of the quantity in question.

Disproving Universal Statements by Counterexample

Disproving Universal Statements by Counterexample (1/2)

Disproof by Counterexample

To disprove a statement of the form "$\forall x \in D$, if $P(x)$ then $Q(x)$," find a value of x in D for which the hypothesis $P(x)$ is true and the conclusion $Q(x)$ is false. Such an x is called a **counterexample**.

Example 4.1.4 – *Disproof by Counterexample*

Disprove the following statement by finding a counterexample:

∀ real numbers *a* and *b*, if $a^2 = b^2$ then $a = b$.

Example 4.1.4 – *Solution*

To disprove this statement, you need to find real numbers *a* and *b* such that the hypothesis $a^2 = b^2$ is true and the conclusion *a* = *b* is false.

The fact that both positive and negative integers have positive squares helps in the search.

If you flip through some possibilities in your mind, you will quickly see that 1 and −1 will work (or 2 and −2, or 0.5 and −0.5, and so forth). You only need one such pair to give a counterexample.

Disproving Universal Statements by Counterexample (2/2)

Statement: $\forall$ real numbers a and b, if $a^2 = b^2$, then $a = b$.

Counterexample: Let $a = 1$ and $b = -1$. Then $a^2 = 1^2 = 1$ and $b^2 = (-1)^2 = 1$, and so $a^2 = b^2$. But $a \neq b$ since $1 \neq -1$.

Proving Universal Statements

Example 4.1.5 – *The Method of Exhaustion*

Use the method of exhaustion to prove the following statement:

$\forall n \in \mathbf{Z}$, if n is even and $4 \leq n \leq 26$ then n can be written as a sum of two prime numbers.

Example 4.1.5 – *Solution*

$4 = 2 + 2$ $6 = 3 + 3$ $8 = 3 + 5$ $10 = 5 + 5$

$12 = 5 + 7$ $14 = 11 + 3$ $16 = 5 + 11$ $18 = 7 + 11$

$20 = 7 + 13$ $22 = 5 + 17$ $24 = 5 + 19$ $26 = 7 + 19$

Proving Universal Statements (1/2)

The most powerful technique for proving a universal statement is one that works regardless of the size of the domain over which the statement is quantified.

It is based on a logical principle sometimes called *universal generalization*. A more descriptive name is *generalizing from the generic particular*.

Generalizing from the Generic Particular

To show that *every* element of a set satisfies a certain property, suppose x is a *particular* but *arbitrarily chosen* element of the set, and show that x satisfies the property.

Example 4.1.6 – *Generalizing from the Generic Particular (1/3)*

At some time you may have been shown a “mathematical trick” like the following. You ask a person to pick any number, add 5, multiply by 4, subtract 6, divide by 2, and subtract twice the original number.

Then you astound the person by announcing that their final result was 7. How does this “trick” work? Imagine that the empty box $\square$ contains whatever number the person picked.

Example 4.1.6 – *Generalizing from the Generic Particular (2/3)*

The table shows that by the end of the calculations, whatever was in the empty box was subtracted out of the answer.

Step	Visual Result	Algebraic Result
Pick a number.	□	x
Add 5.	□ |||||	$x + 5$
Multiply by 4.	□ ||||| □ ||||| □ ||||| □ |||||	$(x + 5) \cdot 4 = 4x + 20$
Subtract 6.	□ || □ || □ ||||| □ |||||	$(4x + 20) - 6 = 4x + 14$
Divide by 2.	□ || □ |||||	$\frac{4x + 14}{2} = 2x + 7$
Subtract twice the original number.	|| |||||	$(2x + 7) - 2x = 7$

Example 4.1.6 – *Generalizing from the Generic Particular (3/3)*

The *x* in the table above is another way of representing the number the person picked. It is *particular* (because it represents a single quantity), but it is also *arbitrarily chosen* or *generic* (because any number whatsoever can be put in its place).

So you can generalize from the generic particular to conclude that if you follow the steps you will always get 7, regardless of the initial value you put in place of *x*.

Proving Universal Statements (2/2)

When the method of generalizing from the generic particular is applied to a property of the form "If $P(x)$ then $Q(x)$," the result is the method of *direct proof*.

Method of Direct Proof

1. Express the statement to be proved in the form "For every $x \in D$, if $P(x)$ then $Q(x)$." (This step is often done mentally.)
2. Start the proof by supposing x is a particular but arbitrarily chosen element of D for which the hypothesis $P(x)$ is true. (This step is often abbreviated "Suppose $x \in D$ and $P(x)$.")
3. Show that the conclusion $Q(x)$ is true by using definitions, previously established results, and the rules for logical inference.

Example 4.1.7 – *A Direct Proof of a Theorem*

Prove that the sum of any two even integers is even.

Example 4.1.7 – *Solution (1/7)*

Whenever you are presented with a statement to be proved, it is a good idea to ask yourself whether you believe it to be true. In this case you might imagine some pairs of even integers—say 2 + 4, 6 + 10, 12 + 12, 28 + 54—and mentally check that their sums are even.

However, since you cannot possibly check all pairs of even numbers, you cannot know for sure that the statement is true in general by checking its truth in these particular instances. Many properties hold for a large number of examples and yet fail to be true in general.

Example 4.1.7 – *Solution (2/7)* continued

To prove this statement in general, you need to show that no matter what even integers are given, their sum is even.

But given any two even integers, it is possible to represent them as $2r$ and $2s$ for some integers r and s.

And by the distributive law of algebra, $2r + 2s = 2(r + s)$, which is even because $r + s$ is an integer. Thus the statement is true in general.

Example 4.1.7 – *Solution (3/7)* continued

Suppose the statement to be proved is much more complicated than this. What method can you use to derive a proof? You can begin by expressing the statement formally.

Formal Restatement: ∀ integers m and n, if m and n are even then $m + n$ is even.

This statement is universally quantified over an infinite domain. Thus to prove it in general, you need to show that no matter what two integers you might be given, if both of them are even then their sum will also be even.

Example 4.1.7 – *Solution (4/7)* continued

Next ask yourself, “How should I start the proof?” or “What am I supposing?” The answer to such a question gives you the starting point, or first sentence, of the proof.

Starting Point: Suppose *m* and *n* are any particular but arbitrarily chosen integers that are even.

Or, in abbreviated form:

Suppose *m* and *n* are any even integers.

Then ask yourself, “What conclusion do I need to show in order to complete the proof?”

Example 4.1.7 – *Solution (5/7)* continued

To Show: $m + n$ is even.

Since both involve the term *even integer*, you must use the definition of this term—and thus you must know what it means for an integer to be even.

It follows from the definition that since m and n are even, each equals twice some integer.

Example 4.1.7 – *Solution (6/7)*

continued

One of the basic laws of logic, called *existential instantiation*, says, in effect, that if you know something exists, you can give it a name. However, you cannot use the same name to refer to two different things, both of which are currently under discussion.

Existential Instantiation

If the existence of a certain kind of object is assumed or has been deduced, then it can be given a name, as long as that name is not currently being used to refer to something else in the same discussion.

Example 4.1.7 – *Solution (7/7)*

continued

Theorem 4.1.1

The sum of any two even integers is even.

Getting Proofs Started

Getting Proofs Started (1/1)

Believe it or not, once you understand the idea of generalizing from the generic particular and the method of direct proof, you can write the beginnings of proofs even for theorems you do not understand.

The reason is that the starting point and what is to be shown in a proof depend only on the linguistic form of the statement to be proved, not on the content of the statement.

Example 4.1.8 – *Identifying the "Starting Point" and the "Conclusion to Be Shown"*

Write the first sentence of a proof (the "starting point") and the last sentence of a proof (the "conclusion to be shown") for the following statement:

Every complete bipartite graph is connected.

Example 4.1.8 – *Solution (1/2)*

It is helpful to rewrite the statement formally using a quantifier and a variable:

Formal Restatement: For every $\overbrace{\text{graph}}^{\text{domain}}$ G, if $\overbrace{G \text{ is complete bipartite}}^{\text{hypothesis}}$, then $\overbrace{G \text{ is connected}}^{\text{conclusion}}$.

The first sentence, or starting point, of a proof supposes the existence of an object (in this case *G*) in the domain (in this case the set of all graphs) that satisfies the hypothesis of the if-then part of the statement (in this case that *G* is complete bipartite). The conclusion to be shown is just the conclusion of the if-then part of the statement (in this case that *G* is connected).

Example 4.1.8 – *Solution (2/2)* continued

Starting Point: Suppose G is a *[particular but arbitrarily chosen]* graph such that G is complete bipartite.

Conclusion to Be Shown: G is connected.

Thus the proof has the following shape:

Proof:

Suppose G is a *[particular but arbitrarily chosen]* graph such that G is complete bipartite.

⋮

Therefore, G is connected.

Example 4.1.9 – *Fill in the Blanks for a Proof (1/2)*

Fill in the blanks in the proof of the following theorem.

Theorem: For all integers r and s, if r is even and s is odd then $3r + 2s$ is even.

Proof: Suppose r and s are any *[particular but arbitrarily chosen]* integers such that r is even and s is odd.

[We must show that $3r + 2s$ is even.]

By __(a)__, $r = 2m$ and $s = 2n + 1$ for some integers m and n.

Example 4.1.9 – *Fill in the Blanks for a Proof (2/2)*

Then

$$3r + 2s = 3(2m) + 2(2n + 1)$$ by ___(b)___

$$= 6m + 4n + 2$$ by multiplying out

$$= 2(3m + 2n + 1)$$ by factoring out 2

Let $t = 3m + 2n + 1$.

Then t is an integer because m, n, 3, 2, and 1 are integers and because ___(c)___.

Example 4.1.9 – *Solution*

Hence $3r + 2s = 2t$, where t is an integer, and so by __(d)__, $3r + 2s$ is even *[as was to be shown].*

(a) definition of even and odd, (b) substitution, (c) products and sums of integers are integers, (d) definition of even.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.2 Direct Proof and Counterexample II: Writing Advice

Copyright © Cengage Learning. All rights reserved.

Directions for Writing Proofs of Universal Statements

Directions for Writing Proofs of Universal Statements (1/9)

Over the years, the following rules of style have become fairly standard for writing the final versions of proofs:

1. **Copy the statement of the theorem to be proved on your paper.**

   This makes the theorem statement available for reference to anyone reading the proof.

2. **Clearly mark the beginning of your proof with the word Proof.**

   This word separates general discussion about the theorem from its actual proof.

Directions for Writing Proofs of Universal Statements (2/9)

3. Make your proof self-contained.

This means that you should explain the meaning of each variable used in your proof in the body of the proof. Thus you will begin proofs by introducing the initial variables and stating what kind of objects they are. The first sentence of your proof would be something like "Suppose m and n are any even integers" or "Let x be a real number such that x is greater than 2." This is similar to declaring variables and their data types at the beginning of a computer program.

Directions for Writing Proofs of Universal Statements (3/9)

At a later point in your proof, you may introduce a new variable to represent a quantity that is known at that point to exist. For example, if you have assumed that a particular integer n is even, then you know that n equals 2 times some integer, and you can give this integer a name so that you can work with it concretely later in the proof. Thus if you decide to call the integer, say, s, you would write, "Since n is even, $n = 2s$ for some integer s," or "since n is even, there exists an integer, say s, such that $n = 2s$."

Directions for Writing Proofs of Universal Statements (4/9)

4. Write your proof in complete, grammatically correct sentences.

This does not mean that you should avoid using symbols and shorthand abbreviations, just that you should incorporate them into sentences. For example, the proof of Theorem 4.1.1 contains the sentence

Then $m + n = 2r + 2s$ by substitution

$= 2(r + s)$ by factoring out 2.

To read such text as a sentence, read the first equals sign as “equals” and each subsequent equals sign as “which equals.”

Directions for Writing Proofs of Universal Statements (5/9)

5. Keep your reader informed about the status of each statement in your proof.

Your reader should never be in doubt about whether something in your proof has been assumed or established or is still to be deduced. If something is assumed, preface it with a word like *Suppose* or *Assume*. If it is still to be shown, preface it with words like, *We must show that* or *In other words*, *we must show that*. This is especially important if you introduce a variable in rephrasing what you need to show.

Directions for Writing Proofs of Universal Statements (6/9)

6. Give a reason for each assertion in your proof.

Each assertion in a proof should come directly from the hypothesis of the theorem, or follow from the definition of one of the terms in the theorem, or be a result obtained earlier in the proof, or be a mathematical result that has previously been established or is agreed to be assumed. Indicate the reason for each step of your proof using phrases such as *by hypothesis*, *by definition of* … *by theorem* … and so forth.

Directions for Writing Proofs of Universal Statements (7/9)

7. Include the "little words and phrases" that make the logic of your arguments clear.

When writing a mathematical argument, especially a proof, indicate how each sentence is related to the previous one. Does it follow from the previous sentence or from a combination of the previous sentence and earlier ones? If so, start the sentence with the word *Because* or *Since* and state the reason why it follows, or write *Then*, or *Thus*, or *So*, or *Hence*, or *Therefore*, or *Consequently*, or *It follows that*, and include the reason at the end of the sentence.

Directions for Writing Proofs of Universal Statements (8/9)

If a sentence expresses a new thought or fact that does not follow as an immediate consequence of the preceding statement but is needed for a later part of a proof, introduce it by writing *Observe that*, or *Note that*, or *Recall that*, or *But*, or *Now*. Sometimes in a proof it is desirable to define a new variable in terms of previous variables. In such a case, introduce the new variable with the word *Let*.

Directions for Writing Proofs of Universal Statements (9/9)

8. Display equations and inequalities.

The convention is to display equations and inequalities on separate lines to increase readability, both for other people and for ourselves so that we can more easily check our work for accuracy.

Variations among Proofs

Variations among Proofs (1/1)

It is rare that two proofs of a given statement, written by two different people, are identical. Even when the basic mathematical steps are the same, the two people may use different notation or may give differing amounts of explanation for their steps, or may choose different words to link the steps together into paragraph form.

Common Mistakes

Common Mistakes (1/15)

The following are some of the most common mistakes people make when writing mathematical proofs.

1. Arguing from examples.

Looking at examples is one of the most helpful practices a problem solver can engage in and is encouraged by all good mathematics teachers. However, it is a mistake to think that a general statement can be proved by showing it to be true for some individual cases. A property referred to in a universal statement may be true in many instances without being true in general.

Common Mistakes (2/15)

Consider the following “proof” that the sum of any two even integers is even.

> This is true because if $m = 14$ and $n = 6$, which are both even, then $m + n = 20$, which is also even.

Some people find this kind of argument convincing because it does, after all, consist of evidence in support of a true conclusion. But remember that when we discussed valid arguments, we pointed out that an argument may be invalid and yet have a true conclusion.

Common Mistakes (3/15)

In the same way, an argument from examples may be mistakenly used to “prove” a true statement. In the previous example, it is not sufficient to show that the conclusion “$m + n$ is even” is true for $m = 14$ and $n = 6$. You must give an argument to show that the conclusion is true for any arbitrarily chosen even integers m and n.

Common Mistakes (4/15)

2. Using the same letter to mean two different things.

Some beginning theorem provers give a new variable quantity the same letter name as a previously introduced variable. Consider the following “proof” fragment:

> Suppose m and n are any odd integers. Then by definition of odd, $m = 2k + 1$ and $n = 2k + 1$ where k is an integer.

You might think of a variable in a mathematical proof as similar to a global variable in a computer program: once introduced, it has the same meaning throughout the program. In other words, its scope extends to the end of the program.

Common Mistakes (5/15)

In this example, using the symbol k in the expressions for both m and n makes k a global variable. As a result, both m and n equal $2k + 1$, and thus are equal to each other. The proof then only shows that a sum of two identical odd integers is even, not that the sum of two arbitrarily chosen odd integers is even.

3. Jumping to a conclusion.

To jump to a conclusion means to allege the truth of something without giving an adequate reason. Consider the following “proof” that the sum of any two even integers is even.

Common Mistakes (6/15)

> Suppose m and n are any even integers. By definition of even, $m = 2r$ and $n = 2s$ for some integers r and s. Then $m + n = 2r + 2s$. So $m + n$ is even.

The problem with this "proof" is that to show an integer is even one needs to show that it equals twice some integer. This proof jumps to the conclusion that $m + n$ is even without having found an integer that, when doubled, equals $m + n$.

Common Mistakes (7/15)

4. Assuming what is to be proved.

To assume what is to be proved is a variation of jumping to a conclusion. As an example, consider the following "proof" of the fact that the product of any two odd integers is odd:

> Suppose *m* and *n* are any odd integers. When any odd integers are multiplied, their product is odd. Hence *mn* is odd.

Common Mistakes (8/15)

5. Confusion between what is known and what is still to be shown.

A more subtle way to jump to a conclusion occurs when the conclusion is restated using a variable. Here is an example in a "proof" that the product of any two odd integers is odd:

> Suppose m and n are any odd integers. We must show that mn is odd. This means that there exists an integer s such that
>
> $$mn = 2s + 1.$$

Common Mistakes (9/15)

Also by definition of odd, there exist integers a and b such that

$$m = 2a + 1 \text{ and } n = 2b + 1.$$

Then

$$mn = (2a + 1)(2b + 1) = 2s + 1.$$

So, since s is an integer, mn is odd by definition of odd.

In this example, when the author restated the conclusion to be shown (that mn is odd), the author wrote "there exists an integer s such that $mn = 2s + 1$."

Common Mistakes (10/15)

But we only know that the integer s exists if we know that mn is odd, which is what the author is trying to show. Thus, in the sentence starting with the word "Then," the author jumped to an unjustified conclusion. This mistake might have been avoided if the author had written

> "This means *we must show that there exists* an integer s such that $mn = 2s + 1$."

An even better way to avoid this kind of error is not to introduce a variable into a proof unless it is either part of the hypothesis or deducible from it.

Common Mistakes (11/15)

6. Use of *any* when the correct word is *some*.

There are a few situations in which the words *any* and *some* can be used interchangeably. For instance, in starting a proof that the square of any odd integer is odd, one could correctly write, “Suppose *m* is any odd integer” or “Suppose *m* is some odd integer.” In most situations, however, the words *any* and *some* are not interchangeable. Here is the start of a “proof” that the square of any odd integer is odd, which uses *any* when the correct word is *some*:

Common Mistakes (12/15)

> Suppose m is a particular but arbitrarily chosen odd integer. By definition of odd, $m = 2a + 1$ for any integer a.

In the second sentence it is incorrect to say that "$m = 2a + 1$ for any integer a" because a cannot be just "any" integer; in fact, solving $m = 2a + 1$ for a shows that the only possible value for a is $(m - 1)/2$. The correct way to finish the second sentence is, "$m = 2a + 1$ for some integer a" or "there exists an integer a such that $m = 2a + 1$."

Common Mistakes (13/15)

7. Misuse of the word *if.*

Another common error is not serious in itself, but it reflects imprecise thinking that sometimes leads to problems later in a proof. This error involves using the word *if* when the word *because* is really meant. Consider the following proof fragment:

> Suppose p is a prime number. If p is prime, then p cannot be written as a product of two smaller positive integers

Common Mistakes (14/15)

The use of the word *if* in the second sentence is inappropriate. It suggests that the primeness of p is in doubt. But p is known to be prime by the first sentence. It cannot be written as a product of two smaller positive integers *because* it is prime. Here is a correct version of the fragment:

> Suppose p is a prime number. Because p is prime, p cannot be written as a product of two smaller positive integers.

Example 4.2.1 – *An Odd Integer Minus an Even Integer (1/2)*

Prove that the difference of any odd integer and any even integer is odd. Use only the definitions of odd and even and the Assumptions listed below, not any other properties of odd and even integers.

Assumptions

- In this text we assume a familiarity with the laws of basic algebra.
- We also use the three properties of equality: For all objects A, B, and C, (1) $A = A$, (2) if $A = B$, then $B = A$, and (3) if $A = B$ and $B = C$, then $A = C$.
- And we use the principle of substitution: For all objects A and B, if $A = B$, then we may substitute B wherever we have A.
- In addition, we assume that there is no integer between 0 and 1 and that the set of all integers is closed under addition, subtraction, and multiplication. This means that sums, differences, and products of integers are integers.

Example 4.2.1 – *An Odd Integer Minus an Even Integer (2/2)*

Follow the directions given in this section for writing proofs of universal statements

Example 4.2.1 – *Solution (1/2)*

You may already have a sense that the statement to be proved is true, but to make sure your intuition is correct and to develop a careful proof, rewrite the statement using names such as a and b for the odd and even integers so that you will have a convenient way to refer to them:

> For all integers a and b, if a is odd and b is even, then $a - b$ is odd.

Or: For every odd integer a and every even integer b, the difference $a - b$ is odd.

Or: If a is any odd integer and b is any even integer, then $a - b$ is odd.

Example 4.2.1 – *Solution (2/2)* continued

Thus the starting point for your proof would be something like, “Suppose a is any odd integer and b is any even integer,” and the conclusion to be shown would be “We must show that $a - b$ is odd.”

If, in addition, you know how to use the definitions of odd and even, you will have reduced the creative part of developing the proof to a small, but crucial, section in the middle.

Common Mistakes (15/15)

Theorem 4.2.1

The difference of any odd integer and any even integer is odd.

Example 4.2.2 – *Identifying a Mistake in a Proposed Proof*

Find the mistake in the following "proof."

Theorem: If *n* is any even integer, then $(-1)^n = 1$.

Proof:

1. Suppose *n* is any even integer. *[We must show that* $(-1)^n$ *is even.]*

2. By definition of even, *n* = 2*a* for some integer *a*.

3. Then $(-1)^n = (-1)^{2a}$ by substitution

4. $= ((-1)^a)^2$ by a law of exponents

5. $= 1$ because any nonzero real number squared is positive.

Example 4.2.2 – *Solution*

This “proof” incorrectly jumps to a conclusion in line 5. Although it is true that the square of any nonzero real number is positive, it does not follow that the square of $(-1)^a$ is 1. Exercise 10 at the end of this section asks you to give a correct proof of this theorem.

Showing That an Existential Statement Is False

Showing That an Existential Statement Is False (1/1)

We know that the negation of an existential statement is universal. It follows that to prove an existential statement is false, you must prove a universal statement (its negation) is true.

Example 4.2.3 – *Disproving an Existential Statement*

Show that the following statement is false:

There is a positive integer *n* such that $n^2 + 3n + 2$ is prime.

Example 4.2.3 – *Solution (1/2)*

Proving that the given statement is false is equivalent to proving its negation is true. The negation is

For all positive integers *n*, $n^2 + 3n + 2$ is not prime.

Because the negation is universal, it is proved by generalizing from the generic particular.

Claim: The statement "There is a positive integer *n* such that $n^2 + 3n + 2$ is prime" is false.

Example 4.2.3 – *Solution (2/2)*

continued

Proof:

Suppose *n* is any *[particular but arbitrarily chosen]* positive integer. *[We will show that* $n^2 + 3n + 2$

Factoring shows that

$$n^2 + 3n + 2 = (n + 1)(n + 2).$$

In addition, *n* + 1 and *n* + 2 are integers (because they are sums of integers), and both *n* + 1 > 1 and *n* + 2 > 1 (because *n* ≥ 1). Thus $n^2 + 3n + 2$ is a product of two integers each greater than 1, and so $n^2 + 3n + 2$ is not prime.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.3 Direct Proof and Counterexample III: Rational Numbers

Copyright © Cengage Learning. All rights reserved.

Direct Proof and Counterexample III: Rational Numbers

Definition

A real number r is **rational** if, and only if, it can be expressed as a quotient of two integers with a nonzero denominator. A real number that is not rational is **irrational**. More formally, if r is a real number, then

$$r \text{ is rational} \Leftrightarrow \exists \text{ integers } a \text{ and } b \text{ such that } r = \frac{a}{b} \text{ and } b \neq 0.$$

Example 4.3.1 – *Determining Whether Numbers Are Rational or Irrational (1/2)*

a. Is 10/3 a rational number?

b. Is $-\frac{5}{39}$ a rational number?

c. Is 0.281 a rational number?

d. Is 7 a rational number?

e. Is 0 a rational number?

f. Is 2/0 a rational number?

g. Is 2/0 an irrational number?

Example 4.3.1 – *Determining Whether Numbers Are Rational or Irrational (2/2)*

h. Is 0.12121212… a rational number (where the digits 12 are assumed to repeat forever)?

i. If m and n are integers and neither m nor n is zero, is $(m + n)/mn$ a rational number?

Example 4.3.1 – *Solution (1/3)*

a. Yes, 10/3 is a quotient of the integers 10 and 3 and hence is rational.

b. Yes, $-\frac{5}{39} = \frac{-5}{39}$, which is a quotient of the integers −5 and 39 and hence is rational.

c. Yes, 0.281 = 281/1000. Note that the numbers shown on a typical calculator display are all finite decimals. An explanation similar to the one in this example shows that any such number is rational.

It follows that a calculator with such a display can accurately represent only rational numbers.

Example 4.3.1 – *Solution (2/3)* continued

d. Yes, $7 = 7/1$.

e. Yes, $0 = 0/1$.

f. No, $2/0$ is not a real number (division by 0 is not allowed).

g. No, because every irrational number is a real number, and $2/0$ is not a real number.

Example 4.3.1 – *Solution (3/3)* continued

h. Yes. Let $x = 0.12121212\ldots$ Then $100x = 12.12121212\ldots$

Thus

$$100x - x = 12.12121212\ldots - 0.12121212\ldots = 12.$$

But also $100x - x = 99x$ by basic algebra.

Hence $99x = 12$,

and so $$x = \frac{12}{99}.$$

Therefore, $0.12121212\ldots = 12/99$, which is a ratio of two nonzero integers and thus is a rational number.

Example 4.3.1 – *Solution*

continued

Note that you can use an argument similar to this one to show that any repeating decimal is a rational number.

i. Yes, since m and n are integers, so are $m + n$ and mn (because sums and products of integers are integers). Also $mn \neq 0$ by the *zero product property*.

One version of this property says the following:

Zero Product Property

If neither of two real numbers is zero, then their product is also not zero.

More on Generalizing from the Generic Particular

More on Generalizing from the Generic Particular (1/1)

Most problems are stated in informal language, but solving them often requires translating them into more formal terms.

Theorem 4.3.1

Every integer is a rational number.

Proving Properties of Rational Numbers

Example 4.3.2 – *Any Sum of Rational Numbers Is Rational*

Prove that the sum of any two rational numbers is rational.

Example 4.3.2 – *Solution (1/5)*

Begin by mentally or explicitly rewriting the statement to be proved in the form "∀ _____, if _____ then _____."

Formal Restatement: ∀ real numbers r and s, if r and s are rational then $r + s$ is rational.

Next ask yourself, "Where am I starting from?" or "What am I supposing?" The answer gives you the starting point, or first sentence, of the proof.

Example 4.3.2 – *Solution (2/5)* continued

Starting Point: Suppose r and s are any particular but arbitrarily chosen real numbers such that r and s are rational; *or, more simply,*

Suppose r and s are any rational numbers.

Then ask yourself, "What must I show to complete the proof?"

To Show: $r + s$ is rational.

Finally ask, "How do I get from the starting point to the conclusion?" or "Why must $r + s$ be rational if both r and s are rational?" The answer depends in an essential way on the definition of rational.

Example 4.3.2 – *Solution (3/5)* continued

Rational numbers are quotients of integers, so to say that *r* and *s* are rational means that

$$r = \frac{a}{b} \quad \text{and} \quad s = \frac{c}{d}$$

for some integers *a*, *b*, *c*, and *d* where $b \neq 0$ and $d \neq 0$.

It follows by substitution that

$$r + s = \frac{a}{b} + \frac{c}{d}. \qquad (4.3.1)$$

Example 4.3.2 – *Solution (4/5)* continued

You need to show that $r + s$ is rational, which means that $r + s$ can be written as a single fraction or ratio of two integers with a nonzero denominator.

But the right-hand side of equation (4.3.1) is

$$\frac{a}{b} + \frac{c}{d} = \frac{ad}{bd} + \frac{bc}{bd}$$

by rewriting the fraction with a common denominator

$$= \frac{ab + bc}{bd}$$

by adding fractions with a common denominator.

Example 4.3.2 – *Solution (5/5)* continued

Is this fraction a ratio of integers? Yes. Because products and sums of integers are integers, $ad + bc$ and bd are both integers.

Is the denominator $bd \neq 0$? Yes, by the zero product property (since $b \neq 0$ and $d \neq 0$). Thus $r + s$ is a rational number.

Theorem 4.3.2

The sum of any two rational numbers is rational.

Deriving New Mathematics from Old

Example 4.3.3 – *Deriving Additional Results about Even and Odd Integers (1/2)*

Suppose that you have already proved the following properties of even and odd integers:

1. The sum, product, and difference of any two even integers are even.
2. The sum and difference of any two odd integers are even.
3. The product of any two odd integers is odd.
4. The product of any even integer and any odd integer is even.

Example 4.3.3 – *Deriving Additional Results about Even and Odd Integers (2/2)*

5. The sum of any odd integer and any even integer is odd.
6. The difference of any odd integer minus any even integer is odd.
7. The difference of any even integer minus any odd integer is odd.

Use the properties listed above to prove that if *a* is any even integer and *b* is any odd integer, then $\frac{a^2 + b^2 + 1}{2}$ is an integer.

Example 4.3.3 – *Solution*

Suppose *a* is any even integer and *b* is any odd integer. By property 3, b^2 is odd, and by property 1, a^2 is even. Then by property 5, $a^2 + b^2$ is odd, and because 1 is also odd, the sum $(a^2 + b^2) + 1 = a^2 + b^2 + 1$ is even by property 2.

Hence, by definition of even, there exists an integer *k* such that $a^2 + b^2 + 1 = 2k.$

Dividing both sides by 2 gives $\frac{a^2 + b^2 + 1}{2} = k,$ which is an integer. Thus $\frac{a^2 + b^2 + 1}{2}$ is an integer.

Example 4.3.4 – *The Double of a Rational Number*

Derive the following as a corollary of Theorem 4.3.2.

Corollary 4.2.3

The double of a rational number is rational.

Example 4.3.4 – *Solution*

The double of a number is just its sum with itself. But since the sum of any two rational numbers is rational (Theorem 4.3.2), the sum of a rational number with itself is rational.

Hence the double of a rational number is rational. Here is a formal version of this argument:

Proof: Suppose r is any rational number. Then $2r = r + r$ is a sum of two rational numbers. So, by Theorem 4.3.2, $2r$ is rational.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.4 Direct Proof and Counterexample IV: Divisibility

Copyright © Cengage Learning. All rights reserved.

Direct Proof and Counterexample IV: Divisibility (1/4)

When you were first introduced to the concept of division in elementary school, you were probably taught that 12 divided by 3 is 4 because if you separate 12 objects into groups of 3, you get 4 groups with nothing left over.

You may also have been taught to describe this fact by saying that “12 is evenly divisible by 3” or “3 divides 12 evenly.”

Direct Proof and Counterexample IV: Divisibility (2/4)

The notion of divisibility is the central concept of one of the most beautiful subjects in advanced mathematics: **number theory**, the study of properties of integers.

Definition

If n and d are integers then

n is **divisible by** d if, and only if, n equals d times some integer and $d \neq 0$.

Instead of "n is divisible by d," we can say that

n **is a multiple of** d, or
d **is a factor of** n, or
d **is a divisor of** n, or
d **divides** n.

The notation $\mathbf{d|n}$ is read "d divides n." Symbolically, if n and d are integers:

$$d|n \quad \Leftrightarrow \quad \exists \text{ an integer, say } k, \text{ such that } n = dk \text{ and } d \neq 0.$$

The notation $\mathbf{d \nmid n}$ is read "d does not divide n."

Example 4.4.1 – *Divisibility*

a. Is 21 divisible by 3?

b. Does 5 divide 40?

c. Does $7 \mid 42$?

d. Is 32 a multiple of −16?

e. Is 6 a factor of 54?

f. Is 7 a factor of −7?

Example 4.4.1 – *Solution*

a. Yes, $21 = 3 \cdot 7$.

b. Yes, $40 = 5 \cdot 8$.

c. Yes, $42 = 7 \cdot 6$.

d. Yes, $32 = (-16) \cdot (-2)$.

e. Yes, $54 = 6 \cdot 9$.

f. Yes, $-7 = 7 \cdot (-1)$.

Example 4.4.2 – *Divisors of Zero*

If k is any nonzero integer, does k divide 0?

Example 4.4.2 – *Solution*

Yes, because $0 = k \cdot 0$.

Direct Proof and Counterexample IV: Divisibility (3/4)

Two useful properties of divisibility are (1) that if one positive integer divides a second positive integer, then the first is less than or equal to the second, and (2) that the only divisors of 1 are 1 and −1.

Theorem 4.4.1 A Positive Divisor of a Positive Integer

For all integers a and b, if a and b are positive and a divides b then $a \leq b$.

Theorem 4.4.2 Divisors of 1

The only divisors of 1 are 1 and -1.

Example 4.4.3 – *Divisibility and Algebraic Expressions*

a. If a and b are integers, is $3a + 3b$ divisible by 3?

b. If k and m are integers, is $10km$ divisible by 5?

Example 4.4.3 – *Solution*

a. Yes. By the distributive law of algebra, $3a + 3b = 3(a + b)$ and $a + b$ is an integer because it is a sum of two integers.

b. Yes. By the associative law of algebra, $10km = 5 \cdot (2km)$ and $2km$ is an integer because it is a product of three integers.

Direct Proof and Counterexample IV: Divisibility (4/4)

When the definition of divides is rewritten formally using the existential quantifier, the result is

$$d \mid n \quad \Leftrightarrow \quad \exists \text{ an integer } k \text{ such that } n = dk \text{ and } d \neq 0.$$

Since the negation of an existential statement is universal, it follows that *d* does not divide *n* (denoted $d \nmid n$) if, and only if, ∀ integer *k*, $n \neq dk$ or $d = 0$; in other words, the quotient n/d is not an integer.

$$\text{For all integers } n \text{ and } d, \quad d \nmid n \quad \Leftrightarrow \quad \frac{n}{d} \text{ is not an integer.}$$

Example 4.4.4 – *Checking Nondivisibility*

Does $4 \mid 15$?

Example 4.4.4 – *Solution*

No, $\frac{15}{4} = 3.75,$ which is not an integer.

Example 4.4.5 – *Prime Numbers and Divisibility*

An alternative way to define a prime number is to say that an integer $n > 1$ is prime if, and only if, its only positive integer divisors are 1 and itself.

Proving Properties of Divisibility

Example 4.4.6 – *Transitivity of Divisibility*

Prove that for all integers *a*, *b*, and *c*, if $a|b$ and $b|c$, then $a|c$.

Example 4.4.6 – *Solution (1/3)*

Since the statement to be proved is already written formally, you can immediately pick out the starting point, or first sentence of the proof, and the conclusion that must be shown.

Starting Point: Suppose *a*, *b*, and *c* are particular but arbitrarily chosen integers such that $a|b$ and $b|c$.

To Show: $a|c$.

You need to show that $a|c$, or, in other words, that

$$c = a \cdot (\text{some integer}).$$

Example 4.4.6 – *Solution (2/3)* continued

But since $a|b$,

$$b = ar \quad \text{for some integer } r. \qquad 4.4.1$$

And since $b|c$,

$$c = bs \quad \text{for some integer } s. \qquad 4.4.2$$

Equation 4.4.2 expresses *c* in terms of *b*, and equation 4.4.1 expresses *b* in terms of *a*. Thus if you substitute 4.4.1 into 4.4.2, you will have an equation that expresses *c* in terms of *a*.

Example 4.4.6 – *Solution (3/3)* continued

$c = bs$ by equation 4.4.2

$= (ar)s$ by equation 4.4.1.

But $(ar)s = a(rs)$ by the associative law for multiplication. Hence

$$c = a(rs).$$

Now you are almost finished. You have expressed c as $a \cdot$ (something). It remains only to verify that that something is an integer. But of course it is, because it is a product of two integers.

Proving Properties of Divisibility (1/1)

Theorem 4.4.3 Transitivity of Divisibility

For all integers a, b, and c, if a divides b and b divides c, then a divides c.

Theorem 4.4.4 Divisibility by a Prime

Any integer $n > 1$ is divisible by a prime number.

Counterexamples and Divisibility

Example 4.4.7 – *Checking a Proposed Divisibility Property*

Is the following statement true or false? For all integers *a* and *b*, if $a \mid b$ and $b \mid a$ then $a = b$.

Example 4.4.7 – *Solution (1/4)*

This statement is false. Can you think of a counterexample just by concentrating for a minute or so?

The following discussion describes a mental process that may take just a few seconds. It is helpful to be able to use it consciously, however, to solve more difficult problems.

To discover the truth or falsity of a statement such as the one given above, start off much as you would if you were trying to prove it.

Example 4.4.7 – *Solution (2/4)* continued

Starting Point: Suppose a and b are integers such that $a \mid b$ and $b \mid a$.

Ask yourself, "*Must* it follow that $a = b$, or *could* it happen that $a \neq b$ for some a and b?" Focus on the supposition. What does it mean? By definition of divisibility, the conditions $a \mid b$ and $b \mid a$ mean that

$$b = ha \quad \text{and} \quad a = kb \quad \text{for some integers } h \text{ and } k.$$

Must it follow that $a = b$, or can you find integers a and b that satisfy these equations for which $a \neq b$?

Example 4.4.7 – *Solution (3/4)* continued

The equations imply that

$$b = ha = h(kb) = (hk)b.$$

Since $b|a, b \neq 0,$ $b \neq 0$, and so you can cancel b from the extreme left and right sides to obtain

$$1 = hk.$$

In other words, h and k are divisors of 1. But, by Theorem 4.4.2, the only divisors of 1 are 1 and −1. Thus h and k are both 1 or are both −1.

Example 4.4.7 – *Solution (4/4)*

continued

If $h = k = 1$, then $b = a$. But if $h = k = -1$, then $b = -a$ and so $a \neq b$.

This analysis suggests that you can find a counterexample by taking $b = -a$. Here is a formal answer:

Proposed Divisibility Property: For all integers a and b, if $a|b$ and $b|a$ then $a = b$.

Counterexample: Let $a = 2$ and $b = -2$. Then $-2 = (-1)\cdot 2$ and $2 = (-1)\cdot(-2)$, and thus

$$a|b \text{ and } b|a, \text{ but } a \neq b \text{ because } 2 \neq -2.$$

Therefore, the statement is false.

The Unique Factorization of Integers Theorem

The Unique Factorization of Integers Theorem (1/4)

The most comprehensive statement about divisibility of integers is contained in the *unique factorization of integers theorem*.

Because of its importance, this theorem is also called the *fundamental theorem of arithmetic*.

The unique factorization of integers theorem says that any integer greater than 1 either is prime or can be written as a product of prime numbers in a way that is unique except, perhaps, for the order in which the primes are written.

The Unique Factorization of Integers Theorem (2/4)

For example,

$$72 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 = 2 \cdot 3 \cdot 3 \cdot 2 \cdot 2 = 3 \cdot 2 \cdot 2 \cdot 3 \cdot 2,$$

and so forth.

The three 2's and two 3's may be written in any order, but any factorization of 72 as a product of primes must contain exactly three 2's and two 3's—no other collection of prime numbers besides three 2's and two 3's multiplies out to 72.

The Unique Factorization of Integers Theorem (3/4)

Theorem 4.4.5 Unique Factorization of Integers Theorem (Fundamental Theorem of Arithmetic)

Given any integer $n > 1$, there exist a positive integer k, distinct prime numbers p_1, $p_2, \ldots, p_k$, and positive integers $e_1, e_2, \ldots, e_k$ such that

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots p_k^{e_k},$$

and any other expression for n as a product of prime numbers is identical to this except, perhaps, for the order in which the factors are written.

The Unique Factorization of Integers Theorem (4/4)

Definition

Given any integer $n > 1$, the **standard factored form** of n is an expression of the form

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k},$$

where k is a positive integer, $p_1, p_2, \ldots, p_k$ are prime numbers, $e_1, e_2, \ldots, e_k$ are positive integers, and $p_1 < p_2 < \cdots < p_k$.

Example 4.4.8 – *Writing Integers in Standard Factored Form*

Write 3,300 in standard factored form.

Example 4.4.8 – *Solution*

First find all the factors of 3,300. Then write them in ascending order:

$$3{,}300 = 100 \cdot 33 = 4 \cdot 25 \cdot 3 \cdot 11$$

$$= 2 \cdot 2 \cdot 5 \cdot 5 \cdot 3 \cdot 11 = 2^2 \cdot 3^1 \cdot 5^2 \cdot 11^1.$$

Example 4.4.9 – *Using Unique Factorization to Solve a Problem*

Suppose m is an integer such that

$$8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot m = 17 \cdot 16 \cdot 15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 \cdot 10.$$

Does $17 | m$?

Example 4.4.9 – *Solution*

Since 17 is one of the prime factors of the right-hand side of the equation, it is also a prime factor of the left-hand side (by the unique factorization of integers theorem).

But 17 does not equal any prime factor of 8, 7, 6, 5, 4, 3, or 2 (because it is too large).

Hence 17 must occur as one of the prime factors of *m*, and so $17 \mid m$.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.5 Direct Proof and Counterexample V: Division into Cases and the Quotient-Remainder Theorem

Copyright © Cengage Learning. All rights reserved.

Direct Proof and Counterexample V: Division into Cases and the Quotient-Remainder Theorem (1/2)

Theorem 4.5.1 The Quotient-Remainder Theorem

Given any integer n and positive integer d, there exist unique integers q and r such that

$$n = dq + r \quad \text{and} \quad 0 \le r < d.$$

If n is positive, the quotient-remainder theorem can be illustrated on the number line as follows:

Direct Proof and Counterexample V: Division into Cases and the Quotient-Remainder Theorem (2/2)

If n is negative, the picture changes. Since $n = dq + r$, where r is nonnegative, d must be multiplied by a negative integer q to bring dq either exactly to n (in which case $r = 0$) or to a point below n (in which case the positive integer r is added to bring $dq + r$ back up to n).

This is illustrated as follows:

Example 4.5.1 – *The Quotient-Remainder Theorem*

For each of the following values of n and d, find integers q and r such that $n = dq + r$ and $0 \leq r < d$.

a. $n = 54$, $d = 4$

b. $n = -54$, $d = 4$

c. $n = 54$, $d = 70$

Example 4.5.1 – *Solution*

a. $54 = 4 \cdot 13 + 2$; hence $q = 13$ and $r = 2$.

b. $-54 = 4 \cdot (-14) + 2$; hence $q = -14$ and $r = 2$.

c. $54 = 70 \cdot 0 + 54$; hence $q = 0$ and $r = 54$.

div and mod

div and mod (1/1)

Definition

Given an integer n and a positive integer d,

$$\begin{aligned} \textbf{\textit{n div d}} &= \text{the integer quotient obtained} \\ &\quad\ \text{when } n \text{ is divided by } d\text{, and} \\ \textbf{\textit{n mod d}} &= \text{the nonnegative integer remainder obtained} \\ &\quad\ \text{when } n \text{ is divided by } d. \end{aligned}$$

Symbolically, if n and d are integers and $d > 0$, then

$$n \ div\ d = q \quad \text{and} \quad n \ mod\ d = r \quad \Leftrightarrow \quad n = dq + r,$$

where q and r are integers and $0 \leq r < d$.

Example 4.5.2 – *Computing div and mod by Hand or with a Four-Function Calculator*

Compute 32 *div* 9 and 32 *mod* 9 by hand or with a four-function calculator.

Example 4.5.2 – *Solution (1/2)*

Performing the division by hand gives the following results:

$$
\begin{array}{r}
3 \\
9\overline{)32} \\
\underline{27} \\
5
\end{array}
\quad
\begin{array}{l}
\leftarrow 32 \; div \; 9 \\
\\
\\
\leftarrow 32 \; mod \; 9
\end{array}
$$

To use a four-function calculator to compute *n div d* for a nonnegative integer *n* and a positive integer *d*, just divide *n* by *d* and ignore the part of the answer to the right of the decimal point.

Example 4.5.2 – *Solution (2/2)*

continued

To compute $n \bmod d$, substitute $n \operatorname{div} d$ in place of q and $n \bmod d$ in place of r in the equation $n = dq + r$. The result is

$$n = d \cdot (n \operatorname{div} d) + n \bmod d.$$

Solving for $n \bmod d$ gives

$$n \bmod d = n - d \cdot (n \operatorname{div} d).$$

Thus when you use a four-function calculator to divide 32 by 9, you obtain an expression like 3.555555556. Discarding the fractional part gives $32 \operatorname{div} 9 = 3$, and so

$$32 \bmod 9 = 32 - 9 \cdot (32 \operatorname{div} 9) = 32 - 27 = 5.$$

Example 4.5.4 – *Solving Problems about mod*

a. Prove that if n is a positive integer, then $n \bmod 10$ is the digit in the ones place in the decimal representation for n.

b. Suppose m is an integer. If $m \bmod 11 = 6$, what is $4m \bmod 11$?

Example 4.5.4 – *Solution (1/2)*

a. **Proof:** Suppose n is any positive integer. The decimal representation for n is $d_k\, d_{k-1} \ldots d_2\, d_1\, d_0$, where d_0, d_1, d_2, … , d_k are integers from 0 to 9 inclusive, $d_k \neq 0$ unless $n = 0$ and $k = 0$.

$$n = d_k \cdot 10^k + d_{k-1} \cdot 10^{k-1} + \cdots + d_2 \cdot 10^2 + d_1 \cdot 10 + d_0,$$

and d_0 is the digit in the ones place. Factoring out 10 from all but the final term gives

$$n = 10 \cdot (d_k \cdot 10^{k-1} + d_{k-1} \cdot 10^{k-2} + \cdots + d_2 \cdot 10^1 + d_1) + d_0.$$

Thus $n = 10 \cdot$(an integer) + d_0, and so $n \bmod 10 = d_0$, which is the digit in the ones place in the decimal representation for n.

Example 4.5.4 – *Solution (2/2)*

continued

b. Because $m \bmod 11 = 6$, the remainder obtained when m is divided by 11 is 6. This means that there is some integer q so that

$$m = 11q + 6.$$

Thus

$$4m = 44q + 24 = 44q + 22 + 2 = 11(4q + 2) + 2.$$

Since $4q + 2$ is an integer (because products and sums of integers are integers) and since $2 < 11$, the remainder obtained when $4m$ is divided by 11 is 2. Therefore,

$$4m \bmod 11 = 2.$$

Representations of Integers

Representations of Integers (1/4)

We defined an even integer to have the form twice some integer. At that time we could have defined an odd integer to be one that was not even.

Note The fact that any integer is either even or odd is called the **parity property**.

Example 4.5.5 – *Consecutive Integers Have Opposite Parity*

Prove that given any two consecutive integers, one is even and the other is odd.

Example 4.5.5 – *Solution (1/2)*

Two integers are called *consecutive* if, and only if, one is one more than the other. So if one integer is m, the next consecutive integer is $m + 1$.

To prove the given statement, you can divide the analysis into two cases: case 1, where the smaller of the two integers is even, and case 2, where the smaller of the two integers is odd.

Case 1 (m is even): In this case, $m = 2k$ for some integer k, and so $m + 1 = 2k + 1$, which is odd [*by definition of odd*]. Hence in this case, one of m and $m + 1$ is even and the other is odd.

Example 4.5.5 – *Solution (2/2)*

continued

Case 2 (m is odd): In this case, $m = 2k + 1$ for some integer k, and so $m + 1 = (2k + 1) + 1 = 2k + 2 = 2(k + 1)$. But $k + 1$ is an integer because it is a sum of two integers. Therefore, $m + 1$ equals twice some integer, and thus $m + 1$ is even. Hence in this case also, one of m and $m + 1$ is even and the other is odd.

It follows that regardless of which case actually occurs for the particular m and $m + 1$ that are chosen, one of m and $m + 1$ is even and the other is odd. [*This is what was to be shown.*]

Representations of Integers (2/4)

Theorem 4.5.2 The Parity Property

Any two consecutive integers have opposite parity.

Representations of Integers (3/4)

Method of Proof by Division into Cases

To prove a statement of the form "If A_1 or A_2 or . . . or A_n, then C," prove all of the following:

$$\begin{aligned} &\text{If } A_1, \text{ then } C, \\ &\text{If } A_2, \text{ then } C, \\ &\quad\vdots \\ &\text{If } A_n, \text{ then } C. \end{aligned}$$

This process shows that C is true regardless of which of $A_1, A_2, \ldots, A_n$ happens to be the case.

Example 4.5.6 – *Representing Integers mod 4*

Show that any integer can be written in one of the four forms

$$n = 4q \quad \text{or} \quad n = 4q + 1 \quad \text{or} \quad n = 4q + 2 \quad \text{or} \quad n = 4q + 3$$

for some integer q.

Example 4.5.6 – *Solution*

Given any integer n, apply the quotient-remainder theorem to n with the divisor equal to 4. This implies that there exist an integer quotient q and a remainder r such that

$$n = 4q + r \quad \text{and} \quad 0 \le r < 4.$$

But the only nonnegative remainders r that are less than 4 are 0, 1, 2, and 3.

Hence

$$n = 4q \quad \text{or} \quad n = 4q + 1 \quad \text{or} \quad n = 4q + 2 \quad \text{or} \quad n = 4q + 3$$

for some integer q. In other words, n *mod* 4 equals 0, 1, 2, or 3.

Example 4.5.7 – *The Square of an Odd Integer*

Prove: The square of any odd integer has the form $8m + 1$ for some integer m.

Example 4.5.7 – *Solution*

If checking some examples convinces you that the statement may be true, begin to develop a proof by asking, "Where am I starting from?" and "What do I need to show?" To help answer these questions, introduce variables to rewrite the statement more formally.

Formal Restatement: ∀ odd integer *n*, ∃ an integer *m* such that $n^2 = 8m + 1$.

From this, you can immediately identify the starting point and what is to be shown.

Example 4.5.7 – *Solution (1/3)* continued

Starting Point: Suppose *n* is a particular but arbitrarily chosen odd integer.

To Show: ∃ an integer *m* such that $n^2 = 8m + 1.$

This looks tough. Why should there be an integer *m* with the property that $n^2 = 8m + 1?$ That would say that $(n^2 - 1)/8$ is an integer, or that 8 divides $n^2 - 1.$ Perhaps you could make use of the fact that $n^2 - 1 = (n - 1)(n + 1).$

Example 4.5.7 – *Solution (2/3)*

continued

Does 8 divide $(n + 1)(n + 1)$? Since n is odd, both $(n - 1)$ and $(n + 1)$ are even. That means that their product is divisible by 4. But that's not enough. You need to show that the product is divisible by 8.

You could try another approach by arguing that since n is odd, you can represent it as $2q + 1$ for some integer q.

Then $n^2 = (2q + 1)^2 = 4q^2 + 4q + 1 = 4(q^2 + q) + 1.$ It is clear from this analysis that n^2 can be written in the form $4m + 1$, but it may not be clear that it can be written as $8m + 1$.

Example 4.5.7 – *Solution (3/3)* continued

Yet another possibility is to use the result of Example 4.5.6. That example showed that any integer can be written in one of the four forms $4q$, $4q + 1$, $4q + 2$, or $4q + 3$. Two of these, $4q + 1$ and $4q + 3$, are odd. Thus any odd integer can be written in the form $4q + 1$ or $4q + 3$ for some integer q. You could try breaking into cases based on these two different forms.

Representations of Integers (4/4)

Theorem 4.5.3

The square of any odd integer has the form $8m + 1$ for some integer m.

Absolute Value and the Triangle Inequality

Absolute Value and the Triangle Inequality (1/2)

The triangle inequality is one of the most important results involving absolute value. It has applications in many areas of mathematics.

Definition

For any real number x, the **absolute value of x**, denoted $|x|$, is defined as follows:

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0. \end{cases}$$

Absolute Value and the Triangle Inequality (2/2)

Lemma 4.5.4

For every real number r, $-|r| \leq r \leq |r|$.

Lemma 4.5.5

For every real number r, $|-r| = |r|$.

Theorem 4.5.6 The Triangle Inequality

For all real numbers x and y, $|x+y| \leq |x| + |y|$.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.6 Direct Proof and Counterexample VI: Floor and Ceiling

Copyright © Cengage Learning. All rights reserved.

Direct Proof and Counterexample VI: Floor and Ceiling (1/7)

Imagine a real number sitting on a number line. The *floor* and *ceiling* of the number are the integers to the immediate left and to the immediate right of the number (unless the number is an integer, in which case its floor and ceiling both equal the number itself).

Many computer languages have built-in functions that compute floor and ceiling automatically. These functions are very convenient to use when writing certain kinds of computer programs. In addition, the concepts of floor and ceiling are important in analyzing the efficiency of many computer algorithms.

Direct Proof and Counterexample VI: Floor and Ceiling (2/7)

Definition

Given any real number x, the **floor of x**, denoted $\lfloor x \rfloor$, is defined as follows:

$$\lfloor x \rfloor = \text{that unique integer } n \text{ such that } n \leq x < n + 1.$$

Symbolically, if x is a real number and n is an integer, then

$$\lfloor x \rfloor = n \quad \Leftrightarrow \quad n \leq x < n + 1.$$

Direct Proof and Counterexample VI: Floor and Ceiling (3/7)

Definition

Given any real number x, the **ceiling of x**, denoted $\lceil x \rceil$, is defined as follows:

$$\lceil x \rceil = \text{that unique integer } n \text{ such that } n - 1 < x \leq n.$$

Symbolically, if x is a real number and n is an integer, then

$$\lceil x \rceil = n \quad \Leftrightarrow \quad n - 1 < x \leq n.$$

Example 4.6.1 – *Computing Floors and Ceilings*

Compute $\lfloor x \rfloor$ and $\lceil x \rceil$ for each of the following values of x:

a. $25/4$ b. 0.999 c. -2.01

Note On some calculators $\lfloor x \rfloor$ is denoted INT (x).

Example 4.6.1 – *Solution*

a. $25/4 = 6.25$ and $6 < 6.25 < 7$; hence $\lfloor 25/4 \rfloor = 6$ and $\lceil 25/4 \rceil = 7$.

b. $0 < 0.999 < 1$; hence $\lfloor 0.999 \rfloor = 0$ and $\lceil 0.999 \rceil = 1$.

c. $-3 < -2.01 < -2$; hence $\lfloor -2.01 \rfloor = -3$ and $\lceil -2.01 \rceil = -2$.

Example 4.6.2 – *An Application*

The 1,370 students at a college are given the opportunity to take buses to an out-of-town event. Each bus holds a maximum of 40 passengers.

a. For reasons of economy, the leader of the event will send only full buses. What is the maximum number of buses the event leader will send?

b. If the event leader is willing to send one partially filled bus, how many buses will be needed to allow all the students to take the trip?

Example 4.6.2 – *Solution*

a. $\lfloor 1370/40 \rfloor = \lfloor 34.25 \rfloor = 34$

b. $\lceil 1370/40 \rceil = \lceil 34.25 \rceil = 35$

Example 4.6.4 – *Disproving an Alleged Property of Floor*

Is the following statement true or false?

For all real numbers x and y, $\lfloor x + y \rfloor = \lfloor x \rfloor + \lfloor y \rfloor$.

Example 4.6.4 – *Solution (1/4)*

The statement is false. As a counterexample, take $x = y = \frac{1}{2}$.

Then

$$\lfloor x \rfloor + \lfloor y \rfloor = \left\lfloor \frac{1}{2} \right\rfloor + \left\lfloor \frac{1}{2} \right\rfloor = 0 + 0 = 0,$$

whereas

$$\lfloor x + y \rfloor = \left\lfloor \frac{1}{2} + \frac{1}{2} \right\rfloor = \lfloor 1 \rfloor = 1.$$

Hence it is not always the case that $\lfloor x + y \rfloor \neq \lfloor x \rfloor + \lfloor y \rfloor$.

Example 4.6.4 – *Solution (2/4)* continued

To arrive at this counterexample, you could have reasoned as follows: Suppose x and y are real numbers.

Must it necessarily be the case that $\lfloor x + y \rfloor = \lfloor x \rfloor + \lfloor y \rfloor$, or could x and y be such that $\lfloor x + y \rfloor \neq \lfloor x \rfloor + \lfloor y \rfloor$?

Imagine values that the various quantities could take.

Example 4.6.4 – *Solution (3/4)* continued

For instance, if both x and y are positive, then $\lfloor x \rfloor$ and $\lfloor y \rfloor$ are the integer parts of $\lfloor x \rfloor$ and $\lfloor y \rfloor$ respectively; just as

$$2\frac{3}{5} = \underbrace{2}_{\text{integer part}} + \underbrace{\frac{3}{5}}_{\text{fractional part}}$$

so is $\quad x = \lfloor x \rfloor +$ fractional part of x

and $\quad y = \lfloor y \rfloor +$ fractional part of y,

where the term *fractional part* is understood here to mean the part of the number to the right of the decimal point when the number is written in decimal notation.

Example 4.6.4 – *Solution (4/4)* continued

Thus if x and y are positive,

$x + y = \lfloor x \rfloor + \lfloor y \rfloor +$ *the sum of the fractional parts of x and y*.

But also

$$x + y = \lfloor x + y \rfloor + \textit{the fractional part of } (x + y).$$

These equations show that if there exist numbers x and y such that the sum of the fractional parts of x and y is at least 1, then a counterexample can be found. As previously indicated, there do exist such x and y; for instance,

$x = \frac{1}{2}$ and $y = \frac{1}{2}$.

Example 4.6.5 – *Proving a Property of Floor*

Prove that for every real number x and for every integer m, $\lfloor x + m \rfloor = \lfloor x \rfloor + m$.

Example 4.6.5 – *Solution (1/2)*

Begin by supposing that x is a particular but arbitrarily chosen real number and that m is a particular but arbitrarily chosen integer.

You must show that $\lfloor x + m \rfloor = \lfloor x \rfloor + m$. Since this is an equation involving $\lfloor x \rfloor$ and $\lfloor x + m \rfloor$, it is reasonable to give one of these quantities a name: Let $n = \lfloor x \rfloor$.

By definition of floor,

n is an integer and $n \leq x < n + 1$.

Example 4.6.5 – *Solution (2/2)* continued

This double inequality enables you to compute the value of $\lfloor x + m \rfloor$ in terms of n by adding m to all sides:

$$n + m \leq x + m < n + m + 1.$$

Thus the left-hand side of the equation to be shown is

$$\lfloor x + m \rfloor = n + m.$$

On the other hand, since $n = \lfloor x \rfloor$, the right-hand side of the equation to be shown is

$$\lfloor x \rfloor + m = n + m$$

also. Thus $\lfloor x + m \rfloor = \lfloor x \rfloor + m$.

Direct Proof and Counterexample VI: Floor and Ceiling (4/7)

This discussion is summarized as follows:

Theorem 4.6.1

For every real number x and every integer m, $\lfloor x + m \rfloor = \lfloor x \rfloor + m$.

Theorem 4.6.2 The Floor of $n/2$

For any integer n,

$$\left\lfloor \frac{n}{2} \right\rfloor = \begin{cases} \dfrac{n}{2} & \text{if } n \text{ is even} \\ \dfrac{n-1}{2} & \text{if } n \text{ is odd.} \end{cases}$$

Direct Proof and Counterexample VI: Floor and Ceiling (5/7)

Given any integer *n* and any positive integer *d*, the quotient-remainder theorem guarantees the existence of unique integers *q* and *r* such that

$$n = dq + r \text{ and } 0 \leq r < d.$$

The following theorem states that the floor notation can be used to describe *q* and *r* as follows:

$$q = \left\lfloor \frac{n}{d} \right\rfloor \quad \text{and} \quad r = n - d \cdot \left\lfloor \frac{n}{d} \right\rfloor.$$

Direct Proof and Counterexample VI: Floor and Ceiling (6/7)

Thus if, on a calculator or in a computer language, floor is built in but *div* and *mod* are not, *div* and *mod* can be defined as follows: For a nonnegative integer *n* and a positive integer *d*,

$$n \; div \; d = \left\lfloor \frac{n}{d} \right\rfloor \quad \text{and} \quad n \; mod \; d = n - d \cdot \left\lfloor \frac{n}{d} \right\rfloor. \qquad 4.6.1$$

Note that *d* divides *n* if, and only if, *n mod d* = 0. In floor notation this means that *d* divides *n* if, and only if,

$n = d \cdot \lfloor n/d \rfloor$.

Direct Proof and Counterexample VI: Floor and Ceiling (7/7)

Theorem 4.6.3

If n is any integer and d is a positive integer, and if $q = \lfloor n/d \rfloor$ and $r = n - d \cdot \lfloor n/d \rfloor$, then

$$n = dq + r \quad \text{and} \quad 0 \leq r < d.$$

Example 4.6.6 – *Computing div and mod*

Use the floor notation to compute 3,850 *div* 17 and 3,850 *mod* 17.

Example 4.6.6 – *Solution*

By formula (4.6.1),

$$n \; div \; d = \left\lfloor \frac{n}{d} \right\rfloor \quad \text{and} \quad n \; mod \; d = n - d \cdot \left\lfloor \frac{n}{d} \right\rfloor.$$

$3{,}850 \; div \; 17 = \lfloor 3{,}850/17 \rfloor = \lfloor 226.4705882 \ldots \rfloor = 226$

$$\begin{aligned} 3{,}850 \; mod \; 17 &= 3{,}850 - 17 \cdot \lfloor 3{,}850/17 \rfloor \\ &= 3{,}850 - 17 \cdot 226 \\ &= 3{,}850 - 3{,}842 = 8. \end{aligned}$$

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.7 Indirect Argument: Contradiction and Contraposition

Copyright © Cengage Learning. All rights reserved.

Indirect Argument: Contradiction and Contraposition (1/2)

Indirect proofs are more roundabout. One kind of indirect proof, *argument by contradiction*, is based on the fact that either a statement is true or it is false but not both.

So if you can show that the assumption that a given statement is not true leads logically to a contradiction, impossibility, or absurdity, then that assumption must be false: and, hence, the given statement must be true.

This method of proof is also known as *reductio ad impossible* or *reductio ad absurdum* because it relies on reducing a given assumption to an impossibility or absurdity.

Indirect Argument: Contradiction and Contraposition (2/2)

Method of Proof by Contradiction

1. Suppose the statement to be proved is false. That is, suppose that the negation of the statement is true.
2. Show that this supposition leads logically to a contradiction.
3. Conclude that the statement to be proved is true.

Example 4.7.1 – *There Is No Greatest Integer*

Use proof by contradiction to show that there is no greatest integer.

Example 4.7.1 – *Solution (1/2)*

For this proof, the “certain property” is the property of being the greatest integer.

So prove that there is no object with this property, begin by supposing the negation: that there is an object with the property.

Starting Point: Suppose not. Suppose there is a greatest integer; call it N. This means that $N \geq n$ for every integer n.

Example 4.7.1 – *Solution (2/2)*

continued

To Show: This supposition leads logically to a contradiction.

Theorem 4.7.1

There is no greatest integer.

Example 4.7.2 – *No Integer Can Be Both Even and Odd*

The fact that no integer is both even and odd can also be deduced from the uniqueness part of the quotient-remainder theorem.

Theorem 4.7.2

There is no integer that is both even and odd.

Example 4.7.3 – *The Sum of a Rational Number and an Irrational Number*

Use proof by contradiction to show that the sum of any rational number and any irrational number is irrational.

Example 4.7.3 – *Solution (1/5)*

Begin by supposing the negation of what you are to prove. Be very careful when writing down what this means.

If you take the negation incorrectly, the entire rest of the proof will be flawed.

In this example, the statement to be proved can be written formally as

$\forall$ real numbers r and s, if r is rational and s is irrational, then $r + s$ is irrational.

Example 4.7.3 – *Solution (2/5)* continued

From this you can see that the negation is

$\exists$ a rational number r and an irrational number s such that $r + s$ is rational.

It follows that the starting point and what is to be shown are as follows:

Starting Point: Suppose not. That is, suppose there is a rational number r and an irrational number s such that $r + s$ is rational.

Example 4.7.3 – *Solution (3/5)* continued

To Show: This supposition leads to a contradiction.

To derive a contradiction, you need to understand what you are supposing: that there are numbers *r* and *s* such that *r* is rational, *s* is irrational, and *r* + *s* is rational. By definition of rational and irrational, this means there are convenient expressions that can be substituted for *r* and *r* + *s*, but all you can say about *s* is that it cannot be written as a quotient of integers

$$r = \frac{a}{b} \quad \text{for some integers } a \text{ and } b \text{ with } b \neq 0, \text{ and} \qquad 4.7.1$$

$$r + s = \frac{c}{d} \quad \text{for some integers } c \text{ and } d \text{ with } d \neq 0. \qquad 4.7.2$$

Example 4.7.3 – *Solution (4/5)* continued

If you substitute (4.7.1) into (4.7.2), you obtain

$$\frac{a}{b} + s = \frac{c}{d}.$$

Subtracting *a/b* from both sides gives

$$s = \frac{c}{d} - \frac{a}{b}$$

$$= \frac{bc}{bd} - \frac{ad}{bd} \quad \text{by rewriting } c/d \text{ and } a/b \text{ as equivalent fractions}$$

$$= \frac{bc - ad}{bd} \quad \text{by the rule for subtracting fractions with the same denominator.}$$

Example 4.7.3 – *Solution (5/5)* continued

Now both $bc - ad$ and bd are integers because products and differences of integers are integers, and $bd \neq 0$ by the zero product property.

Hence s can be expressed as a quotient of two integers with a nonzero denominator, and so s is rational, which contradicts the supposition that it is irrational.

Theorem 4.7.3

The sum of any rational number and any irrational number is irrational.

Argument by Contraposition

Argument by Contraposition (1/2)

A second form of indirect argument, *argument by contraposition*, is based on the logical equivalence between a statement and its contrapositive.

To prove a statement by contraposition, you take the contrapositive of the statement, prove the contrapositive by a direct proof, and conclude that the original statement is true.

Argument by Contraposition (2/2)

Method of Proof by Contraposition

1. Express the statement to be proved in the form

$$\forall x \text{ in } D, \text{ if } P(x) \text{ then } Q(x).$$

   (This step may be done mentally.)

2. Rewrite this statement in the contrapositive form

$$\forall x \text{ in } D, \text{ if } Q(x) \text{ is false then } P(x) \text{ is false.}$$

   (This step may also be done mentally.)

3. Prove the contrapositive by a direct proof.
   a. Suppose x is a (particular but arbitrarily chosen) element of D such that $Q(x)$ is false.
   b. Show that $P(x)$ is false.

Example 4.7.4 – *If the Square of an Integer Is Even, Then the Integer Is Even*

Prove that for every integer n, if n^2 is even then n is even.

Example 4.7.4 – *Solution (1/2)*

First form the contrapositive of the statement to be proved.

Contrapositive: For every integer *n*, if *n* is not even then n^2 is not even.

By the quotient-remainder theorem with divisor equal to 2, any integer is even or odd, and, by Theorem 4.7.2, no integer is both even and odd.

Theorem 4.7.2

There is no integer that is both even and odd.

Example 4.7.4 – *Solution (2/2)*

continued

So if an integer is not even, then it is odd. Thus the contrapositive can be restated as follows:

Contrapositive: For every integer *n*, if *n* is odd then n^2 is odd.

Proposition 4.7.4

For every integer n, if n^2 is even then n is even.

Relation between Proof by Contradiction and Proof by Contraposition

Relation between Proof by Contradiction and Proof by Contraposition (1/3)

Observe that any proof by contraposition can be recast in the language of proof by contradiction. In a proof by contraposition, the statement

$$\forall x \text{ in } D, \text{ if } P(x) \text{ then } Q(x)$$

is proved by giving a direct proof of the equivalent statement

$$\forall x \text{ in } D, \text{ if } \sim Q(x) \text{ then } \sim P(x).$$

To do this, you suppose you are given an arbitrary element x of D such that $\sim Q(x)$.

Relation between Proof by Contradiction and Proof by Contraposition (2/3)

You then show that $\sim P(x)$. This is illustrated in Figure 4.7.1.

$$\boxed{\text{Suppose } x \text{ is an arbitrary element of } D \text{ such that } \sim Q(x).} \xrightarrow{\text{sequence of steps}} \boxed{\sim P(x)}$$

Proof by Contraposition

Figure 4.7.1

To rewrite the proof as a proof by contradiction, you suppose there is an x in D such that $P(x)$ and $\sim Q(x)$.

You then follow the steps of the proof by contraposition to deduce the statement $\sim P(x)$.

Relation between Proof by Contradiction and Proof by Contraposition (3/3)

But $\sim P(x)$ is a contradiction to the supposition that $P(x)$ and $\sim Q(x)$. (Because to contradict a conjunction of two statements, it is only necessary to contradict one of them.) This process is illustrated in Figure 4.7.2.

Suppose $\exists x$ in D such that $P(x)$ and $\sim Q(x)$.	$\xrightarrow{\text{same sequence of steps}}$	Contradiction: $P(x)$ and $\sim P(x)$

Proof by Contradiction

Figure 4.7.2

Proposition 4.7.4

For every integer n, if n^2 is even then n is even.

Proof as a Problem-Solving Tool

Proof as a Problem-Solving Tool (1/3)

Direct proof, disproof by counterexample, proof by contradiction, and proof by contraposition are all tools that may be used to help determine whether statements are true or false.

Working with examples might have given you a sense that a statement of the form

For all elements in a domain, if (hypothesis) then (conclusion),

might be true. To explore further, imagine elements in the domain that satisfy the hypothesis.

Proof as a Problem-Solving Tool (2/3)

Ask yourself: Must they satisfy the conclusion? If you can see that the answer is “yes” in all cases, then the statement is true and your insight will form the basis for a direct proof.

If after some thought it is not clear that the answer is “yes,” ask yourself whether there are elements of the domain that satisfy the hypothesis and *not* the conclusion. If you are successful in finding some, then the statement is false and you have a counterexample.

On the other hand, if you are not successful in finding such elements, perhaps none exist.

Proof as a Problem-Solving Tool (3/3)

Perhaps you can show that assuming the existence of elements in the domain that satisfy the hypothesis and not the conclusion leads logically to a contradiction. If so, then the given statement is true and you have the basis for a proof by contradiction.

Alternatively, you could imagine elements of the domain for which the conclusion is false and ask whether such elements also fail to satisfy the hypothesis. If the answer in all cases is “yes,” then you have a basis for a proof by contraposition.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.8 Indirect Argument: Two Famous Theorems

Copyright © Cengage Learning. All rights reserved.

The Irrationality of $\sqrt{2}$ (1/4)

When mathematics flourished at the time of the ancient Greeks, mathematicians believed that given any two line segments, say *A*: _____ and *B*: _____ , a certain unit of length could be found so that segment *A* was exactly *m* units long and segment *B* was exactly *n* units long, for some integers *m* and *n*. (The segments were said to be *commensurable* with respect to this special unit of length.) Then the ratio of the lengths of *A* and *B* would be in the same proportion as the ratio of the integers *m* and *n*. Symbolically:

$$\frac{\text{length } A}{\text{length } B} = \frac{m}{n}.$$

The Irrationality of $\sqrt{2}$ (2/4)

Now it is easy to find a line segment of length $\sqrt{2}$; just take the diagonal of a unit square:

By the Pythagorean theorem, $c^2 = 1^2 + 1^2 = 2$, and so $c = \sqrt{2}$.

The Irrationality of $\sqrt{2}$ (3/4)

If the belief of the ancient Greeks were correct, there would be integers *m* and *n* such that

$$\frac{\text{length (diagonal)}}{\text{length (side)}} = \frac{m}{n}.$$

And this would imply that

$$\frac{c}{1} = \frac{\sqrt{2}}{1} = \sqrt{2} = \frac{m}{n}.$$

But then $\sqrt{2}$ would be a ratio of two integers, or, in other words, $\sqrt{2}$ would be rational.

The Irrationality of $\sqrt{2}$ (4/4)

Theorem 4.8.1 Irrationality of $\sqrt{2}$

$\sqrt{2}$ is irrational.

Example 4.8.1 – *Irrationality of* $1 + 3\sqrt{2}$

Prove by contradiction that $1 + 3\sqrt{2}$ is irrational.

Example 4.8.1 – *Solution*

The essence of the argument is the observation that if $1+3\sqrt{2}$ could be written as a ratio of integers, then so could $\sqrt{2}$. But by Theorem 4.8.1, we know that to be impossible.

The Irrationality of $\sqrt{2}$

Proposition 4.8.2

$1 + 3\sqrt{2}$ is irrational.

Are There Infinitely Many Prime Numbers?

Are There Infinitely Many Prime Numbers? (1/2)

You know that a prime number is a positive integer that cannot be factored as a product of two smaller positive integers. Is the set of all such numbers infinite, or is there a largest prime number? The answer was known to Euclid, and a proof that the set of all prime numbers is infinite.

Euclid’s proof requires one additional fact we have not yet established: If a prime number divides an integer, then it does not divide the next successive integer.

Are There Infinitely Many Prime Numbers? (2/2)

Proposition 4.8.3

For any integer a and any prime number p, if $p \mid a$ then $p \nmid (a + 1)$.

Theorem 4.8.4 Infinitude of the Primes

The set of prime numbers is infinite.

When to Use Indirect Proof

When to Use Indirect Proof (1/1)

The example in this section have not provided a definitive answer to the question of when to prove a statement directly and when to prove it indirectly. Many theorems can be proved either way. Usually, however, when both types of proof are possible, indirect proof is clumsier than direct proof.

In the absence of obvious clues suggesting indirect argument, try first to prove a statement directly. Then, if that does not succeed, look for a counterexample. If the search for a counterexample is unsuccessful, look for a proof by contradiction or contraposition.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.9 Application: The Handshake Theorem

Copyright © Cengage Learning. All rights reserved.

Application: The Handshake Theorem (1/3)

Definition

The total degree of a graph is the sum of the degrees of all the vertices of the graph.

Example 4.9.1 – *The Total Degree of a Graph*

Find the degree of each vertex of the graph *G* shown below, and then find the total degree of the graph.

Example 4.9.1 – *Solution*

$\deg(v_1) = 1$; $\deg(v_2) = 4$; $\deg(v_3) = 2$; $\deg(v_4) = 3$

total degree of $G = 1 + 4 + 2 + 3 = 10$

Application: The Handshake Theorem (2/3)

Theorem 4.9.1 The Handshake Theorem

If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G. Specifically, if the vertices of G are $v_1, v_2, \ldots, v_n$, where n is a nonnegative integer, then

$$\begin{aligned}\text{the total degree of } G &= \deg(v_1) + \deg(v_2) + \cdots + \deg(v_n)\\ &= 2 \cdot (\text{the number of edges of } G).\end{aligned}$$

Corollary 4.9.2

The total degree of a graph is even.

Example 4.9.2 – *Determining Whether Certain Graphs Exist*

Draw a graph with the specified properties or show that no such graph exists.

a. A graph with four vertices of degrees 1, 1, 2, and 3

b. A graph with four vertices of degree 1, 1, 3, and 3

Example 4.9.2 – *Solution (1/2)*

a. No such graph exists. By Corollary 4.9.2 the total degree of a graph is even. But a graph with four vertices of degrees 1, 1, 2, and 3 would have a total degree of 1 + 1 + 2 + 3 = 7, which is odd.

b. Let *G* be any of the graphs shown below.

Example 4.9.2 – *Solution (2/2)*

continued

In each case, no matter how the edges are labeled, deg(a) = 1, deg(b) = 1, deg(c) = 3, and deg(d) = 3.

Application: The Handshake Theorem (3/3)

Proposition 4.9.3

In any graph there is an even number of vertices of odd degree.

Example 4.9.4 – *Applying the Fact That the Number of Vertices with Odd Degree Is Even*

Is there a graph with ten vertices of degrees 1, 1, 2, 2, 2, 3, 4, 4, 4, and 6?

Example 4.9.4 – *Solution*

No. Such a graph would have three vertices of odd degree, which is impossible by Proposition 4.9.3.

Proposition 4.9.3

In any graph there is an even number of vertices of odd degree.

Note that this same result could have been deduced directly from Corollary 4.9.2 by computing the total degree ($1 + 1 + 2 + 2 + 2 + 3 + 4 + 4 + 4 + 6 = 29$) and noting that it is odd. However, Proposition 4.9.3 gives the result without the need to perform this addition.

Example 4.9.5 – *How Many People Attended the Meeting?*

At a meeting attended by a group of people, four people shook hands with one other person, six people shook hands with two other people, two people shook hands with three other people, and the rest shook hands with four other people. No two people shook hands with each other more than once, and a total of 23 handshakes occurred. How many people attended the meeting?

Example 4.9.5 – *Solution (1/4)*

Define a graph G by letting each vertex represent a person at the meeting and letting each edge represent one handshake between two people. Let x be the number of people who shook hands with four other people. Then

the total degree of the graph

$$= 4 \cdot 1 + 6 \cdot 2 + 2 \cdot 3 + x \cdot 4 = 22 + 4x$$

because 4 people shook hands with 1 other person, 6 people shook hands with 2 other people, 2 people shook hands with 3 other people, and x people shook hands with 4 other people.

Example 4.9.5 – *Solution (2/4)*

continued

In addition, since a total of 23 handshakes occurred, the graph has 23 edges. By the handshake theorem (Theorem 4.9.1), the total degree is twice the number of edges.

Theorem 4.9.1 The Handshake Theorem

If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G. Specifically, if the vertices of G are $v_1, v_2, \ldots, v_n$, where n is a nonnegative integer, then

$$\begin{aligned}\text{the total degree of } G &= \deg(v_1) + \deg(v_2) + \cdots + \deg(v_n)\\ &= 2 \cdot (\text{the number of edges of } G).\end{aligned}$$

Example 4.9.5 – *Solution (3/4)* continued

Hence

the total degree of the graph = 2 · 23 = 46.

It follows that

$$22 + 4x = 46.$$

Thus

$$4x = 46 - 22 = 24,$$

and so

$$x = \frac{24}{4} = 6.$$

Example 4.9.5 – *Solution (4/4)* continued

In other words, six people at the meeting shook hands with four other people. Now the total number of people at the meeting is the sum of the number who shook hands with one other person, plus the number who shook hands with two other people, plus the number who shook hands with three other people, plus the number who shook hands with four other people.

Therefore,

the number of people at the meeting = 4 + 6 + 2 + 6 = 18.

Special Graphs

Special Graphs (1/3)

Definition and Notation

A **simple graph** is a graph that does not have any loops or parallel edges. In a simple graph, an edge with endpoints v and w is denoted $\{v, w\}$.

Example 4.9.6 – *Some Simple Graphs*

Draw all simple graphs with the four vertices $\{u, v, w, x\}$ and two edges, one of which is $\{u, v\}$.

Example 4.9.6 – *Solution*

Since one edge of the graph is specified to be $\{u, v\}$, the possibilities for the other edge are $\{u, w\}$, $\{u, x\}$, $\{v, w\}$, $\{v, x\}$, and $\{w, x\}$.

The resulting graphs are shown below.

Special Graphs (2/3)

Definition

Let n be a positive integer. A **complete graph on n vertices**, denoted K_n, is a simple graph with n vertices and exactly one edge connecting each pair of distinct vertices.

Example 4.9.8 – *Complete Graphs on n Vertices: K_1, K_2, K_3, K_4, K_5*

The complete graphs K_1, K_2, K_3, K_4, and K_5 can be drawn as follows:

Example 4.9.9 – *The Number of Edges of* K_n

Prove that for any positive integer n, a complete graph on n vertices has $\frac{n(n-1)}{2}$ edges.

Example 4.9.9 – *Solution (1/3)*

Suppose n is a positive integer and K_n is a complete graph on n vertices. Because each vertex of K_n is connected by exactly one edge to each of the other $n - 1$ vertices of K_n, the degree of each vertex of K_n is $n - 1$.

Thus

$$\text{the total degree of } K_n = \begin{pmatrix}\text{the number of}\\ \text{vertices of } K_n\end{pmatrix} \cdot \begin{pmatrix}\text{the degree of}\\ \text{each vertex of } K_n\end{pmatrix} = n(n-1).$$

Example 4.9.9 – *Solution (2/3)*

continued

By the handshake theorem (Theorem 4.9.1), the total degree of K_n equals twice the number of its edges.

Theorem 4.9.1 The Handshake Theorem

If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G. Specifically, if the vertices of G are $v_1, v_2, \ldots, v_n$, where n is a nonnegative integer, then

$$\begin{aligned}\text{the total degree of } G &= \deg(v_1) + \deg(v_2) + \cdots + \deg(v_n)\\ &= 2 \cdot (\text{the number of edges of } G).\end{aligned}$$

So since the total degree is $n(n - 1)$,

$$n(n - 1) = 2 \cdot (\text{the number of edges of } K_n)$$

Example 4.9.9 – *Solution (3/3)*

continued

Dividing by 2 gives that

the number of edges of $K_n = \dfrac{n(n-1)}{2}$.

Special Graphs (3/3)

Definition

Let m and n be positive integers. A **complete bipartite graph on** (m, n) **vertices**, denoted $K_{m,n}$, is a simple graph whose vertices are divided into two distinct subsets, V with m vertices and W with n vertices, in such a way that

1. every vertex of $K_{m,n}$ belongs to one of V or W, but no vertex belongs to both V and W;
2. there is exactly one edge from each vertex of V to each vertex of W;
3. there is no edge from any one vertex of V to any other vertex of V;
4. there is no edge from any one vertex of W to any other vertex of W.

Example 4.9.10 – *Complete Bipartite Graphs: $K_{3,2}$ and $K_{3,3}$*

The complete bipartite graphs $K_{3,2}$ and $K_{3,3}$ are illustrated below.

CHAPTER 4

ELEMENTARY NUMBER THEORY AND METHODS OF PROOF

Copyright © Cengage Learning. All rights reserved.

4.10 Application: Algorithms

Copyright © Cengage Learning. All rights reserved.

Application: Algorithms

The word *algorithm* refers to a step-by-step method for performing some action.

Some examples of algorithms in everyday life are food preparation recipes, directions for assembling equipment or hobby kits, sewing pattern instructions, and instructions for filling out income tax forms.

Part of elementary school mathematics is devoted to learning algorithms for doing arithmetic such as multidigit addition and subtraction, multidigit (long) multiplication, and long division.

An Algorithmic Language

An Algorithmic Language (1/10)

The algorithmic language used in this book is a kind of pseudocode, combining elements of Python, C, C^{++}, and Java, and ordinary, but fairly precise, English.

We will use some of the formal constructs of computer languages—such as assignment statements, loops, and so forth—but we will ignore the more technical details, such as the requirement for explicit end-of-statement delimiters, the range of integer values available on a particular installation, and so forth.

An Algorithmic Language (2/10)

In high-level computer languages, the term **variable** is used to refer to a specific storage location in a computer's memory.

To say that the variable *x* has the value 3 means that the memory location corresponding to *x* contains the number 3.

A given storage location can hold only one value at a time. So if a variable is given a new value during program execution, then the old value is erased.

An Algorithmic Language (3/10)

The **data type** of a variable indicates the set in which the variable takes its values, whether the set of integers, or real numbers, or character strings, or the set {0, 1} (for a Boolean variable), and so forth.

An **assignment statement** gives a value to a variable. It has the form

$$x := e,$$

where *x* is a variable and *e* is an expression. This is read "*x* is assigned the value *e*" or "let *x* be *e*."

An Algorithmic Language (4/10)

When an assignment statement is executed, the expression *e* is evaluated (using the current values of all the variables in the expression), and then its value is placed in the memory location corresponding to *x* (replacing any previous contents of this location).

Ordinarily, algorithm statements are executed one after another in the order in which they are written. **Conditional statements** allow this natural order to be overridden by using the current values of program variables to determine which algorithm statement will be executed next.

An Algorithmic Language (5/10)

Conditional statements are denoted either

a. **if** (*condition*) or b. **if** (*condition*) **then** s_1

then s_1

else s_2

where *condition* is a predicate involving algorithm variables and where s_1 and s_2 are algorithm statements or groups of algorithm statements. We generally use indentation to indicate that statements belong together as a unit.

An Algorithmic Language (6/10)

When ambiguity is possible, however, we may explicitly bind a group of statements together into a unit by preceding the group with the word **do** and following it with the words **end do**.

Execution of an **if-then-else** statement occurs as follows:

1. The *condition* is evaluated by substituting the current values of all algorithm variables appearing in it and evaluating the truth or falsity of the resulting statement.
2. If *condition* is true, then s_1 is executed and execution moves to the next algorithm statement following the **if-then-else** statement.
3. If *condition* is false, then s_2 is executed and execution moves to the next algorithm statement following the **if-then-else** statement.

Example 4.10.1 – *Execution of if-then-else and if-then Statements*

Consider the following algorithm segments:

a. **if** $x > 2$
then $y := x + 1$
else do $x := x - 1$
$y := 3 \cdot x$ **end do**

b. $y := 0$
if $x > 2$ **then** $y := 2^x$

What is the value of *y* after execution of these segments for the following values of *x*?

i. $x = 5$

ii. $x = 2$

Example 4.10.1 – *Solution (1/3)*

a. (i) Because the value of x is 5 before execution, the guard condition $x > 2$ is true at the time it is evaluated.

Hence the statement following **then** is executed, and so the value of $x + 1 = 5 + 1$ is computed and placed in the storage location corresponding to y. So after execution, $y = 6$.

(ii) Because the value of x is 2 before execution, the guard condition $x > 2$ is false at the time it is evaluated.

Example 4.10.1 – *Solution (2/3)* continued

Hence the statement following **else** is executed. The value of $x - 1 = 2 - 1$ is computed and placed in the storage location corresponding to x, and the value of $3 \cdot x = 3 \cdot 1$ is computed and placed in the storage location corresponding to y. So after execution, $y = 3$.

b. (i) Since $x = 5$ initially, the condition $x > 2$ is true at the time it is evaluated. So the statement following **then** is executed, and y obtains the value $2^5 = 32$.

Example 4.10.1 – *Solution (3/3)* continued

(ii) Since $x = 2$ initially, the condition $x > 2$ is false at the time it is evaluated.

Execution, therefore, moves to the next statement following the if-then statement, and the value of y does not change from its initial value of 0.

An Algorithmic Language (7/10)

A **while** loop has the form

> **while** (*condition*)
> *[statements that make up*
> *the body of the loop]*
> **end while**

where *condition* is a predicate involving algorithm variables. The word **while** marks the beginning of the loop, and the words **end while** mark its end.

An Algorithmic Language (8/10)

Execution of a **while** loop occurs as follows:

1. The *condition* is evaluated by substituting the current values of all the algorithm variables and evaluating the truth or falsity of the resulting statement.
2. If *condition* is true, all the statements in the body of the loop are executed in order. Then execution moves back to the beginning of the loop and the process repeats.
3. If *condition* is false, execution passes to the next algorithm statement following the loop.

Example 4.10.2 – *Tracing Execution of a while Loop*

Trace the execution of the following algorithm segment by finding the values of all the algorithm variables each time they are changed during execution:

$i := 1, s := 0$

while $(i \leq 2)$

  $s := s + i$

  $i := i + 1$

end while

Example 4.10.2 – *Solution (1/3)*

Since i is given an initial value of 1, the condition $i \leq 2$ is true when the **while** loop is entered.

So the statements within the loop are executed in order:

$$s = 0 + 1 = 1 \qquad \text{and} \qquad i = 1 + 1 = 2.$$

Then execution passes back to the beginning of the loop.

The condition $i \leq 2$ is evaluated using the current value of i, which is 2.

Example 4.10.2 – *Solution (2/3)* continued

The condition is true, and so the statements within the loop are executed again:

$$s = 1 + 2 = 3 \quad \text{and} \quad i = 2 + 1 = 3.$$

Then execution passes back to the beginning of the loop.

The condition $i \leq 2$ is evaluated using the current value of i, which is 3.

This time the condition is false, and so execution passes beyond the loop to the next statement of the algorithm.

Example 4.10.2 – *Solution (3/3)* continued

This discussion can be summarized in a table, called a **trace table**, that shows the current values of algorithm variables at various points during execution. The trace table for a **while** loop generally gives all values immediately following each iteration of the loop. ("After the zeroth iteration" means the same as "before the first iteration.")

Trace Table

Variable Name	Iteration Number 0	1	2
i	1	2	3
s	0	1	3

An Algorithmic Language (9/10)

The second form of iteration we will use is a **for-next** loop. A **for-next** loop has the following form:

for *variable* := *initial expression* **to** *final expression*
[statements that make up
the body of the loop]
next (*same*) *variable*

An Algorithmic Language (10/10)

A **for-next** loop is executed as follows:

1. The **for-next** loop *variable* is set equal to the value of *initial expression*.
2. A check is made to determine whether the value of *variable* is less than or equal to the value of *final expression*.
3. If the value of *variable* is less than or equal to the value of *final expression*, then the statements in the body of the loop are executed in order, *variable* is increased by 1, and execution returns back to step 2.
4. If the value of *variable* is greater than the value of *final expression*, then execution passes to the next algorithm statement following the loop.

Example 4.10.3 – *Trace Table for a for-next Loop*

Convert the **for-next** loop shown below into a **while** loop. Construct a trace table for the loop.

for $i := 1$ **to** 4

$\quad x := i^2$

next i

Example 4.10.3 – *Solution (1/2)*

The given **for-next** loop is equivalent to the following:

$i := 1$

while $(i \leq 4)$

$\quad x := i^2$

$\quad i := i + 1$

end while

Example 4.10.3 – *Solution (2/2)* continued

Its trace table is as follows:

Trace Table

Variable Name	Iteration Number 0	1	2	3	4
x		1	4	9	16
i	1	2	3	4	5

A Notation for Algorithms

A Notation for Algorithms (1/3)

We will express algorithms as subroutines that can be called upon by other algorithms as needed and used to transform a set of input variables with given values into a set of output variables with specific values.

The output variables and their values are assumed to be returned to the calling algorithm. For example, the division algorithm specifies a procedure for taking any two positive integers as input and producing the quotient and remainder of the division of one number by the other as output.

A Notation for Algorithms (2/3)

Whenever an algorithm requires such a computation, the algorithm can just “call” the division algorithm to do the job.

We generally include the following information when describing algorithms formally:

1. The name of the algorithm, together with a list of input and output variables.

2. A brief description of how the algorithm works.

A Notation for Algorithms (3/3)

3. The input variable names, labeled by data type (whether integer, real number, and so forth).

4. The statements that make up the body of the algorithm, possibly with explanatory comments.

5. The output variable names, labeled by data type.

The Division Algorithm

The Division Algorithm (1/2)

For an integer a and a positive integer d, the quotient-remainder theorem guarantees the existence of integers q and r such that

$$a = dq + r \quad \text{and} \quad 0 \leq r < d.$$

The Division Algorithm (2/2)

Algorithm 4.10.1 Division Algorithm

[Given a nonnegative integer a and a positive integer d, the aim of the algorithm is to find integers q and r that satisfy the conditions $a = dq + r$ and $0 \le r < d$. This is done by subtracting d repeatedly from a until the result is less than d but is still nonnegative.

$$0 \le a - d - d - d - \cdots - d = a - dq < d.$$

The total number of d's that are subtracted is the quotient q. The quantity $a - dq$ equals the remainder r.]

Input: *a [a nonnegative integer], d [a positive integer]*

Algorithm Body:

$r := a$, $q := 0$

[Repeatedly subtract d from r until a number less than d is obtained. Add 1 to q each time d is subtracted.]

while $(r \ge d)$

    $r := r - d$

    $q := q + 1$

end while

[After execution of the **while** *loop, $a = dq + r$.]*

Output: *q, r [nonnegative integers]*

Example 4.10.4 – *Tracing the Division Algorithm*

Trace the action of Algorithm 4.10.1 on the input variables $a = 19$ and $d = 4$.

Example 4.10.4 – *Solution*

Make a trace table as shown below.

Variable Name	Iteration Number 0	1	2	3	4
a	19				
d	4				
r	19	15	11	7	3
q	0	1	2	3	4

The column under the *k*th iteration gives the states of the variables after the *k*th iteration of the loop.

The Euclidean Algorithm

The Euclidean Algorithm (1/4)

The greatest common divisor of two integers *a* and *b* is the largest integer that divides both *a* and *b*. For example, the greatest common divisor of 12 and 30 is 6. The Euclidean algorithm provides a very efficient way to compute the greatest common divisor of two integers.

Definition

Let a and b be integers that are not both zero. The **greatest common divisor** of a and b, denoted $\mathbf{gcd}(\boldsymbol{a}, \boldsymbol{b})$, is that integer d with the following properties:

1. d is a common divisor of both a and b. In other words,

$$d \mid a \quad \text{and} \quad d \mid b.$$

2. For every integer c, if c is a common divisor of both a and b, then c is less than or equal to d. In other words,

$$\text{for every integer } c, \text{ if } c \mid a \text{ and } c \mid b \text{ then } c \leq d.$$

Example 4.10.5 – *Calculating Some gcd's*

a. Find gcd(72, 63).

b. Find $\gcd(10^{20}, 6^{30})$.

c. In the definition of greatest common divisor, gcd(0, 0) is not allowed. Why not? What would gcd(0, 0) equal if it were found in the same way as the greatest common divisors for other pairs of numbers?

Example 4.10.5 – *Solution (1/2)*

a. 72 = 9 ⋅ 8 and 63 = 9 ⋅ 7. So $9|72 \text{ and } 9|63,$ and no integer larger than 9 divides both 72 and 63. Hence gcd(72, 63) = 9.

b. By the laws of exponents,

$$10^{20} = 2^{20} \cdot 5^{20} \text{ and } 6^{30} = 2^{30} \cdot 3^{30} = 2^{20} \cdot 2^{10} \cdot 3^{30}.$$

It follows that

$$2^{20}|10^{20} \quad \text{and} \quad 2^{20}|6^{30},$$

and by the unique factorization of integers theorem, no integer larger than 2^{20} Divides both $10^{20} \text{ and } 6^{30}$ (because no more than twenty 2's divide 10^{20}, no 3's divide 10^{20}, and no 5's divide 6^{30}).

Example 4.10.5 – *Solution (2/2)* continued

Hence $\gcd(10^{20}, 6^{30}) = 2^{20}$.

c. Suppose gcd(0, 0) were defined to be the largest common factor that divides 0 and 0. The problem is that *every* positive integer divides 0 and there is no largest integer.

So there is no largest common divisor!

The Euclidean Algorithm (2/4)

Lemma 4.10.1

If r is a positive integer, then $\gcd(r, 0) = r$.

Lemma 4.10.2

If a and b are any integers not both zero, and if q and r are any integers such that

$$a = bq + r,$$

then

$$\gcd(a, b) = \gcd(b, r).$$

The Euclidean Algorithm (3/4)

Euclidean Algorithm Description

1. Let A and B be integers with $A > B \geq 0$.
2. To find the greatest common divisor of A and B, first check whether $B = 0$. If it is, then $\gcd(A, B) = A$ by Lemma 4.10.1. If it isn't, then $B > 0$ and the quotient-remainder theorem can be used to divide A by B to obtain a quotient q and a remainder r:

$$A = Bq + r \quad \text{where } 0 \leq r < B.$$

By Lemma 4.10.2, $\gcd(A, B) = \gcd(B, r)$. Thus the problem of finding the greatest common divisor of A and B is reduced to the problem of finding the greatest common divisor of B and r.

[What makes this information useful is the fact that the larger number of the pair (B, r) is smaller than the larger number of the pair (A, B). The reason is that the value of r found by the quotient-remainder theorem satisfies

$$0 \leq r < B.$$

And, since by assumption $B < A$, we have that

$$0 \leq r < B < A.]$$

3. Now just repeat the process, starting again at (2), but use B instead of A and r instead of B. The repetitions are guaranteed to terminate eventually with $r = 0$ because each new remainder is less than the preceding one and all are nonnegative.

Example 4.10.6 – *Hand-Calculation of gcd's Using the Euclidean Algorithm*

Use the Euclidean algorithm to find gcd(330, 156).

Example 4.10.6 – *Solution (1/5)*

1. Divide 330 by 156:

Thus 330 = 156 · 2 + 18, and hence gcd(330, 156) = gcd(156, 18) by Lemma 4.10.2.

$$\begin{array}{r} 2 \\ 156\overline{)\,330} \\ 312 \\ \hline 18 \end{array} \begin{array}{l} \leftarrow \text{quotient} \\ \\ \\ \\ \leftarrow \text{remainder} \end{array}$$

Lemma 4.10.2

If a and b are any integers not both zero, and if q and r are any integers such that

$$a = bq + r,$$

then

$$\gcd(a, b) = \gcd(b, r).$$

Example 4.10.6 – *Solution (2/5)* continued

2. Divide 156 by 18:

$$
\begin{array}{r}
8 \leftarrow \text{quotient} \\
18\overline{\big)\,156} \\
\underline{144} \\
12 \leftarrow \text{remainder}
\end{array}
$$

Thus $156 = 18 \cdot 8 + 12$, and hence $\gcd(156, 18) = \gcd(18, 12)$ by Lemma 4.10.2.

Example 4.10.6 – *Solution (3/5)* continued

3. Divide 18 by 12:

Thus 18 = 12 · 1 + 6, and hence gcd(18, 12) = gcd(12, 6) by Lemma 4.10.2.

$$\begin{array}{r} 1 \leftarrow \text{quotient} \\ 12\overline{)\,18} \\ 12 \\ \hline 6 \leftarrow \text{remainder} \end{array}$$

Lemma 4.10.2

If a and b are any integers not both zero, and if q and r are any integers such that

$$a = bq + r,$$

then

$$\gcd(a, b) = \gcd(b, r).$$

Example 4.10.6 – *Solution (4/5)* continued

4. Divide 12 by 6:

$$
\begin{array}{r}
2 \\
6\overline{\big)\,12} \\
\underline{12} \\
0
\end{array}
\begin{array}{l}
\leftarrow \text{quotient} \\
\\
\\
\leftarrow \text{remainder}
\end{array}
$$

Thus $12 = 6 \cdot 2 + 0$, and hence $\gcd(12, 6) = \gcd(6, 0)$ by Lemma 4.10.2. Putting all the equations above together gives

$$
\begin{aligned}
\gcd(330, 156) &= \gcd(156, 18) \\
&= \gcd(18, 12)
\end{aligned}
$$

Example 4.10.6 – *Solution (5/5)* continued

$= \gcd(12, 6)$

$= \gcd(6, 0)$

$= 6$ by Lemma 4.10.1.

Lemma 4.10.1

If r is a positive integer, then $\gcd(r, 0) = r$.

Therefore, $\gcd(330, 156) = 6$.

The Euclidean Algorithm (4/4)

The following is a version of the Euclidean algorithm written using formal algorithm notation.

Algorithm 4.10.2 Euclidean Algorithm

[Given two integers A and B with $A > B \geq 0$, this algorithm computes $\gcd(A, B)$*. It is based on two facts:*

1. $\gcd(a, b) = \gcd(b, r)$ *if a, b, q, and r are integers with $a = b \cdot q + r$ and $0 \leq r < b$.*
2. $\gcd(a, 0) = a$*.]*

Input: A, B *[integers with $A > B \geq 0$]*

Algorithm Body:

$a := A, b := B, r := B$
[If $b \neq 0$, compute a mod b, the remainder of the integer division of a by b, and set r equal to this value. Then repeat the process using b in place of a and r in place of b.]
while $(b \neq 0)$
  $r := a \bmod b$
[The value of a mod b can be obtained by calling the division algorithm.]
  $a := b$
  $b := r$
end while
[After execution of the **while** *loop,* $\gcd(A, B) = a$*.]*
$\gcd := a$

Output: gcd *[a positive integer]*

Example 4.10.7 – *A Trace Table for the Euclidean Algorithm*

Construct a trace table for Algorithm 4.10.2 using $A = 330$ and $B = 156$, the same numbers as in Example 4.10.6.

Example 4.10.7 – *Solution*

A	330				
B	156				
a	330	156	18	12	6
b	156	18	12	6	0
r	156	18	12	6	0
gcd					6

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.1 Sequences

Copyright © Cengage Learning. All rights reserved.

Sequences (1/10)

Imagine that a person decides to count his ancestors. He has two parents, four grandparents, eight great-grandparents, and so forth. These numbers can be written in a row as

2, 4, 8, 16, 32, 64, 128, …

To express the pattern of the numbers, suppose that each is labeled by an integer giving its position in the row.

Position in the row	1	2	3	4	5	6	7 . . .
Number of ancestors	2	4	8	16	32	64	128 . . .

Sequences (2/10)

The number corresponding to position 1 is 2, which equals 2^1. The number corresponding to position 2 is 4, which equals 2^2. For positions 3, 4, 5, 6, and 7, the corresponding numbers are 8, 16, 32, 64, and 128, which equal 2^3, 2^4, 2^5, 2^6, and 2^7, respectively.

For a general value of *k*, let A_k be the number of ancestors in the *k*th generation back. The pattern of computed values strongly suggests the following for each *k*:

$$A_k = 2^k.$$

Sequences (3/10)

Definition

A **sequence** is a function whose domain is either all the integers between two given integers or all the integers greater than or equal to a given integer.

We typically represent a sequence as a set of elements written in a row. In the sequence denoted

$$a_m, a_{m+1}, a_{m+2}, \ldots, a_n,$$

each individual element a_k (read "*a* sub *k*") is called a **term**.

Sequences (4/10)

The k in a_k is called a **subscript** or **index**, m (which may be any integer) is the subscript of the **initial term**, and n (which must be an integer that is greater than or equal to m) is the subscript of the **final term**.

The notation

$$a_m, a_{m+1}, a_{m+2}, \ldots$$

denotes an **infinite sequence**. An **explicit formula** or **general formula** for a sequence is a rule that shows how the values of a_k depend on k.

Example 5.1.1 – *Finding Terms of Sequences Given by Explicit Formulas*

Define sequences $a_1, a_2, a_3, \ldots$ and $b_2, b_3, b_4, \ldots$ by the following explicit formulas:

$$a_k = \frac{k}{k+1} \quad \text{for every integer } k \geq 1,$$

$$b_i = \frac{i-1}{i} \quad \text{for every integer } i \geq 2.$$

Compute the first five terms of both sequences.

Example 5.1.1 – *Solution (1/2)*

$$a_1 = \frac{1}{1+1} = \frac{1}{2} \qquad b_2 = \frac{2-1}{2} = \frac{1}{2}$$

$$a_2 = \frac{2}{2+1} = \frac{2}{3} \qquad b_3 = \frac{3-1}{3} = \frac{2}{3}$$

$$a_3 = \frac{3}{3+1} = \frac{3}{4} \qquad b_4 = \frac{4-1}{4} = \frac{3}{4}$$

$$a_4 = \frac{4}{4+1} = \frac{4}{5} \qquad b_5 = \frac{5-1}{5} = \frac{4}{5}$$

$$a_5 = \frac{5}{5+1} = \frac{5}{6} \qquad b_6 = \frac{6-1}{6} = \frac{5}{6}$$

Example 5.1.1 – *Solution (2/2)* continued

As you can see, the first terms of both sequences are $\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6};$ in fact, it can be shown that all terms of both sequences are identical.

Example 5.1.2 – *An Alternating Sequence*

Compute the first six terms of the sequence c_0, c_1, c_2, … defined as follows:

$$c_j = (-1)^j \quad \text{for every integer } j \geq 0.$$

Example 5.1.2 – *Solution*

$$c_0 = (-1)^0 = 1$$
$$c_1 = (-1)^1 = -1$$
$$c_2 = (-1)^2 = 1$$
$$c_3 = (-1)^3 = -1$$
$$c_4 = (-1)^4 = 1$$
$$c_5 = (-1)^5 = -1$$

Thus the first six terms are 1, −1, 1, −1, 1, −1. Even powers of −1 equal 1 and odd powers of −1 equal −1. It follows that the sequence oscillates endlessly between 1 and −1.

Example 5.1.3 – *Finding an Explicit Formula to Fit Given Initial Terms*

Find an explicit formula for a sequence with the following initial terms:

$$1, \quad -\frac{1}{4}, \quad \frac{1}{9}, \quad -\frac{1}{16}, \quad \frac{1}{25}, \quad -\frac{1}{36}, \ldots.$$

Example 5.1.3 – *Solution (1/3)*

Denote the general term of the sequence by a_k and suppose the first term is a_1. Next observe that the denominator of each term is a perfect square. Thus the terms can be rewritten as

$$\begin{array}{cccccc} \dfrac{1}{1^2}, & \dfrac{(-1)}{2^2}, & \dfrac{1}{3^2}, & \dfrac{(-1)}{4^2}, & \dfrac{1}{5^2}, & \dfrac{(-1)}{6^2}. \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{array}$$

Now note that the denominator of each term equals the square of the subscript of that term, and that the numerator equals ±1.

Example 5.1.3 – *Solution (2/3)* continued

Hence

$$a_k = \frac{\pm 1}{k^2}.$$

Also the numerator oscillates back and forth between +1 and −1; it is +1 when *k* is odd and −1 when *k* is even.

To achieve this oscillation, insert a factor of $(-1)^{k+1}$ (or $(-1)^{k-1}$) into the formula for a_k. *[For when k is odd, k* + 1 *is even and thus* $(-1)^{k+1} = +1;$ *and when k is even, k* + 1 *is odd and thus* $(-1)^{k+1} = -1.$*]*

Example 5.1.3 – *Solution (3/3)* continued

Consequently, an explicit formula that gives the correct first six terms is

$$a_k = \frac{(-1)^{k+1}}{k^2} \quad \text{for every integer } k \geq 1.$$

Note that making the first term a_0 would have led to the alternative formula

$$a_k = \frac{(-1)^k}{(k+1)^2} \quad \text{for every integer } k \geq 0.$$

You should check that this formula also gives the correct first six terms. It is convenient to use a shorthand notation to write such sums.

Sequences (5/10)

It is convenient to use a shorthand notation to write such sums. In 1772 the French mathematician Joseph Louis Lagrange introduced the capital Greek letter sigma, Σ, to denote the word *sum* (or *summation*), and defined the summation notation as follows:

Definition

If m and n are integers and $m \leq n$, the symbol $\sum_{k=m}^{n} a_k$, read the **summation from k equals m to n of a-sub-k**, is the sum of all the terms $a_m, a_{m+1}, a_{m+2}, \ldots, a_n$. We say that $a_m + a_{m+1} + a_{m+2} + \cdots + a_n$ is the **expanded form** of the sum, and we write

$$\sum_{k=m}^{n} a_k = a_m + a_{m+1} + a_{m+2} + \cdots + a_n.$$

We call k the **index** of the summation, m the **lower limit** of the summation, and n the **upper limit** of the summation.

Example 5.1.4 – *Computing Summations*

Let $a_1 = -2$, $a_2 = -1$, $a_3 = 0$, $a_4 = 1$, and $a_5 = 2$. Compute the following:

a. $\sum_{k=1}^{5} a_k$

b. $\sum_{k=2}^{2} a_k$

c. $\sum_{k=1}^{2} a_{2k}$

Example 5.1.4 – *Solution*

a. $\sum_{k=1}^{5} a_k = a_1 + a_2 + a_3 + a_4 + a_5 = (-2) + (-1) + 0 + 1 + 2 = 0$

b. $\sum_{k=2}^{2} a_k = a_2 = -1$

c. $\sum_{k=1}^{2} a_{2k} = a_{2\cdot 1} + a_{2\cdot 2} = a_2 + a_4 = -1 + 1 = 0$

Sequences (6/10)

The terms of a summation are expressed using an explicit formula. For instance, it is common to see summations such as

$$\sum_{k=1}^{5} k^2 \quad \text{or} \quad \sum_{i=0}^{8} \frac{(-1)^i}{i+1}.$$

Example 5.1.5 – *When the Terms of a Summation Are Given by a Formula*

Compute

$$\sum_{k=1}^{5} k^2.$$

Example 5.1.5 – *Solution*

$$\sum_{k=1}^{5} k^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55.$$

Sequences (7/10)

When the upper limit of a summation is a variable, an ellipsis is used to write the summation in expanded form.

Example 5.1.6 – *Changing from Summation Notation to Expanded Form*

Write $\sum_{i=0}^{n} \frac{(-1)^i}{i+1}$ in expanded form:

Example 5.1.6 – *Solution*

$$\sum_{i=0}^{n} \frac{(-1)^i}{i+1} = \frac{(-1)^0}{0+1} + \frac{(-1)^1}{1+1} + \frac{(-1)^2}{2+1} + \frac{(-1)^3}{3+1} + \cdots + \frac{(-1)^n}{n+1}$$

$$= \frac{1}{1} + \frac{(-1)}{2} + \frac{1}{3} + \frac{(-1)}{4} + \cdots + \frac{(-1)^n}{n+1}$$

$$= 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{(-1)^n}{n+1}$$

Example 5.1.7 – *Changing from Expanded Form to Summation Notation*

Express the following using summation notation:

$$\frac{1}{n}+\frac{2}{n+1}+\frac{3}{n+2}+\cdots+\frac{n+1}{2n}.$$

Example 5.1.7 – *Solution*

The general term of this summation can be expressed as $\frac{i+1}{n+i}$ for each integer *i* from 0 to *n*.

Hence

$$\frac{1}{n}+\frac{2}{n+1}+\frac{3}{n+2}+\cdots+\frac{n+1}{2n}=\sum_{i=0}^{n}\frac{i+1}{n+i}.$$

Sequences (8/10)

The top index *n* of the summation is a *free variable* because it may be replaced by any integer greater than or equal to the bottom index, and each such replacement leads to a different summation.

For any particular summation the top index acts like a constant. Thus when the top index also appears in the terms of the summation, its value does not change from term to term.

Sequences (9/10)

By contrast, the index variable in these examples is bound by the summation symbol. It must take every value from the bottom limit to the top limit in succession.

Writing a summation in expanded form helps relate it to our previous experience of working with sums. But for small values of *n* the expanded form may be misleading.

Example 5.1.8 – *Evaluating* $a_1 + a_2 + a_3 + \ldots + a_n$ *for Small n*

What is the value of $2^0 + 2^1 + 2^2 + \cdots + 2^n$ when $n = 0$, $n = 1$, and $n = 2$?

Example 5.1.8 – *Solution (1/2)*

When you evaluate a summation like $2^0 + 2^1 + 2^2 + \cdots + 2^n$ for small values of *n*, you can avoid a mistake by imagining it in summation notation.

For instance,

$$2^0 + 2^1 + 2^2 + \cdots + 2^n = \sum_{i=0}^{n} 2^i.$$

So when $n = 0$, $2^0 + 2^1 + 2^2 + \cdots + 2^n$ has the value $\sum_{i=0}^{0} 2^i = 2^0 = 1.$

Example 5.1.8 – *Solution (2/2)*

continued

When $n = 1$, $2^0 + 2^1 + 2^2 + \cdots + 2^n$ has the value

$$\sum_{i=0}^{1} 2^i = 2^0 + 2^1 = 1 + 2.$$

When $n = 2$, $2^0 + 2^1 + 2^2 + \cdots + 2^n$ has the value

$$\sum_{i=0}^{2} 2^i = 2^0 + 2^1 + 2^2 = 1 + 2 + 4.$$

Example 5.1.9 – *Using a Single Summation Sign and Separating Off a Final Term*

a. Write $\sum_{k=0}^{n} 2^k + 2^{n+1}$ as a single summation.

b. Rewrite $\sum_{i=1}^{n+1} \frac{1}{i^2}$ by separating off the final term.

Example 5.1.9 – *Solution*

a. $$\sum_{k=0}^{n} 2^k + 2^{k+1} = (2^0 + 2^1 + 2^2 + \cdots + 2^n) + 2^{n+1} = \sum_{k=0}^{n+1} 2^k$$

b. $$\sum_{i=1}^{n+1} \frac{1}{i^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2} + \frac{1}{(n+1)^2} = \sum_{i=1}^{n} \frac{1}{i^2} + \frac{1}{(n+1)^2}$$

Sequences (10/10)

In certain sums each term is a difference of two quantities. When you write such sums in expanded form, you sometimes see that all the terms cancel except the first and the last.

Example 5.1.10 – *A Telescoping Sum*

Some sums can be transformed so that successive cancellation of terms collapses the final result like a telescope. For instance, observe that for every integer $k \geq 1$,

$$\frac{1}{k} - \frac{1}{k+1} = \frac{(k+1)-k}{k(k+1)} = \frac{1}{k(k+1)}.$$

Use this identity to find a simple expression for $\sum_{k=1}^{n} \frac{1}{k(k+1)}$.

Example 5.1.10 – *Solution*

$$\sum_{k=1}^{n} \frac{1}{k(k+1)} = \sum_{k=1}^{n} \left(\frac{1}{k} - \frac{1}{k+1} \right)$$

$$= \left(\frac{1}{1} - \cancel{\frac{1}{2}} \right) + \left(\cancel{\frac{1}{2}} - \cancel{\frac{1}{3}} \right) + \left(\cancel{\frac{1}{3}} - \cancel{\frac{1}{4}} \right) + \cdots + \left(\cancel{\frac{1}{n-1}} - \cancel{\frac{1}{n}} \right) + \left(\cancel{\frac{1}{n}} - \frac{1}{n+1} \right)$$

$$= 1 - \frac{1}{n+1}$$

Product Notation

Product Notation (1/2)

The notation for the product of a sequence of numbers is analogous to the notation for their sum. The Greek capital letter pi, Π, denotes a product. For example,

$$\prod_{k=1}^{5} a_k = a_1 a_2 a_3 a_4 a_5.$$

Definition

If m and n are integers and $m \leq n$, the symbol $\prod_{k=m}^{n} a_k$, read the **product from k equals m to n of a-sub-k**, is the product of all the terms $a_m, a_{m+1}, a_{m+2}, \ldots, a_n$. We write

$$\prod_{k=m}^{n} a_k = a_m \cdot a_{m+1} \cdot a_{m+2} \cdots a_n.$$

Product Notation (2/2)

A recursive definition for the product notation is the following: If *m* is any integer, then

$$\prod_{k=m}^{m} a_k = a_m \quad \text{and} \quad \prod_{k=m}^{n} a_k = \left(\prod_{k=m}^{n-1} a_k\right) \cdot a_n \quad \text{for every integer } n > m.$$

Example 5.1.11 – *Computing Products*

Compute the following products:

a. $\prod_{k=1}^{5} k$

b. $\prod_{k=1}^{1} \frac{k}{k+1}$

Example 5.1.11 – *Solution*

a. $\prod_{k=1}^{5} k = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$

b. $\prod_{k=1}^{1} \frac{k}{k+1} = \frac{1}{1+1} = \frac{1}{2}$

Properties of Summations and Products

Properties of Summations and Products (1/1)

The following theorem states general properties of summations and products.

Theorem 5.1.1

If $a_m, a_{m+1}, a_{m+2}, \ldots$ and $b_m, b_{m+1}, b_{m+2}, \ldots$ are sequences of real numbers and c is any real number, then the following equations hold for any integer $n \geq m$:

1. $\sum_{k=m}^{n} a_k + \sum_{k=m}^{n} b_k = \sum_{k=m}^{n} (a_k + b_k)$
2. $c \cdot \sum_{k=m}^{n} a_k = \sum_{k=m}^{n} c \cdot a_k$ generalized distributive law
3. $\left(\prod_{k=m}^{n} a_k\right) \cdot \left(\prod_{k=m}^{n} b_k\right) = \prod_{k=m}^{n} (a_k \cdot b_k).$

Example 5.1.12 – *Using Properties of Summation and Product*

Let $a_k = k + 1$ and $b_k = k - 1$ for every integer k. Write each of the following expressions as a single summation or product:

a. $\sum_{k=m}^{n} a_k + 2 \cdot \sum_{k=m}^{n} b_k$

b. $\left(\prod_{k=m}^{n} a_k\right) \cdot \left(\prod_{k=m}^{n} b_k\right)$

Example 5.1.12 – *Solution (1/2)*

a. $$\sum_{k=m}^{n} a_k + 2 \cdot \sum_{k=m}^{n} b_k = \sum_{k=m}^{n} (k+1) + 2 \cdot \sum_{k=m}^{n} (k-1)$$ by substitution

$$= \sum_{k=m}^{n} (k+1) + \sum_{k=m}^{n} 2 \cdot (k-1)$$ by Theorem 5.1.1 (2)

$$= \sum_{k=m}^{n} ((k+1) + 2 \cdot (k-1))$$ by Theorem 5.1.1 (1)

$$= \sum_{k=m}^{n} (3k-1)$$ by algebraic simplification

Example 5.1.12 – *Solution (2/2)* continued

b. $$\left(\prod_{k=m}^{n} a_k\right)\cdot\left(\prod_{k=m}^{n} b_k\right) = \left(\prod_{k=m}^{n}(k+1)\right)\cdot\left(\prod_{k=m}^{n}(k-1)\right)$$ by substitution

$$= \prod_{k=m}^{n}((k+1)\cdot(k-1))$$ by Theorem 5.1.1 (3)

$$= \prod_{k=m}^{n}(k^2-1)$$ by algebraic simplification

Change of Variable

Change of Variable (1/3)

Observe that

$$\sum_{k=1}^{3} k^2 = 1^2 + 2^2 + 3^2$$

and also that

$$\sum_{i=1}^{3} i^2 = 1^2 + 2^2 + 3^2.$$

Hence

$$\sum_{k=1}^{3} k^2 = \sum_{i=1}^{3} i^2.$$

Change of Variable (2/3)

The symbol used to represent an index of a summation is an example of a local variable, often called a **dummy variable**, because, it can be replaced by any other symbol as long as the replacement is made in each location where it occurs.

Outside of that context (both before and after), the symbol may have another meaning entirely. In the same way, a symbol used to represent a variable in a universally or existentially quantified state can be replaced by any other symbol as long as the replacements are made consistently.

Change of Variable (3/3)

The appearance of a summation can be altered by more complicated changes of variable as well. For example, observe that

$$\sum_{j=2}^{4}(j-1)^2 = (2-1)^2 + (3-1)^2 + (4-1)^2$$

$$= 1^2 + 2^2 + 3^2$$

$$= \sum_{k=1}^{3} k^2.$$

Example 5.1.13 – *Transforming a Sum by a Change of Variable*

Transform the following summation by making the specified change of variable:

$$\textit{summation: } \sum_{k=0}^{6} \frac{1}{k+1} \qquad \textit{change of variable: } j = k+1$$

Example 5.1.13 – *Solution (1/2)*

First calculate the lower and upper limits of the new summation:

When $k = 0$, $j = k + 1 = 0 + 1 = 1$.

When $k = 6$, $j = k + 1 = 6 + 1 = 7$.

Thus the new sum goes from $j = 1$ to $j = 7$.

Example 5.1.13 – *Solution (2/2)* continued

Next calculate the general term of the new summation. You will need to replace each occurrence of *k* by an expression in *j*:

Since $j = k + 1$, then $k = j - 1$.

$$\text{Hence } \frac{1}{k+1} = \frac{1}{(j-1)+1} = \frac{1}{j}.$$

Finally, put the steps together to obtain

$$\sum_{k=0}^{6} \frac{1}{k+1} = \sum_{j=1}^{7} \frac{1}{j}. \qquad 5.1.1$$

Example 5.1.14 – *When the Upper Limit Appears in the Expression to Be Summed*

Rewrite the summation $\sum_{k=1}^{n+1}\left(\frac{k}{n+k}\right)$ so that the lower limit becomes 0 and the upper limit becomes n but the index of the summation remains k.

a. First, transform the summation by making the change of variable $j = k - 1$.

b. Second, transform the summation obtained in part (a) by changing all j's to k's.

Example 5.1.14 – *Solution (1/3)*

a. The index variable k is bound by the summation symbol to take each of the values from 1 to $n + 1$ in succession.

When $k = 1$, then $j = 1 - 1 = 0$, and when $k = n + 1$, then $j = (n + 1) - 1 = n$.

So the new lower limit is 0 and the new upper limit is n.

Now n is a constant with respect to the terms of the sum. Unlike k, its value does not change from one term to the next.

Example 5.1.14 – *Solution (2/3)* continued

In addition, since $j = k - 1$, then $k = j + 1$.

Thus

$$\frac{k}{n+k} = \frac{j+1}{n+(j+1)}$$

and so the general term of the new summation is

$$\frac{j+1}{n+(j+1)}.$$

Therefore,

$$\sum_{k=1}^{n+1} \frac{k}{n+k} = \sum_{j=0}^{n} \frac{j+1}{n+(j+1)}. \qquad 5.1.3$$

Example 5.1.14 – *Solution (3/3)* continued

b. Changing all the *j*'s to *k*'s in the right-hand side of equation (5.1.3) gives

$$\sum_{j=0}^{n} \frac{j+1}{n+(j+1)} = \sum_{k=0}^{n} \frac{k+1}{n+(k+1)}. \qquad 5.1.4$$

Combining equations (5.1.3) and (5.1.4) results in

$$\sum_{k=1}^{n+1} \frac{k}{n+k} = \sum_{k=0}^{n} \frac{k+1}{n+(k+1)}.$$

Factorial and “*n* Choose *r*” Notation

Factorial and "*n* Choose *r*" Notation (1/5)

The product of all consecutive integers up to a given integer occurs so often in mathematics that it is given a special notation—*factorial* notation.

Definition

For each positive integer n, the quantity ***n* factorial** denoted $\boldsymbol{n!}$, is defined to be the product of all the integers from 1 to n:

$$n! = n \cdot (n-1) \cdots 3 \cdot 2 \cdot 1.$$

Zero factorial, denoted $0!$, is defined to be 1:

$$0! = 1.$$

Example 5.1.15 – *The First Ten Factorials*

$$0! = 1 \qquad 1! = 1$$

$$2! = 2 \cdot 1 = 2 \qquad 3! = 3 \cdot 2 \cdot 1 = 6$$

$$4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24 \qquad 5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$$

$$6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720 \qquad 7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5{,}040$$

$$8! = 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 40{,}320 \qquad 9! = 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 362{,}880$$

Factorial and "*n* Choose *r*" Notation (2/5)

A recursive definition for factorial is the following: Given any nonnegative integer *n*,

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{if } n \geq 1. \end{cases}$$

The next example illustrates the usefulness of the recursive definition for making computations.

Example 5.1.16 – *Computing with Factorials*

Simplify the following expressions:

a. $\dfrac{8!}{7!}$ b. $\dfrac{5!}{2!\cdot 3!}$ c. $\dfrac{1}{2!\cdot 4!}+\dfrac{1}{3!\cdot 3!}$ d. $\dfrac{(n+1)!}{n!}$ e. $\dfrac{n!}{(n-3)!}$

Example 5.1.16 – *Solution (1/2)*

a. $\dfrac{8!}{7!} = \dfrac{8 \cdot \cancel{7!}}{\cancel{7!}} = 8$

b. $\dfrac{5!}{2! \cdot 3!} = \dfrac{5 \cdot 4 \cdot \cancel{3!}}{2! \cdot \cancel{3!}} = \dfrac{5 \cdot 4}{2 \cdot 1} = 10$

c. $\dfrac{1}{2! \cdot 4!} + \dfrac{1}{3! \cdot 3!} = \dfrac{1}{2! \cdot 4!} \cdot \dfrac{3}{3} + \dfrac{1}{3! \cdot 3!} \cdot \dfrac{4}{4}$

by multiplying each numerator and denominator by just what is necessary to obtain a common denominator

$= \dfrac{3}{3 \cdot 2! \cdot 4!} + \dfrac{4}{3! \cdot 4 \cdot 3!}$

by rearranging factors

Example 5.1.16 – *Solution (2/2)* continued

$$= \frac{3}{3!\cdot 4!} + \frac{4}{3!\cdot 4!}$$

because $3\cdot 2! = 3!$ and $4\cdot 3! = 4!$

$$= \frac{7}{144}$$

by the rule for adding fractions with a common denominator

d. $$\frac{(n+1)!}{n!} = \frac{(n+1)\cdot \cancel{n!}}{\cancel{n!}} = n+1$$

e. $$\frac{n!}{(n-3)!} = \frac{n\cdot(n-1)\cdot(n-2)\cdot\cancel{(n-3)!}}{\cancel{(n-3)!}} = n\cdot(n-1)\cdot(n-2)$$

$$= n^3 - 3n^2 + 2n$$

Factorial and "*n* Choose *r*" Notation (3/5)

An important use for the factorial notation is in calculating values of quantities, called *n choose r*, that occur in many branches of mathematics, especially those connected with the study of counting techniques and probability.

Definition

Let n and r be integers with $0 \leq r \leq n$. The symbol

$$\binom{n}{r}$$

is read "***n* choose *r***" and represents the number of subsets of size r that can be chosen from a set with n elements.

Factorial and "*n* Choose *r*" Notation (4/5)

Formula for Computing $\binom{n}{r}$

For all integers n and r with $0 \le r \le n$,

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}.$$

In this chapter, we show how to compute its values. Because *n* choose *r* is always an integer, you can be sure that all the factors in the denominator of the formula will be canceled out by factors in the numerator.

Factorial and "*n* Choose *r*" Notation (5/5)

Many electronic calculators have keys for computing values of $\binom{n}{r}$.

These are denoted in various ways such as nCr, $C(n, r)$, ${}^{n}C_{r}$, and $C_{n,r}$.

The letter *C* is used because the quantities $\binom{n}{r}$ are also called *combinations*. Sometimes they are referred to as *binomial coefficients* because of the connection with the binomial theorem.

Example 5.1.17 – *Computing* $\binom{n}{r}$

Use the formula for computing $\binom{n}{r}$ to evaluate the following expressions:

a. $\binom{8}{5}$ b. $\binom{4}{4}$ c. $\binom{n+1}{n}$

Example 5.1.17 – *Solution (1/2)*

a. $$\binom{8}{5} = \frac{8!}{5!(8-5)!}$$

$$= \frac{8 \cdot 7 \cdot \cancel{6} \cdot \cancel{5} \cdot \cancel{4} \cdot \cancel{3 \cdot 2 \cdot 1}}{(\cancel{5} \cdot \cancel{4} \cdot \cancel{3} \cdot \cancel{2} \cdot 1) \cdot (\cancel{3 \cdot 2 \cdot 1})}$$

always cancel common factors before multiplying

$$= 56.$$

Example 5.1.17 – *Solution (2/2)* continued

b. $$\binom{4}{4} = \frac{4!}{4!(4-4)!} = \frac{4!}{4!0!} = \frac{\cancel{4\cdot 3\cdot 2\cdot 1}}{(\cancel{4\cdot 3\cdot 2\cdot 1})(1)} = 1$$

The fact that $0! = 1$ makes this formula computable. It gives the correct value because a set of size 4 has exactly one subset of size 4, namely itself.

c. $$\binom{n+1}{n} = \frac{(n+1)!}{n!((n+1)-n)!} = \frac{(n+1)!}{n!1!} = \frac{(n+1)\cdot \cancel{n!}}{\cancel{n!}} = n+1$$

Sequences in Computer Programming

Sequences in Computer Programming (1/2)

An important data type in computer programming consists of finite sequences. In computer programming contexts, these are usually referred to as *one-dimensional arrays*. For example, consider a program that analyzes the wages paid to a sample of 50 workers.

Such a program might compute the average wage and the difference between each individual wage and the average.

This would require that each wage be stored in memory for retrieval later in the calculation.

Sequences in Computer Programming (2/2)

To avoid the use of entirely separate variable names for all of the 50 wages, each is written as a term of a one-dimensional array:

W[1], W[2], W[3], … , W[50].

Note that the subscript labels are written inside square brackets. The reason is that until relatively recently, it was impossible to type actual dropped subscripts on most computer keyboards.

Example 5.1.18 – *Dummy Variable in a Loop*

The index variable for a **for-next** loop is a local, or dummy, variable. For example, the following three algorithm segments all produce the same output:

1. **for** i := 1 **to** n
   print $a[i]$
   next i

2. **for** j := 0 **to** $n - 1$
   print $a[j + 1]$
   next j

3. **for** k := 2 **to** $n + 1$
   print $a[k - 1]$
   next k

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (1/8)

A systematic algorithm to convert any nonnegative integer to binary notation uses repeated division by 2.

Suppose a is a nonnegative integer. Divide a by 2 using the quotient-remainder theorem to obtain a quotient $q[0]$ and a remainder $r[0]$. If the quotient is nonzero, divide by 2 again to obtain a quotient $q[1]$ and a remainder $r[1]$.

Continue this process until a quotient of 0 is obtained. At each stage, the remainder must be less than the divisor, which is 2. Thus each remainder is always either 0 or 1.

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (2/8)

The process is illustrated below for *a* = 38. (Read the divisions from the bottom up.)

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (3/8)

The results of all these divisions can be written as a sequence of equations:

$$
\begin{aligned}
38 &= 19 \cdot 2 + 0, \\
19 &= 9 \cdot 2 + 1, \\
9 &= 4 \cdot 2 + 1, \\
4 &= 2 \cdot 2 + 0, \\
2 &= 1 \cdot 2 + 0, \\
1 &= 0 \cdot 2 + 1.
\end{aligned}
$$

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (4/8)

By repeated substitution, then

$$
\begin{aligned}
38 &= 19\cdot 2+0\\
&= (9\cdot 2+1)\cdot 2+0 = 9\cdot 2^2+1\cdot 2+0\\
&= (4\cdot 2+1)\cdot 2^2+1\cdot 2+0 = 4\cdot 2^3+1\cdot 2^2+1\cdot 2+0\\
&= (2\cdot 2+0)\cdot 2^3+1\cdot 2^2+1\cdot 2+0\\
&= 2\cdot 2^4+0\cdot 2^3+1\cdot 2^2+1\cdot 2+0\\
&= (1\cdot 2+0)\cdot 2^4+0\cdot 2^3+1\cdot 2^2+1\cdot 2+0\\
&= 1\cdot 2^5+0\cdot 2^4+0\cdot 2^3+1\cdot 2^2+1\cdot 2+0.
\end{aligned}
$$

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (5/8)

Note that each coefficient of a power of 2 on the right-hand side is one of the remainders obtained in the repeated division of 38 by 2.

This is true for the left-most 1 as well, because $1 = 0 \cdot 2 + 1$.

Thus

$$38_{10} = 100110_2 = (r[5]r[4]r[3]r[2]r[1]r[0])_2.$$

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (6/8)

In general, if a nonnegative integer *a* is repeatedly divided by 2 until a quotient of zero is obtained and the remainders are found to be $r[0]$, $r[1]$, … , $r[k]$, then by the quotient-remainder theorem each $r[i]$ equals 0 or 1, and by repeated substitution from the theorem.

$$a = 2^k \cdot r[k] + 2^{k-1} \cdot r[k-1] + \cdots + 2^2 \cdot r[2] + 2^1 \cdot r[1] + 2^0 \cdot r[0]. \qquad 5.1.5$$

Thus the binary representation for *a* can be read from equation (5.1.5):

$$a_{10} = (r[k]r[k-1] \cdots r[2]r[1]r[0])_2.$$

Example 5.1.19 – *Converting from Decimal to Binary Notation Using Repeated Division by 2*

Use repeated division by 2 to write the number 29_{10} in binary notation.

Example 5.1.19 – *Solution*

Hence $29_{10} = (r[4]r[3]r[2]r[1]r[0])_2 = 11101_2$.

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (7/8)

Algorithm 5.1.1 Decimal to Binary Conversion Using Repeated Division by 2

[In Algorithm 5.1.1 the input is a nonnegative integer a. The aim of the algorithm is to produce a sequence of binary digits r[0], r[1], r[2], . . . , r[k] so that the binary representation of n is

$$(r[k]r[k-1]\cdots r[2]r[1]r[0])_2.$$

That is,

$$a = 2^k \cdot r[k] + 2^{k-1} \cdot r[k-1] + \cdots + 2^2 \cdot r[2] + 2^1 \cdot r[1] + 2^0 \cdot r[0].]$$

Input: *a [a nonnegative integer]*

Algorithm Body:

$q := a, i := 0$

[Repeatedly perform the integer division of q by 2 until q becomes 0. Store successive remainders in a one-dimensional array r[0], r[1], r[2], . . . , r[k]. Even if the initial-value of q equals 0, the loop should execute one time (so that r[0] is computed).

Application: Algorithm to Convert from Base 10 to Base 2 Using Repeated Division by 2 (8/8) continued

Algorithm 5.1.1 Decimal to Binary Conversion Using Repeated Division by 2

Thus the guard condition for the **while** *loop is* $i = 0$ *or* $q \neq 0$.*]*

while ($i = 0$ *or* $q \neq 0$)
  $r[i] := q \bmod 2$
  $q := q \operatorname{div} 2$
  [$r[i]$ *and* q *can be obtained by calling the division algorithm.]*
  $i := i + 1$
end while

[After execution of this step, the values of $r[0], r[1], \ldots, r[i-1]$ *are all* 0*'s and* 1*'s, and* $a = (r[i-1]r[i-2]\cdots r[2]r[1]r[0])_2$.*]*

Output: $r[0], r[1], r[2], \ldots, r[i-1]$ *[a sequence of integers]*

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.2 Mathematical Induction I: Proving Formulas

Copyright © Cengage Learning. All rights reserved.

Mathematical Induction I: Proving Formulas (1/11)

In natural science courses, deduction and induction are presented as alternative modes of thought—deduction being to infer a conclusion from general principles using the laws of logical reasoning, and induction being to enunciate a general principle after observing it to hold in a large number of specific instances.

Discovery of new mathematical facts often occurs through experimentation with examples, but *mathematical induction as a proof technique is not inductive but deductive*.

Mathematical Induction I: Proving Formulas (2/11)

Once proved by mathematical induction, a theorem is known just as certainly as if were proved by any other mathematical method.

Principle of Mathematical Induction

Let $P(n)$ be a property that is defined for integers n, and let a be a fixed integer. Suppose the following two statements are true:

1. $P(a)$ is true.
2. For every integer $k \geq a$, if $P(k)$ is true then $P(k+1)$ is true.

Then the statement

$$\text{for every integer } n \geq a,\ P(n)$$

is true.

Mathematical Induction I: Proving Formulas (3/11)

The first known use of mathematical induction occurs in the work of the Italian scientist Francesco Maurolico in 1575.

In 1653 Blaise Pascal gave a clear description of the technique, and in 1883 Augustus De Morgan (best known for De Morgan's laws) gave the process the name *mathematical induction*.

An equivalent logical inference rule, now known as the *well-ordering principle*.

Mathematical Induction I: Proving Formulas (4/11)

To visualize the idea of mathematical induction, imagine an infinite collection of dominoes positioned one behind the other in such a way that if any given domino falls backward, it makes the one behind it fall backward also.

(See Figure 5.2.1.) Then imagine that the first domino falls backward. What happens? ... They all fall down!

If the *k*th domino falls backward, it pushes the (*k* + 1)st domino backward also.

Figure 5.2.1

Mathematical Induction I: Proving Formulas (5/11)

To see the connection between this image and the principle of mathematical induction, let $P(n)$ be the sentence "The nth domino falls backward."

It is given that for each $k \geq 1$, if $P(k)$ is true (the kth domino falls backward), then $P(k + 1)$ is also true (the $(k + 1)$st domino falls backward). It is also given that $P(1)$ is true (the first domino falls backward).

Thus by the principle of mathematical induction, $P(n)$ (the nth domino falls backward) is true for every integer $n \geq 1$.

Mathematical Induction I: Proving Formulas (6/11)

Proving a statement by mathematical induction is a two-step process. The first step is called the *basis step*, and the second step is called the *inductive step*.

Method of Proof by Mathematical Induction

Consider a statement of the form, "For every integer $n \geq a$, a property $P(n)$ is true." To prove such a statement, perform the following two steps:

Step 1 (basis step): Show that $P(a)$ is true.

Step 2 (inductive step): Show that for every integer $k \geq a$, if $P(k)$ is true then $P(k+1)$ is true. To perform this step,

> **suppose** that $P(k)$ is true, where k is any particular but arbitrarily chosen integer with $k \geq a$.
>
> *[This supposition is called the* **inductive hypothesis***.]*

Then

> **show** that $P(k+1)$ is true.

Example 5.2.1 – *Sum of the First n Integers*

Use mathematical induction to prove that

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

for every integer $n \geq 1$.

Example 5.2.1 – *Solution (1/9)*

To construct a proof by induction, you must first identify the property $P(n)$. In this case, $P(n)$ is the equation

$$\boxed{1 + 2 + \cdots + n = \frac{n(n+1)}{2}.} \quad \leftarrow \text{the property } (P(n))$$

Note The property is just the equation. The proof will show that the equation is true for every integer $n \geq 1$. *[To see that* $P(n)$ *is a sentence*, *note that its subject is "the sum of the integers from* 1 *to n" and its verb is "equals*."*]*

Example 5.2.1 – *Solution (2/9)* continued

In the basis step of the proof, you must show that the property is true for $n = 1$, or, in other words, that $P(1)$ is true. Now $P(1)$ is obtained by substituting 1 in place of n in $P(n)$.

The left-hand side of $P(1)$ is the sum of all the successive integers starting at 1 and ending at 1. This is just 1.

Note To write $P(1)$, just copy $P(n)$ and replace each n by 1.

Example 5.2.1 – *Solution (3/9)* continued

Thus $P(1)$ is

$$\boxed{1 = \frac{1(1+1)}{2}.} \qquad \leftarrow \text{basis } (P(1))$$

Of course, this equation is true because the right-hand side is

$$\frac{1(1+1)}{2} = \frac{1 \cdot 2}{2} = 1,$$

which equals the left-hand side.

Example 5.2.1 – *Solution (4/9)* continued

In the inductive step, you assume that $P(k)$ is true, for a particular but arbitrarily chosen integer k with $k \geq 1$. *[This assumption is the inductive hypothesis.]* You must then show that $P(k + 1)$ is true. What are $P(k)$ and $P(k + 1)$? $P(k)$ is obtained by substituting k for every n in $P(n)$.

Thus $P(k)$ is

$$\boxed{1 + 2 + \cdots + k = \frac{k(k+1)}{2}.} \qquad \leftarrow \text{inductive hypothesis } (P(k))$$

Example 5.2.1 – *Solution (5/9)* continued

Similarly, $P(k + 1)$ is obtained by substituting the quantity $(k + 1)$ for every n that appears in $P(n)$. Thus $P(k + 1)$ is

$$1 + 2 + \cdots + (k+1) = \frac{(k+1)((k+1)+1)}{2},$$

or, equivalently,

$$\boxed{1 + 2 + \cdots + (k+1) = \frac{(k+1)(k+2)}{2}.} \quad \leftarrow \text{to show } (P(k+1))$$

Example 5.2.1 – *Solution (6/9)* continued

Now the inductive hypothesis is the supposition that $P(k)$ is true. How can this supposition be used to show that $P(k + 1)$ is true? $P(k + 1)$ is an equation, and the truth of an equation can be shown in a variety of ways.

One of the most straightforward is to use the inductive hypothesis along with algebra and other known facts to separately transform the left-hand and right-hand sides until you see that they are the same.

Example 5.2.1 – *Solution (7/9)* continued

In this case, the left-hand side of $P(k + 1)$ is

$$1 + 2 + \ldots + (k + 1),$$

which equals

$$(1 + 2 + \ldots + k) + (k + 1)$$

The next-to-last term is k because the terms are successive integers and the last term is $k + 1$.

By substitution from the inductive hypothesis,

$$(1 + 2 + \ldots + k) + (k + 1)$$

Example 5.2.1 – *Solution (8/9)* continued

$$= \frac{k(k+1)}{2} + (k+1)$$

since the inductive hypothesis says that $1 + 2 + \cdots + k = \frac{k(k+1)}{2}$

$$= \frac{k(k+1)}{2} + \frac{2(k+1)}{2}$$

by multiplying the numerator and denominator of the second term by 2 to obtain a common denominator

$$= \frac{k^2+k}{2} + \frac{2k+2}{2}$$

by multiplying out the two numerators

$$= \frac{k^2+3k+2}{2}$$

by adding fractions with the same denominator and combining like terms.

Example 5.2.1 – *Solution (9/9)* continued

So the left-hand side of $P(k + 1)$ is $\frac{k^2 + 3k + 2}{2}$.

Now the right-hand side of $P(k + 1)$ is $\frac{(k+1)(k+2)}{2} = \frac{k^2 + 3k + 2}{2}$ by multiplying out the numerator.

Thus the two sides of $P(k + 1)$ are equal to each other, and so the equation $P(k + 1)$ is true.

Mathematical Induction I: Proving Formulas (7/11)

Theorem 5.2.1 Sum of the First n Integers

For every integer $n \geq 1$,

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

Definition

If a sum with a variable number of terms is shown to equal an expression that does not contain either an ellipsis or a summation symbol, we say that the sum is written **in closed form**.

Example 5.2.2 – *Applying the Formula for the Sum of the First n Integers*

a. Evaluate $2 + 4 + 6 + \ldots + 500$.

b. Evaluate $5 + 6 + 7 + 8 + \ldots + 50$.

c. For an integer $h \geq 2$, write $1 + 2 + 3 + \ldots + (h - 1)$ in closed form.

Example 5.2.2 – *Solution (1/3)*

a. $2 + 4 + 6 + \ldots + 500 = 2 \cdot (1 + 2 + 3 + \ldots + 250)$

$$= 2 \cdot \left(\frac{250 \cdot 251}{2} \right)$$

by applying the formula for the sum of the first *n* integers with *n* = 250.

$$= 62{,}750$$

Example 5.2.2 – *Solution (2/3)* continued

b. 5 + 6 + 7 + 8 + ... + 50 = (1 + 2 + 3 + ... + 50) − (1 + 2 + 3 + 4)

$$= \frac{50 \cdot 51}{2} - 10$$

by applying the formula for the sum of the first *n* integers with *n* = 50.

= 1,265

Example 5.2.2 – *Solution (3/3)*

continued

c. 1 + 2 + 3 + ... + (h − 1)

$$= \frac{(h-1)\cdot[(h-1)+1]}{2}$$

by applying the formula for the sum of the first n integers with $n = h - 1$

$$= \frac{(h-1)\cdot h}{2}$$

since $(h - 1) + 1 = h$.

Mathematical Induction I: Proving Formulas (8/11)

Another famous and important formula in mathematics—the formula for the sum of a geometric sequence.

In a **geometric sequence**, each term is obtained from the preceding one by multiplying by a constant factor. If the first term is 1 and the constant factor is *r*, then the sequence is $1, r, r^2, r^3, \ldots, r^n, \ldots$.

Mathematical Induction I: Proving Formulas (9/11)

The sum of the first *n* terms of this sequence is given by the formula

$$\sum_{i=}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}$$

for every integer $n \geq 0$ and every real number *r* not equal to 1. The expanded form of the formula is

$$r^0 + r^1 + r^2 + \cdots + r^n = \frac{r^{n+1} - 1}{r - 1},$$

and because $r^0 = 1$ and $r^1 = r$, the formula for $n \geq 1$ can be rewritten as

$$1 + r + r^2 + \cdots + r^n = \frac{r^{n+1} - 1}{r - 1}.$$

Mathematical Induction I: Proving Formulas (10/11)

In some mathematical contexts 0^0 is regarded as indeterminate. In discrete mathematics we usually define $0^0 = 1$ so that when we write $\sum_{i=0}^{n} r^i$ we do not need to give special attention to the case $r = 0$.

Example 5.2.3 – *Sum of a Geometric Sequence*

Prove that $\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}$, for every integer $n \geq 0$ and every real number r except 1.

Example 5.2.3 – *Solution (1/3)*

In this example the property $P(n)$ is again an equation, although in this case it contains a real variable r.

$$\boxed{\sum_{i=0}^{n} r^i = \frac{r^{n+1}-1}{r-1}.} \quad \leftarrow \text{the property } (P(n))$$

Because r can be any real number other than 1, the proof begins by supposing that r is a particular but arbitrarily chosen real number not equal to 1. Then the proof continues by mathematical induction on n, starting with $n = 0$.

Example 5.2.3 – *Solution (2/3)* continued

In the basis step, you must show that $P(0)$ is true; that is, you show the property is true for $n = 0$. So you substitute 0 for each n in $P(n)$:

$$\boxed{\sum_{i=0}^{0} r^i = \frac{r^{0+1} - 1}{r - 1}.} \quad \leftarrow \text{basis } (P(0))$$

In the inductive step, you suppose k is any integer with $k \geq 0$ for which $P(k)$ is true; that is, you suppose the property is true for $n = k$. So you substitute k for each n in $P(n)$:

$$\boxed{\sum_{i=0}^{k} r^i = \frac{r^{k+1} - 1}{r - 1}.} \quad \leftarrow \text{inductive hypothesis } (P(k))$$

Example 5.2.3 – *Solution (3/3)*

continued

Then you show that $P(k + 1)$ is true; that is, you show the property is true for $n = k + 1$. So you substitute $k + 1$ for each n in $P(n)$:

$$\sum_{i=0}^{k+1} r^i = \frac{r^{(k+1)+1} - 1}{r - 1},$$

or, equivalently,

$$\boxed{\sum_{i=0}^{k+1} r^i = \frac{r^{k+2} - 1}{r - 1}.} \leftarrow \text{to show } (P(k+1))$$

Mathematical Induction I: Proving Formulas (11/11)

Theorem 5.2.2 Sum of a Geometric Sequence

For any real number r except 1, and any integer $n \geq 0$,

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}.$$

Proving an Equality

Proving an Equality (1/2)

The proofs of the basis and inductive steps illustrate two different ways to show that an equation is true: (1) transforming the left-hand side and the right-hand side independently until they are seen to be equal, and (2) transforming one side of the equation until it is seen to be the same as the other side of the equation.

Proving an Equality (2/2)

Sometimes people use a method that they believe proves equality but that is actually invalid. For example, to prove the basis step for Theorem 5.2.2, they perform the following steps:

$$\sum_{i=0}^{0} r^i = \frac{r^{0+1}-1}{r-1}$$

$$r^0 = \frac{r^1-1}{r-1}$$

$$1 = \frac{r-1}{r-1}$$

$$1 = 1.$$

Deducing Additional Formulas

Deducing Additional Formulas (1/3)

The formula for the sum of a geometric sequence can be thought of as a family of different formulas in *r*, one for each real number *r* except 1.

Example 5.2.4 – *Applying the Formula for the Sum of a Geometric Sequence*

In each of (a) and (b) below, assume that *m* is an integer that is greater than or equal to 3. Write each of the sums in closed form.

a. $1 + 3 + 3^2 + \cdots + 3^{m-2}$

b. $3^2 + 3^3 + 3^4 + \cdots + 3^m$

Example 5.2.4 – *Solution*

a. $$1+3+3^2+\cdots+3^{m-2} = \frac{3^{(m-2)+1}-1}{3-1}$$

by applying the formula for the sum of a geometric sequence with $r = 3$ and $n = m - 2$

$$= \frac{3^{m-1}-1}{2}$$

b. $$3^2+3^3+3^4+\cdots+3^m = 3^2\cdot(1+3+3^2+\cdots+3^{m-2})$$ by factoring out 3^2

$$= 9\cdot\left(\frac{3^{m-1}-1}{2}\right)$$ by part (a).

Deducing Additional Formulas (2/3)

As with the formula for the sum of the first *n* integers, there is a way to think of the formula for the sum of the terms of a geometric sequence that makes it seem simple and intuitive. Let

$$S_n = 1 + r + r^2 + \cdots + r^n.$$

Then

$$rS_n = r + r^2 + r^3 + \cdots + r^{n+1},$$

and so

$$\begin{aligned} rS_n - S_n &= (r + r^2 + r^3 + \cdots + r^{n+1}) - (1 + r + r^2 + \cdots + r^n) \\ &= r^{n+1} - 1. \end{aligned}$$

5.2.1

Deducing Additional Formulas (3/3)

Now

$$rS_n - S_n = (r-1)S_n. \qquad 5.2.2$$

Equating the right-hand sides of equations (5.2.1) and (5.2.2) and dividing by $r - 1$ gives

$$S_n = \frac{r^{n+1} - 1}{r - 1}.$$

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.3 Mathematical Induction II: Applications

Copyright © Cengage Learning. All rights reserved.

Mathematical Induction II: Applications (1/3)

In this section we will show how to apply it in a broader variety of situations.

As a first example consider the argument that the U.S. penny should be eliminated because it isn't profitable to produce.

Due to inflation and the rising cost of metals, it actually costs more than one cent to produce a penny. If the penny were eliminated and another coin worth 3¢ were introduced, what prices could be paid using only 3¢ and 5¢ coins?

Mathematical Induction II: Applications (2/3)

The table below shows some examples.

Number of Cents	How to Obtain It
3¢	3¢
5¢	5¢
8¢	3¢ + 5¢
9¢	3¢ + 3¢ + 3¢
10¢	5¢ + 5¢
11¢	3¢ + 3¢ + 5¢
12¢	3¢ + 3¢ + 3¢ + 3¢
13¢	3¢ + 5¢ + 5¢
14¢	3¢ + 3¢ + 3¢ + 5¢
15¢	5¢ + 5¢ + 5¢
16¢	3¢ + 3¢ + 5¢ + 5¢
17¢	3¢ + 3¢ + 3¢ + 3¢ + 5¢

Mathematical Induction II: Applications (3/3)

Proposition 5.3.1

For every integer $n \geq 8$, n¢ can be obtained using 3¢ and 5¢ coins.

Example 5.3.1 – *Proving a Divisibility Property*

Use mathematical induction to prove that for each integer $n \geq 0$, $2^{2n} - 1$ is divisible by 3.

Example 5.3.1 – *Solution (1/5)*

As in the previous proofs by mathematical induction, you need to identify the property $P(n)$. In this example, $P(n)$ is the sentence

$$\boxed{2^{2n} - 1 \text{ is divisible by } 3.} \qquad \leftarrow \text{the property } (P(n))$$

By substitution, the statement for the basis step, $P(0)$, is

$$\boxed{2^{2\cdot 0} - 1 \text{ is divisible by } 3.} \qquad \leftarrow \text{basis } (P(0))$$

Example 5.3.1 – *Solution (2/5)* continued

The supposition for the inductive step, $P(k)$, is

$$\boxed{2^{2k} - 1 \text{ is divisible by } 3,} \qquad \leftarrow \text{inductive hypothesis } (P(k))$$

and the conclusion to be shown, $P(k + 1)$, is

$$\boxed{2^{2(k+1)} - 1 \text{ is divisible by } 3.} \qquad \leftarrow \text{to show } (P(k+1))$$

Recall that an integer m is divisible by 3 if, and only if, $m = 3r$ for some integer r. Now the statement $P(0)$ is true because $2^{2\cdot 0} - 1 = 2^0 - 1 = 1 - 1 = 0,$ which is divisible by 3 because $0 = 3 \cdot 0$.

Example 5.3.1 – *Solution (3/5)* continued

To prove the inductive step, you suppose that *k* is any integer greater than or equal to 0 such that *P*(*k*) is true. In other words, you suppose that $2^{2k} - 1$ is divisible by 3. You must then prove the truth of *P*(*k* + 1).

Thus, you must show that $2^{2(k+1)} - 1$ is divisible by 3. Now

$$\begin{aligned} 2^{2(k+1)} - 1 &= 2^{2k+2} - 1 \\ &= 2^{2k} \cdot 2^2 - 1 \quad \text{by the laws of exponents.} \\ &= 2^{2k} \cdot 4 - 1. \end{aligned}$$

Example 5.3.1 – *Solution (4/5)* continued

Your aim is to show that $2^{2k}\cdot 4-1$ is divisible by 3, but why should that be so? Both $2^{2k}\cdot 4-1$ and $2^{2k}-1$ have a lot in common, and, by the inductive hypothesis, $2^{2k}-1$ is divisible by 3. Observe what happens if you subtract $2^{2k}-1$ from $2^{2k}\cdot 4-1$:

$$\underbrace{2^{2k}\cdot 4-1}_{\text{divisible by 3?}}-\underbrace{(2^{2k}-1)}_{\text{divisible by 3}}=\underbrace{2^{2k}\cdot 3}_{\text{divisible by 3}}.$$

Adding $2^{2k}-1$ to both sides gives

$$\underbrace{2^{2k}\cdot 4-1}_{\text{divisible by 3?}}=\underbrace{2^{2k}\cdot 3}_{\text{divisible by 3}}+\underbrace{(2^{2k}-1)}_{\text{divisible by 3}}.$$

Example 5.3.1 – *Solution (5/5)* continued

Both terms of the sum on the right-hand side of this equation are divisible by 3; hence the sum is divisible by 3.

Therefore, the left-hand side of the equation is also divisible by 3, which is what was to be shown.

Proposition 5.3.2

For each integer $n \geq 0$, $2^{2n} - 1$ is divisible by 3.

Example 5.3.2 – *Proving an Inequality*

Use mathematical induction to prove that for each integer $n \geq 3$,

$$2^n + 1 < 2^n.$$

Example 5.3.2 – *Solution (1/4)*

In this example the property $P(n)$ is the inequality

$$\boxed{2n + 1 < 2^n.} \qquad \leftarrow \text{the property } (P(n))$$

By substitution, the statement for the basis step, $P(3)$, is

$$\boxed{2 \cdot 3 + 1 < 2^3.} \qquad \leftarrow \text{basis } (P(3))$$

The supposition for the inductive step, $P(k)$, is

$$\boxed{2k + 1 < 2^k,} \qquad \leftarrow \text{inductive hypothesis } (P(k))$$

and the conclusion to be shown is

$$\boxed{2(k + 1) + 1 < 2^{k+1}.} \qquad \leftarrow \text{to show } (P(k + 1))$$

Example 5.3.2 – *Solution (2/4)*

continued

To prove the basis step, observe that the statement *P*(3) is true because $2 \cdot 3 + 1 = 7$, $2^3 = 8$, and 7 < 8.

For the inductive step you assume the inductive hypothesis that for a particular but arbitrarily chosen integer $k \geq 3$, $2k + 1 < 2^k$, and then you must show that $2k + 1 < 2^{k+1}$. With inequality proofs you can often apply the inductive hypothesis at an early stage. In this case, you can substitute from the induction hypothesis to obtain

$$2(k+1)+1 \;=\; (2k+1)+2 < 2^k + 2.$$

Example 5.3.2 – *Solution (3/4)* continued

At this point you need to think about your goal. You want the right-hand side of the inequality to be 2^{k+1}.

Since you already know that $2(k+1)+1 < 2^k + 2,$ an easy way to complete your job would be to show that

$$2^k + 2 < 2^{k+1}.$$

But this inequality is true because $2 < 2^k \text{ for } k \geq 3,$ and thus

$$2^k + 2 < 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$$

by algebra and a law of exponents. Transitivity of order then implies that $2(k+1)+1 < 2^{k+1}.$

Example 5.3.2 – *Solution (4/4)*

continued

Proposition 5.3.3

For every integer $n \geq 3$, $2n + 1 < 2^n$.

Example 5.3.3 – *Proving a Property of a Sequence*

Define a sequence $a_1, a_2, a_3, \ldots$ as follows:

$$a_1 = 2$$

$$a_k = 5a_{k-1} \qquad \text{for every integer } k \geq 2.$$

a. Write the first four terms of the sequence.

b. It is claimed that for each integer $n \geq 0$, the nth term of the sequence has the same value as that given by the formula $2 \cdot 5^{n-1}$. In other words, the claim is that the terms of the sequence satisfy the equation $a_n = 2 \cdot 5^{n-1}$.

Prove that this is true.

Example 5.3.3 – *Solution (1/6)*

a. $a_1 = 2$

$a_2 = 5a_{2-1} = 5a_1 = 5 \cdot 2 = 10$

$a_3 = 5a_{3-1} = 5a_2 = 5 \cdot 10 = 50$

$a_4 = 5a_{4-1} = 5a_3 = 5 \cdot 50 = 250$

b. To use mathematical induction to show that every term of the sequence satisfies the equation, begin by showing that the first term of the sequence satisfies the equation.

Example 5.3.3 – *Solution (2/6)* continued

Then suppose that an arbitrarily chosen term a_k satisfies the equation and prove that the next term a_{k+1} also satisfies the equation.

Proof (by mathematical induction):

Let $a_1, a_2, a_3, \ldots$ be the sequence defined by specifying that $a_1 = 2$ and $a_k = 5a_{k-1}$ for every integer $k \geq 2$, and let the property $P(n)$ be the equation

$$a_n = 2 \cdot 5^{n-1}. \qquad \leftarrow P(n)$$

Example 5.3.3 – *Solution (3/6)* continued

We will use mathematical induction to prove that for every integer $n \geq 1$, $P(n)$ is true.

Show that P*(1) *is true:

To establish $P(1)$, we must show that

$$a_1 = 2 \cdot 5^{1-1}. \qquad \leftarrow P(1)$$

Now the left-hand side of $P(1)$ is

$$a_1 = 2 \qquad \text{by definition of } a_1\ a_2,\ a_3,\ \ldots,$$

and the right-hand side of $P(1)$ is

$$2 \cdot 5^{1-1} = 2 \cdot 5^0 = 2 \cdot 1 = 2.$$

Example 5.3.3 – *Solution (4/6)* continued

Thus the two sides of $P(1)$ are equal to the same quantity, and hence $P(1)$ is true.

Show that for each integer $k \geq 1$, if $P(k)$ is true then $P(k+1)$ is also true:

[Suppose that $P(k)$ is true for a particular but arbitrarily chosen integer $k \geq 1$. That is:] Let k be any integer with $k \geq 1$, and suppose that

$$a_k = 2 \cdot 5^{k-1}. \qquad \leftarrow P(k)$$

inductive hypothesis

Example 5.3.3 – *Solution (5/6)* continued

[We must show that P(k + 1) is true. That is:] We must show that

$$a_{k+1} = 2 \cdot 5^{(k+1)-1},$$

or, equivalently,

$$a_{k+1} = 2 \cdot 5^{k}. \qquad \leftarrow P(k+1)$$

But the left-hand side of $P(k + 1)$ is

$$a_{k+1} = 5a_{(k+1)-1} \qquad \text{by definition of } a_1, a_2, a_3, \ldots$$

$$= 5a_k \qquad \text{since } (k+1) - 1 = k$$

Example 5.3.3 – *Solution (6/6)*

continued

$$= 5 \cdot (2 \cdot 5^{k-1}) \quad \text{by inductive hypothesis}$$

$$= 2 \cdot (5 \cdot 5^{k-1}) \quad \text{by regrouping}$$

$$= 2 \cdot 5^{k} \quad \text{by the laws of exponents,}$$

which is the right-hand side of $P(k + 1)$ *[as was to be shown].*

[Since we have proved the basis step and the inductive step, we conclude that the formula holds for all terms of the sequence.]

A Problem with Trominoes

A Problem with Trominoes (1/3)

The word *polyomino*, a generalization of *domino*, was introduced by Solomon Golomb in 1954 when he was a 22-year-old university student. Subsequently, he and others proved many interesting properties about them, and they became the basis for the popular computer game Tetris. A particular type of polyomino, called a *tromino*, is made up of three attached squares, which can be of two types:

straight and L-shaped

A Problem with Trominoes (2/3)

Call a checkerboard that is formed using *m* squares on a side an $m \times m$ ("*m* by *m*") checkerboard. Observe that if one square is removed from a 4×4 checkerboard, the remaining squares can be completely covered by L-shaped trominoes. For instance, a covering for one such board is illustrated in the figure below.

A Problem with Trominoes (3/3)

Theorem 5.3.4 Covering a Board with Trominoes

For any integer $n \geq 1$, if one square is removed from a $2^n \times 2^n$ checkerboard, the remaining squares can be completely covered by L-shaped trominoes.

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.4 Strong Mathematical Induction and the Well-Ordering Principle for the Integers

Copyright © Cengage Learning. All rights reserved.

Strong Mathematical Induction and the Well-Ordering Principle for the Integers (1/2)

Strong mathematical induction is similar to ordinary mathematical induction in that it is a technique for establishing the truth of a sequence of statements about integers. Also, a proof by strong mathematical induction consists of a basis step and an inductive step.

However, the basis step may contain proofs for several initial values, and in the inductive step the truth of the predicate $P(n)$ is assumed not just for one value of n but for *all* values through k, and then the truth of $P(k + 1)$ is proved.

Strong Mathematical Induction and the Well-Ordering Principle for the Integers (2/2)

Principle of Strong Mathematical Induction

Let $P(n)$ be a property that is defined for integers n, and let a and b be fixed integers with $a \leq b$. Suppose the following two statements are true:

1. $P(a), P(a+1), \ldots,$ and $P(b)$ are all true. **(basis step)**
2. For every integer $k \geq b$, if $P(i)$ is true for each integer i from a through k, then $P(k+1)$ is true. **(inductive step)**

Then the statement

$$\text{for every integer } n \geq a,\ P(n)$$

is true. (The supposition that $P(i)$ is true for each integer i from a through k is called the **inductive hypothesis**. Another way to state the inductive hypothesis is to say that $P(a), P(a+1), \ldots, P(k)$ are all true.)

Applying Strong Mathematical Induction

Applying Strong Mathematical Induction (1/3)

The divisibility-by-a-prime theorem states that any integer greater than 1 is divisible by a prime number.

We prove this theorem using strong mathematical induction.

Example 5.4.1 – *Divisibility by a Prime*

Prove Theorem 4.4.4: Any integer greater than 1 is divisible by a prime number.

Theorem 4.4.4 Divisibility by a Prime

Any integer $n > 1$ is divisible by a prime number.

Example 5.4.1 – *Solution (1/5)*

The idea for the inductive step is this: If you are given an integer greater than 1 that is not itself prime, then it is a product of two smaller positive integers, each of which is greater than 1.

By inductive hypothesis, you are assuming that each of these smaller integers is divisible by a prime number, and so, by transitivity of divisibility, those prime numbers also divide the integer you started with.

Example 5.4.1 – *Solution (2/5)* continued

Proof (by strong mathematical induction):

Let the property $P(n)$ be the sentence

n is divisible by a prime number. $\leftarrow P(n)$

Show that P(2) is true:

To establish $P(2)$, we must show that

2 is divisible by a prime number. $\leftarrow P(2)$

But this is true because 2 is divisible by 2 and 2 is a prime number.

Example 5.4.1 – *Solution (3/5)* continued

Show that for every integer $k \geq 2$, if $P(i)$ is true for each integer from 2 through k, then $P(k + 1)$ is also true:

Let k be any integer with $k \geq 2$ and suppose that

i is divisible by a prime number for each integer i from 2 through k. ← inductive hypothesis

We must show that

$k + 1$ is divisible by a prime number. ← $P(k + 1)$

Example 5.4.1 – *Solution (4/5)* continued

***Case 1* (*k* + 1 *is prime*):** In this case $k + 1$ is divisible by a prime number, namely, itself.

***Case 2* (*k* + 1 *is not prime*):** In this case $k + 1 = ab$ where a and b are integers with $1 < a < k + 1$ and $1 < b < k + 1$. Thus, in particular, $2 \leq a \leq k$, and so by inductive hypothesis, a is divisible by a prime number p.

In addition because $k + 1 = ab$, we have that $k + 1$ is divisible by a. Hence, since $k + 1$ is divisible by a and a is divisible by p, by transitivity of divisibility, $k + 1$ is divisible by the prime number p.

Example 5.4.1 – *Solution (5/5)* continued

Therefore, regardless of whether $k + 1$ *is prime or not, it is divisible by a prime number [as was to be shown].*

[Since we have proved both the basis and the inductive step of the strong mathematical induction, we conclude that the given statement is true.]

Example 5.4.2 – *Proving a Property of a Sequence with Strong Induction*

Define a sequence $s_0, s_1, s_2, \ldots$ as follows:

$s_0 = 0$, $s_1 = 4$, $s_k = 6a_{k-1} - 5a_{k-2}$ for every integer $k \geq 2$.

a. Find the first four terms of this sequence.

b. It is claimed that for each integer $n \geq 0$, the nth term of the sequence has the same value as that given by the formula $5^n - 1$. In other words, the claim is that all the terms of the sequence satisfy the equation $s_n = 5^n - 1$. Prove that this is true.

Example 5.4.2 – *Solution (1/7)*

a. $s_0 = 0$,

$s_1 = 4$,

$s_2 = 6s_1 - 5s_0 = 6 \cdot 4 - 5 \cdot 0 = 24$,

$s_3 = 6s_2 - 5s_1 = 6 \cdot 24 - 5 \cdot 4 = 144 - 20 = 124$

b. To use strong mathematical induction to show that every term of the sequence satisfies the equation, the basis step must show that the first two terms satisfy it. This is necessary because, according to the definition of the sequence, computing values of later terms requires knowing the values of the *two* previous terms.

Example 5.4.2 – *Solution (2/7)* continued

So if the basis step only shows that the first term satisfies the equation, it would not be possible to use the inductive step to deduce that the second term satisfies the equation.

In the inductive step you suppose that for an arbitrarily chosen integer $k \geq 1$, all the terms of the sequence from s_0 through s_k satisfy the given equation and you then deduce that s_{k+1} must also satisfy the equation.

Example 5.4.2 – *Solution (3/7)* continued

Proof:

Let $s_0, s_1, s_2, \ldots$ be the sequence defined by specifying that $s_0 = 0$, $s_1 = 4$, and $s_k = 6a_{k-1} - 5a_{k-2}$ for every integer $k \geq 2$, and let the property $P(n)$ be the formula

$$s_n = 5^n - 1. \qquad \leftarrow P(n)$$

We will use strong mathematical induction to prove that for every integer $n \geq 0$, $P(n)$ is true.

Example 5.4.2 – *Solution (4/7)* continued

Show that *P*(0) and *P*(1) are true:

To establish *P*(0) and *P*(1), we must show that

$$s_0 = 5^0 - 1 \text{ and } s_1 = 5^1 - 1.$$ ← *P*(0) and *P*(1)

But, by definition of s_0, s_1, s_2, … , we have that $s_0 = 0$ and $s_1 = 4$.

Since $5^0 - 1 = 1 - 1 = 0 \text{ and } 5^1 - 1 = 5 - 1 = 4,$ the values of s_0 and s_1 agree with the values given by the formula.

Example 5.4.2 – *Solution (5/7)* continued

Show that for every integer k ≥ 1, if P(i) is true for each integer i from 0 through k, then P(k + 1) is also true:

Let k be any integer with $k \geq 1$ and suppose that

$$s_i = 5^i - 1 \text{ for each integer } i \text{ with } 0 \leq i \leq k.$$ ← inductive hypothesis

We must show that

$$s_{k+1} = 5^{k+1} - 1.$$ ← $P(k + 1)$

Example 5.4.2 – *Solution (6/7)* continued

But since $k \geq 1$, we have that $k + 1 \geq 2$, and so

$s_{k+1} = 6s_k - 5s_{k-1}$	by definition of s_0, s_1, s_2, …
$= 6(5^k - 1) - 5(5^{k-1} - 1)$	by definition hypothesis
$= 6 \cdot 5^k - 6 - 5^k + 5$	by multiplying out and applying a law of exponents
$= (6 - 1)5^k - 1$	by factoring out 6 and arithmetic

Example 5.4.2 – *Solution (7/7)* continued

$$= 5 \cdot 5^k - 1$$ by arithmetic

$$= 5^{k+1} - 1$$ by applying a law of exponents,

[as was to be shown].

[Since we have proved both the basis and the inductive step of the strong mathematical induction, we conclude that the given statement is true.]

Example 5.4.3 – *A Sequence That Involves the Floor Function*

Define a sequence $a_1, a_2, a_3, \ldots$ as follows:

$a_1 = 0,$

$a_2 = 2,$

$a_k = 3a_{\lfloor k/2 \rfloor} + 2$ for every integer $k \geq 3$.

a. Find the first seven terms of the sequence.

b. Prove that a_n is even for every integer $n \geq 1$.

Example 5.4.3 – *Solution (1/4)*

a. $a_1 = 0$

$a_2 = 2$

$a_3 = 3a_{\lfloor 3/2 \rfloor} + 2 = 3a_1 + 2 = 3 \cdot 0 + 2 = 2$

$a_4 = 3a_{\lfloor 4/2 \rfloor} + 2 = 3a_2 + 2 = 3 \cdot 2 + 2 = 8$

$a_5 = 3a_{\lfloor 5/2 \rfloor} + 2 = 3a_2 + 2 = 3 \cdot 2 + 2 = 8$

$a_6 = 3a_{\lfloor 6/2 \rfloor} + 2 = 3a_3 + 2 = 3 \cdot 2 + 2 = 8$

$a_7 = 3a_{\lfloor 7/2 \rfloor} + 2 = 3a_3 + 2 = 3 \cdot 2 + 2 = 8$

Example 5.4.3 – *Solution (2/4)* continued

b. Let the property $P(n)$ be the sentence "a_n is even." We use strong mathematical induction to show that the property holds for every integer $n \geq 1$.

Show that P*(1) *and P*(2) *are true: The property is true for $n = 1$ and $n = 2$ because $a_1 = 0$ and $a_2 = 2$ and both 0 and 2 are even integers.

Show that for every integer k* ≥ 1, *if P*(*i*) *is true for each integer i from* 1 *through k, then P*(*k*+1) *is also true: Let k be any integer with $k \geq 1$ and suppose that

a_i is even for each integer i with $1 \leq i \leq k$. ← inductive hypothesis

Example 5.4.3 – *Solution (3/4)* continued

[We must show that a_k is even.] By definition of a_1, a_2, a_3, ...

$$a_k = 3a_{\lfloor k/2 \rfloor} + 2 \qquad \text{for every integer } k \geq 3. \qquad \leftarrow P(k+1)$$

Now $a_{\lfloor k/2 \rfloor}$ is even by the inductive hypothesis *[because $k \geq 1$ and so $1 \leq \lfloor k/2 \rfloor \leq k$].* Thus $3a_{\lfloor k/2 \rfloor}$ is even *[because it is a product of an odd and an even integer]*, and hence $3a_{\lfloor k/2 \rfloor} + 2$ is even *[because a sum of two even integers is even]*. Consequently, a_k, which equals $3a_{\lfloor k/2 \rfloor} + 2$, is even *[as was to be shown]*.

Example 5.4.3 – *Solution (4/4)* continued

[Since we have proved the basis step and the inductive step of the strong mathematical induction, we conclude that the given statement is true.]

Applying Strong Mathematical Induction (2/3)

Convention

Let us agree to say that a single number x_1 is a product with one factor and can be computed with zero multiplications.

Example 5.4.4 – *The Number of Multiplications Needed to Multiply n Numbers*

Prove that for any integer $n \geq 1$, if $x_1, x_2, \ldots, x_n$ are n numbers, then no matter how the parentheses are inserted into their product, the number of multiplications used to compute the product is $n - 1$.

Example 5.4.4 – *Solution (1/7)*

The truth of the basis step follows immediately from the convention about a product with one factor. The inductive step is based on the fact that when several numbers are multiplied together, each step of the process involves multiplying two individual quantities.

For instance, the final step for computing $((x_1x_2)x_3)(x_4x_5)$ is to multiply $(x_1x_2)x_3$ and x_4x_5. In general, when $k + 1$ numbers are multiplied, the two quantities in the final step each consist of fewer than $k + 1$ factors. This is what makes it possible to use the inductive hypothesis.

Example 5.4.4 – *Solution (2/7)* continued

Proof (by strong mathematical induction):

Let the property $P(n)$ be the sentence

> If $x_1, x_2, \ldots, x_n$ are n numbers, then no matter how parentheses are inserted into their product, the number of multiplications used to compute the product is $n - 1$. $\leftarrow P(n)$

Example 5.4.4 – *Solution (3/7)* continued

Show that P(*1*) is true:

To establish $P(1)$, we must show that

The number of multiplications needed to compute the product of x_1 is $1 - 1$. $\leftarrow P(1)$

This is true because, by convention, x_1 is a product that can be computed with 0 multiplications, and $0 = 1 - 1$.

Example 5.4.4 – *Solution (4/7)* continued

Show that for every integer $k \geq 1$, if $P(i)$ is true for each integer i from 1 through k, then $P(k + 1)$ is also true:

Let k be any integer with $k \geq 1$ and suppose that

> For each integer i from 1 through k, if x_1, x_2, … , x_i are numbers, then no matter how parentheses are inserted into their product, the number of multiplications used to compute the product is $i - 1$.

← inductive hypothesis

Example 5.4.4 – *Solution (5/7)* continued

We must show that

> If $x_1, x_2, \ldots, x_{k+1}$ are $k + 1$ numbers, then no matter how parentheses are inserted into their product, the number of multiplications used to compute the product is $(k + 1) - 1 = k$. $\leftarrow P(k + 1)$

Consider a product of $k + 1$ factors: $x_1, x_2 \ldots, x_{k+1}$. When parentheses are inserted in order to compute the product, some multiplication is the final one and each of the two factors making up the final multiplication is a product of fewer than $k + 1$ factors.

Example 5.4.4 – *Solution (6/7)* continued

Let L be the product of the left-hand factors and R be the product of the right-hand factors, and suppose that L is composed of l factors and R is composed of r factors.

Then $l + r = k + 1$, the total number of factors in the product, and

$$1 \leq l \leq k \text{ and } 1 \leq r \leq k.$$

Example 5.4.4 – *Solution (7/7)* continued

By inductive hypothesis, evaluating L takes $l - 1$ multiplications and evaluating R takes $r - 1$ multiplications. Because one final multiplication is needed to evaluate $L \cdot R$, the number of multiplications needed to evaluate the product of all $k + 1$ factors is

$$(l - 1) + (r - 1) + 1 = (l + r) - 1 = (k + 1) - 1 = k$$

[as was to be shown].

[Since we have proved the basis step and the inductive step of the strong mathematical induction, we conclude that the given statement is true.]

Applying Strong Mathematical Induction (3/3)

Theorem 5.4.1 Existence and Uniqueness of Binary Integer Representations

Given any positive integer n, n has a unique representation in the form

$$n = c_r \cdot 2^r + c_{r-1} \cdot 2^{r-1} + \cdots + c_2 \cdot 2^2 + c_1 \cdot 2 + c_0,$$

where r is a nonnegative integer, $c_r = 1$, and $c_j = 1$ or 0 for each $j = 0, 1, 2, \ldots, r-1$.

The Well-Ordering Principle for the Integers

The Well-Ordering Principle for the Integers (1/2)

The well-ordering principle for the integers looks very different from both the ordinary and the strong principles of mathematical induction, but it can be shown that all three principles are equivalent. In other words, if any one of the three is true, then so are both of the others.

Well-Ordering Principle for the Integers

Let S be a set of integers containing one or more integers all of which are greater than some fixed integer. Then S has a least element.

Example 5.4.5 – *Finding Least Elements*

In each case, if the set has a least element, state what it is. If not, explain why the well-ordering principle is not violated.

a. The set of all positive real numbers

b. The set of all nonnegative integers n such that $n^2 < n$

c. The set of all nonnegative integers of the form $46 - 7k$, where k is an integer

Example 5.4.5 – *Solution (1/2)*

a. There is no least positive real number. For if x is any positive real number, then $x/2$ is a positive real number that is less than x. No violation of the well-ordering principle occurs because the well-ordering principle refers only to sets of integers, and this set is not a set of integers.

b. There is no *least* nonnegative integer n such that $n^2 < n$ because there is *no* nonnegative integer that satisfies this inequality. The well-ordering principle is not violated because the well-ordering principle refers only to sets that contain at least one element.

Example 5.4.5 – *Solution (2/2)*

continued

c. The following table shows values of $46 - 7k$ for various values of k.

k	0	1	2	3	4	5	6	7	$\cdots$	-1	-2	-3	$\cdots$
$46 - 7k$	46	39	32	25	18	11	4	-3	$\cdots$	53	60	67	$\cdots$

The table suggests, and you can easily confirm, that $46 - 7k < 0$ for $k \geq 7$ and that $46 - 7k \geq 46$ for $k \leq 0$. Therefore, from the other values in the table it is clear that 4 is the least nonnegative integer of the form $46 - 7k$. This corresponds to $k = 6$.

The Well-Ordering Principle for the Integers (2/2)

Quotient-Remainder Theorem (Existence Part)

Given any integer n and any positive integer d, there exist integers q and r such that

$$n = dq + r \quad \text{and} \quad 0 \leq r < d.$$

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.5 Application: Correctness of Algorithms

Copyright © Cengage Learning. All rights reserved.

Application: Correctness of Algorithms

What does it mean for a computer program to be correct? Each program is designed to do a specific task—calculate the mean or median of a set of numbers, compute the size of the paychecks for a company payroll, rearrange names in alphabetical order, and so forth.

We will say that a program is correct if it produces the output specified in its accompanying documentation for each set of input data of the type specified in the documentation.

Assertions

Assertions (1/1)

Consider an algorithm that is designed to produce a certain final state from a certain initial state. Both the initial and final states can be expressed as predicates involving the input and output variables.

Often the predicate describing the initial state is called the **pre-condition for the algorithm**, and the predicate describing the final state is called the **post-condition for the algorithm**.

Example 5.5.1 – *Algorithm Pre-Conditions and Post-Conditions (1/3)*

Here are pre- and post-conditions for some typical algorithms.

a. Algorithm to compute a product of nonnegative integers

Pre-condition: The input variables m and n are nonnegative integers.

Post-condition: The output variable p equals mn.

Example 5.5.1 – *Algorithm Pre-Conditions and Post-Conditions (2/3)*

b. Algorithm to find quotient and remainder of the division of one positive integer by another

Pre-condition: The input variables a and b are positive integers.

Post-condition: The output variables q and r are integers such that $a = bq + r$ and $0 \leq r < b$.

Example 5.5.1 – *Algorithm Pre-Conditions and Post-Conditions (3/3)*

c. Algorithm to sort a one-dimensional array of real numbers

Pre-condition: The input variable $A[1], A[2], \ldots, A[n]$ is a one-dimensional array of real numbers.

Post-condition: The output variable $B[1], B[2], \ldots, B[n]$ is a one-dimensional array of real numbers with same elements as $A[1], A[2], \ldots, A[n]$ but with the property that $B[i] \leq B[j]$ whenever $i \leq j$.

Loop Invariants

Loop Invariants (1/5)

The method of loop invariants is used to prove correctness of a loop with respect to certain pre- and post-conditions.

It is based on the principle of mathematical induction. Suppose that an algorithm contains a **while** loop and that entry to this loop is restricted by a condition *G*, called the **guard**.

Suppose also that assertions describing the current states of algorithm variables have been placed immediately preceding and immediately the loop.

Loop Invariants (2/5)

The assertion just preceding the loop is called the **pre-condition for the loop** and the one just following is called the **post-condition for the loop**. The annotated loop has the following appearance:

[*Pre-condition for the loop*]

while (*G*)

[*Statements in the body of the loop*. *None contain branching statements that lead outside the loop*.]

end while

[*Post-condition for the loop*]

Loop Invariants (3/5)

Definition

A loop is defined as **correct with respect to its pre- and post-conditions** if, and only if, whenever the algorithm variables satisfy the pre-condition for the loop and the loop terminates after a finite number of steps, the algorithm variables satisfy the post-condition for the loop.

A **loop invariant** is a predicate with domain a set of integers, which satisfies the following condition: For each iteration of the loop, if the predicate is true before the iteration, then it is true after the iteration.

Loop Invariants (4/5)

Furthermore, if the predicate satisfies the following two additional conditions, the loop will be correct with respect to its pre- and post-conditions:

1. The predicate is true before the first iteration of the loop.

2. If the loop terminates after a finite number of iterations, the truth of the loop invariant ensures the truth of the post-condition for the loop.

Loop Invariants (5/5)

The following theorem, called the *loop invariant theorem*, formalizes these ideas. It was first developed by C. A. R. Hoare in 1969.

Theorem 5.5.1 Loop Invariant Theorem

Let a **while** loop with guard G be given, together with pre- and post-conditions that are predicates in the algorithm variables. Also let a predicate $I(n)$, called the **loop invariant**, be given. If the following four properties are true, then the loop is correct with respect to its pre- and post-conditions.

I. **Basis Property:** The pre-condition for the loop implies that $I(0)$ is true before the first iteration of the loop.

II. **Inductive Property:** For every integer $k \geq 0$, if the guard G and the loop invariant $I(k)$ are both true before an iteration of the loop, then $I(k+1)$ is true after an iteration of the loop.

III. **Eventual Falsity of Guard:** After a finite number of iterations of the loop, the guard G becomes false.

IV. **Correctness of the Post-Condition:** If N is the least number of iterations after which G is false and $I(N)$ is true, then the values of the algorithm variables will be as specified in the post-condition of the loop.

Example 5.5.2 – *Correctness of a Loop to Compute a Product (1/3)*

The loop is designed to compute the product mx for a nonnegative integer m and a real number x, without using a built-in multiplication operation.

Prior to the loop, variables i and *product* have been introduced and given initial values $i = 0$ and *product* = 0.

Example 5.5.2 – *Correctness of a Loop to Compute a Product (2/3)*

[Pre-condition: m is a nonnegative integer, x is a real number, i = 0*, and product* = 0*.]*

while *(i ≠ m)*

1. product := *product* + *x*

2. i := *i* + 1

end while

[Post-condition: product = mx]

Example 5.5.2 – *Correctness of a Loop to Compute a Product (3/3)*

Let the loop invariant be

$$I(n)\colon i = n \quad \text{and} \quad product = nx$$

The guard condition *G* of the **while** loop is

$$G\colon i \neq m$$

Use the loop invariant theorem to prove that the **while** loop is correct with respect to the given pre- and post-conditions.

Example 5.5.2 – *Solution (1/5)*

I. **Basis Property:** *[I(0) is true before the first iteration of the loop.]*

$I(0)$ is "$i = 0$ and $product = 0 \cdot x$," which is true before the first iteration of the loop because $0 \cdot x = 0$.

II. **Inductive Property:** *[If $G \wedge I(k)$ is true before a loop iteration (where $k \geq 0$), then $I(k + 1)$ is true after the loop iteration.]*

Suppose k is a nonnegative integer such that $G \wedge I(k)$ is true before an iteration of the loop.

Example 5.5.2 – *Solution (2/5)* continued

Then as execution reaches the top of the loop, $i \neq m$, $product = kx$, and $i = k$. Since $i \neq m$, the guard is passed and statement 1 is executed.

Before execution of statement 1,

$$\text{product}_{\text{old}} = kx.$$

Thus execution of statement 1 has the following effect:

$$\text{product}_{\text{new}} = \text{product}_{\text{old}} + x = kx + x = (k + 1)x.$$

Example 5.5.2 – *Solution (3/5)* continued

Similarly, before statement 2 is executed,

$$i_{\text{old}} = k,$$

so after execution of statement 2,

$$i_{\text{new}} = i_{\text{old}} + 1 = k + 1.$$

Hence after the loop iteration, the statement $I(k + 1)$, namely, ($i = k + 1$ and $product = (k + 1)x$), is true. This is what we needed to show.

Example 5.5.2 – *Solution (4/5)* continued

III. **Eventual Falsity of Guard:** *[After a finite number of iterations of the loop, G becomes false.]*

The guard G is the condition $i \neq m$, and m is a nonnegative integer. By I and II, it is known that

for every integer $n \geq 0$, if the loop is iterated n times, then $i = n$ and $product = nx$.

So after m iterations of the loop, $i = m$. Thus G becomes false after m iterations of the loop.

Example 5.5.2 – *Solution (5/5)* continued

IV. **Correctness of the Post-Condition:** *[If N is the least number of iterations after which G is false and I(N) is true, then the value of the algorithm variables will be as specified in the post-condition of the loop.]*

According to the post-condition, the value of *product* after execution of the loop should be *mx*. But if *G* becomes false after *N* iterations, $i = m$. And if $I(N)$ is true, $i = N$ and $product = Nx$. Since both conditions (*G* false and $I(N)$ true) are satisfied, $m = i = N$ and $product = mx$ as required.

Correctness of the Division Algorithm

Correctness of the Division Algorithm (1/2)

The division algorithm is supposed to take a nonnegative integer a and a positive integer d and compute nonnegative integers q and r such that $a = dq + r$ and $0 \leq r < d$.

Initially, the variables r and q are introduced and given the values $r = a$ and $q = 0$.

Correctness of the Division Algorithm (2/2)

The crucial loop, annotated with pre- and post-conditions, is the following:

[Pre-condition: a is a nonnegative integer and d is a positive integer, r = a, and q = 0.]

while ($r \geq d$)

1. $r := r - d$
2. $q := q + 1$

end while

[Post-condition: q and r are nonnegative integers with the property that $a = qd + r$ and $0 \leq r < d$.]

Correctness of the Euclidean Theorem

Correctness of the Euclidean Theorem (1/2)

The Euclidean algorithm is supposed to take integers A and B with $A > B \geq 0$ and compute their greatest common divisor.

Just before the crucial loop, variables a, b, and r have been introduced with $a = A$, $b = B$, and $r = B$.

Correctness of the Euclidean Theorem (2/2)

The crucial loop, annotated with pre- and post-conditions, is the following:

[Pre-condition: A and B are integers with $A > B \geq 0$*,* $a = A$*,* $b = B$*,* $r = B$*.]*

while ($b \neq 0$)

1. $r := a \bmod b$
2. $a := b$
3. $b := r$

end while

[Post-condition: $a = \gcd(A, B)$*.]*

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.6 Defining Sequences Recursively

Copyright © Cengage Learning. All rights reserved.

Defining Sequences Recursively (1/2)

A sequence can be defined in a variety of different ways. One informal way is to write the first few terms with the expectation that the general pattern will be obvious.

We might say, for instance, “consider the sequence 3, 5, 7, … .” Unfortunately, misunderstandings can occur when this approach is used.

The next term of the sequence could be 9 if we mean a sequence of odd integers, or it could be 11 if we mean the sequence of odd prime numbers.

Defining Sequences Recursively (2/2)

The second way to define a sequence is to give an explicit formula for its *n*th term. For example, a sequence $a_0, a_1, a_2 \ldots$ can be specified by writing

$$a_n = \frac{(-1)^n}{n+1} \quad \text{for every integer } n \geq 0.$$

The advantage of defining a sequence by such an explicit formula is that each term of the sequence is uniquely determined and can be computed in a fixed, finite number of steps by substitution.

The third way to define a sequence is to use recursion.

Defining Sequences Recursively

This requires giving both an equation, called a *recurrence relation*, that defines each later term in the sequence by reference to earlier terms and also one or more initial values for the sequence. Sometimes it is very difficult or impossible to find an explicit formula for a sequence, but it *is* possible to define the sequence using recursion.

Definition

A **recurrence relation** for a sequence $a_0, a_1, a_2, \ldots$ is a formula that relates each term a_k to certain of its predecessors $a_{k-1}, a_{k-2}, \ldots, a_{k-i}$, where i is an integer with $k - i \geq 0$. If i is a fixed integer, the **initial conditions** for such a recurrence relation specify the values of $a_0, a_1, a_2, \ldots, a_{i-1}$. If i depends on k, the initial conditions specify the values of $a_0, a_1, \ldots, a_m$, where m is an integer with $m \geq 0$.

Example 5.6.1 – *Computing Terms of a Recursively Defined Sequence*

Define a sequence $c_0, c_1, c_2, \ldots$ recursively as follows: For every integer $k \geq 2$,

(1) $c_k = c_{k-1} + kc_{k-2} + 1$ recurrence relation

(2) $c_0 = 1$ and $c_1 = 2$ initial conditions

Find c_2, c_3, and c_4.

Example 5.6.1 – *Solution (1/2)*

$c_2 = c_1 + 2c_0 + 1$ by substituting $k = 2$ into (1)

$= 2 + 2 \cdot 1 + 1$ since $c_1 = 2$ and $c_0 = 1$ by (2)

(3) $\therefore\ c_2 = 5$

$c_3 = c_2 + 3c_1 + 1$ by substituting $k = 3$ into (1)

$= 5 + 3 \cdot 2 + 1$ since $c_2 = 5$ by (3) and $c_1 = 2$ by (2)

Example 5.6.1 – *Solution (2/2)*

continued

(4) $\therefore\ c_3 = 12$

$c_4 = c_3 + 4c_2 + 1$ by substituting $k = 4$ into (1)

$= 12 + 4 \cdot 5 + 1$ since $c_3 = 12$ by (4) and $c_2 = 5$ by (3)

(5) $\therefore\ c_4 = 33$

Example 5.6.2 – *Writing a Recurrence Relation in More Than One Way*

Let $s_0, s_1, s_2, \ldots$ be a sequence that satisfies the following recurrence relation:

For every integer $k \geq 1$, $\quad s_k = 3s_{k-1} - 1.$

Explain why the following statement is true:

For every integer $k \geq 0$, $\quad s_{k+1} = 3s_k - 1.$

Example 5.6.2 – *Solution*

In informal language, the recurrence relation says that any term of the sequence equals 3 times the previous term minus 1.

Now for any integer $k \geq 0$, the term previous to s_{k+1} is s_k.

Thus for any integer $k \geq 0$, $s_{k+1} = 3s_k - 1$.

Example 5.6.3 – *Sequences That Satisfy the Same Recurrence Relation*

Let $a_1, a_2, a_3, \ldots$ and $b_1, b_2, b_3, \ldots$ satisfy the recurrence relation that the kth term equals 3 times the $(k - 1)$st term for every integer $k \geq 2$:

(1) $a_k = 3a_{k-1}$ and $b_k = 3b_{k-1}$.

But suppose that the initial conditions for the sequences are different:

(2) $a_1 = 2$ and $b_1 = 1$.

Find (a) a_2, a_3, a_4 and (b) b_2, b_3, b_4.

Example 5.6.3 – *Solution*

a. $a_2 = 3a_1 = 3 \cdot 2 = 6$

$a_3 = 3a_2 = 3 \cdot 6 = 18$

$a_4 = 3a_3 = 3 \cdot 18 = 54$

b. $b_2 = 3b_1 = 3 \cdot 1 = 3$

$b_3 = 3b_2 = 3 \cdot 3 = 9$

$b_4 = 3b_3 = 3 \cdot 9 = 27$

Thus

$a_1, a_2, a_3, \ldots$ begins 2, 6, 18, 54,… and

$b_1, b_2, b_3, \ldots$ begins 1, 3, 9, 27,….

Example 5.6.4 – *Showing That a Sequence Given by an Explicit Formula Satisfies a Certain Recurrence Relation*

The sequence of **Catalan numbers**, named after the Belgian mathematician Eugène Catalan (1814–1894), arises in a remarkable variety of different contexts in discrete mathematics. It can be defined as follows: For each integer $n \geq 1$,

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

a. Find C_1, C_2, and C_3.

b. Show that this sequence satisfies the recurrence relation $C_k = \frac{4k-2}{k+1}C_{k-1}$ for every integer $k \geq 2$.

Example 5.6.4 – *Solution (1/4)*

a. $C_1 = \frac{1}{2}\binom{2}{1} = \frac{1}{2}\cdot 2 = 1,$

$$C_2 = \frac{1}{3}\binom{4}{2} = \frac{1}{3}\cdot 6 = 2,$$

$$C_3 = \frac{1}{4}\binom{6}{3} = \frac{1}{4}\cdot 20 = 5$$

Example 5.6.4 – *Solution (2/4)* continued

b. To obtain the kth and $(k-1)$st terms of the sequence, just substitute k and $k-1$ in place of n in the explicit formula for $C_1, C_2, C_3, \ldots$.

$$C_k = \frac{1}{k+1}\binom{2k}{k}$$

$$C_{k-1} = \frac{1}{(k-1)+1}\binom{2(k-1)}{k-1}$$

$$= \frac{1}{k}\binom{2k-2}{k-1}.$$

Example 5.6.4 – *Solution (3/4)* continued

Then start with the right-hand side of the recurrence relation and transform it into the left-hand side: For each integer $k \geq 2$,

$$\frac{4k-2}{k+1}C_{k-1} = \frac{4k-2}{k+1}\left[\frac{1}{k}\binom{2k-2}{k-1}\right]$$ by substituting

$$= \frac{2(2k-1)}{k+1}\cdot\frac{1}{k}\cdot\frac{(2k-2)!}{(k-1)!(2k-2-(k-1))!}$$ by the formula for *n* choose *r*

$$= \frac{1}{k+1}\cdot(2(2k-1))\cdot\frac{(2k-2)!}{(k(k-1)!)(k-1)!}$$ by rearranging the factors

Example 5.6.4 – *Solution (4/4)*

continued

$$= \frac{1}{k+1} \cdot (2(2k-1)) \cdot \frac{1}{k!(k-1)!} \cdot (2k-2)! \cdot \frac{1}{2} \cdot \frac{1}{k} \cdot 2k$$

because $\frac{1}{2} \cdot \frac{1}{k} \cdot 2k = 1$

$$= \frac{1}{k+1} \cdot \frac{2}{2} \cdot \frac{1}{k!} \cdot \frac{1}{(k-1)!} \cdot \frac{1}{k} \cdot (2k) \cdot (2k-1) \cdot (2k-2)!$$

by rearranging the factors

$$= \frac{1}{k+1} \cdot \frac{(2k)!}{k!k!}$$

because $k(k-1)! = k!$, $\frac{2}{2} = 1$, and $2k \cdot (2k-1) \cdot (2k-2)! = (2k)!$

$$= \frac{1}{k+1} \binom{2k}{k}$$

by the formula for *n* choose *r*

$$= C_k$$

by definition of C_1, C_2, C_3, …

Examples of Recursively Defined Sequences

Examples of Recursively Defined Sequences (1/1)

Figure 5.6.1

Example 5.6.6 – *The Fibonacci Numbers (1/2)*

One of the earliest examples of a recursively defined sequence occurs in the writings of Leonardo of Pisa, commonly known as Fibonacci, who was the greatest European mathematician of the Middle Ages and promoted the use of Hindu-Arabic numerals for calculation. In 1202 Fibonacci posed the following problem:

A single pair of rabbits (male and female) is born at the beginning of a year.

Example 5.6.6 – *The Fibonacci Numbers (2/2)*

Assume the following conditions:

1. Rabbit pairs are not fertile during their first month of life but thereafter give birth to one new male/female pair at the end of every month.

2. No rabbits die.

How many rabbits will there be at the end of the year?

Example 5.6.6 – *Solution (1/6)*

One way to solve this problem is to plunge right into the middle of it using recursion. Suppose you know how many rabbit pairs there were at the ends of previous months. How many will there be at the end of the current month?

The crucial observation is that the number of rabbit pairs born at the end of month k is the same as the number of pairs alive at the end of month $k - 2$. Why?

Because it is exactly the rabbit pairs that were alive at the end of month $k - 2$ that were fertile during month k.

Example 5.6.6 – *Solution (2/6)*

continued

The rabbits born at the end of month $k - 1$ were not.

Now the number of rabbit pairs alive at the end of month k equals the ones alive at the end of month $k - 1$ plus the pairs newly born at the end of the month. Thus

$$\begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs alive}\\ \text{at the end}\\ \text{of month } k\end{bmatrix} = \begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs alive}\\ \text{at the end}\\ \text{of month } k-1\end{bmatrix} + \begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs born}\\ \text{at the end}\\ \text{of month } k\end{bmatrix}$$

Example 5.6.6 – *Solution (3/6)*

continued

$$= \begin{bmatrix} \text{the number} \\ \text{of rabbit} \\ \text{pairs alive} \\ \text{at the end} \\ \text{of month } k-1 \end{bmatrix} + \begin{bmatrix} \text{the number} \\ \text{of rabbit} \\ \text{pairs alive} \\ \text{at the end} \\ \text{of month } k-2 \end{bmatrix}. \qquad 5.6.2$$

For each integer $n \geq 1$, let

$$F_n = \begin{bmatrix} \text{the number of rabbit pairs} \\ \text{alive at the end of month } n \end{bmatrix}$$

and let

$$F_0 = \text{the initial number of rabbit pairs}$$
$$= 1.$$

Example 5.6.6 – *Solution (4/6)* continued

Then by substitution into equation (5.6.2), for every integer $k \geq 2$,

$$F_k = F_{k-1} + F_{k-2}.$$

Now $F_0 = 1$, as already noted, and $F_1 = 1$ also, because the first pair of rabbits is not fertile until the second month.

Hence the complete specification of the Fibonacci sequence is as follows: For every integer $k \geq 2$,

(1) $F_k = F_{k-1} + F_{k-2}$ recurrence relation

(2) $F_0 = 1, \quad F_1 = 1$ initial conditions.

Example 5.6.6 – *Solution (5/6)* continued

To answer Fibonacci's question, compute F_2, F_3, and so forth through F_{12}:

(3)	$F_2 = F_1 + F_0 = 1 + 1$	$= 2$	by (1) and (2)
(4)	$F_3 = F_2 + F_1 = 2 + 1$	$= 3$	by (1), (2), and (3)
(5)	$F_4 = F_3 + F_2 = 3 + 2$	$= 5$	by (1), (3), and (4)
(6)	$F_5 = F_4 + F_3 = 5 + 3$	$= 8$	by (1), (4), and (5)
(7)	$F_6 = F_5 + F_4 = 8 + 5$	$= 13$	by (1), (5), and (6)
(8)	$F_7 = F_6 + F_5 = 13 + 8$	$= 21$	by (1), (6), and (7)

Example 5.6.6 – *Solution (6/6)* continued

(9) $F_8 = F_7 + F_6 = 21 + 13 = 34$ by (1), (7), and (8)

(10) $F_9 = F_8 + F_7 = 34 + 21 = 55$ by (1), (8), and (9)

(11) $F_{10} = F_9 + F_8 = 55 + 34 = 89$ by (1), (9), and (10)

(12) $F_{11} = F_{10} + F_9 = 89 + 55 = 144$ by (1), (10), and (11)

(13) $F_{12} = F_{11} + F_{10} = 144 + 89 = 233$ by (1), (11), and (12)

At the end of the twelfth month there are 233 rabbit pairs, or 466 rabbits in all.

Example 5.6.8 – *Compound Interest with Compounding Several Times a Year (1/3)*

When an annual interest rate of i is compounded m times per year, the interest rate paid per period is i/m.

(For instance, if 3% = 0.03 annual interest is compounded quarterly, then the interest rate paid per quarter is $0.03/4 = 0.0075$.)

For each integer $k \geq 1$, let P_k = the amount on deposit at the end of the kth period, assuming no additional deposits or withdrawals.

Example 5.6.8 – *Compound Interest with Compounding Several Times a Year (2/3)*

Then the interest earned during the *k*th period equals the amount on deposit at the end of the (*k* − 1)st period times the interest rate for the period:

$$\text{interest earned during } k\text{th period} = P_{k-1}\left(\frac{i}{m}\right).$$

The amount on deposit at the end of the *k*th period, P_k, equals the amount at the end of the (*k* − 1)st period, P_{k-1}, plus the interest earned during the *k*th period:

$$P_k = P_{k-1} + P_{k-1}\left(\frac{i}{m}\right) = P_{k-1}\left(1 + \frac{i}{m}\right). \qquad 5.6.4$$

Example 5.6.8 – *Compound Interest with Compounding Several Times a Year (3/3)*

Suppose $10,000 is left on deposit at 3% compounded quarterly.

a. How much will the account be worth at the end of one year, assuming no additional deposits or withdrawals?

b. The **annual percentage yield (APY)** is the percentage increase in the value of the account over a one-year period. What is the APY for this account?

Example 5.6.8 – *Solution (1/3)*

a. For each integer $n \geq 1$, let P_n = the amount on deposit after n consecutive quarters, assuming no additional deposits or withdrawals, and let P_0 be the initial \$10,000.

Then by equation (5.6.4) with $i = 0.03$ and $m = 4$, a recurrence relation for the sequence $P_0, P_1, P_2, \ldots$ is

(1) $P_k = P_{k-1}(1 + 0.0075) = (1.0075) \cdot P_{k-1}$

for every integer $k \geq 1$.

Example 5.6.8 – *Solution (2/3)* continued

The amount on deposit at the end of one year (four quarters), P_4, can be found by successive substitution:

(2) $P_0 = \$10{,}000$

(3) $P_1 = 1.0075 \cdot P_0 = (1.0075) \cdot \$10{,}000.00 = \$10{,}075.00$

by (1) and (2)

(4) $P_2 = 1.0075 \cdot P_1 = (1.0075) \cdot \$10{,}075.00 \cong \$10{,}150.56$

by (1) and (3)

(5) $P_3 = 1.0075 \cdot P_2 \cong (1.0075) \cdot \$10{,}150.56 \cong \$10{,}226.69$

by (1) and (4)

Example 5.6.8 – *Solution (3/3)* continued

(6) $P_4 = 1.0075 \cdot P_3 \cong (1.0075) \cdot \$10{,}226.69 \cong \$10{,}303.39$

by (1) and (5)

Hence after one year there is \$10,303.39 (to the nearest cent) in the account.

b. The percentage increase in the value of the account, or APY, is

$$\frac{10{,}303.39 - 10{,}000}{10{,}000} = 0.03034 = 3.034\%.$$

Recursive Definitions of Sum and Product

Recursive Definitions of Sum and Product (1/1)

Addition and multiplication are called *binary* operations because only two numbers can be added or multiplied at a time. Careful definitions of sums and products of more than two numbers use recursion.

Definition

Given numbers $a_1, a_2, \ldots, a_n$, where n is a positive integer, the **summation from $i = 1$ to n of the a_i**, denoted $\sum_{i=1}^{n} a_i$, is defined as follows:

$$\sum_{i=1}^{1} a_i = a_1 \quad \text{and} \quad \sum_{i=1}^{n} a_i = \left(\sum_{i=1}^{n-1} a_i\right) + a_n, \quad \text{if } n > 1.$$

The **product from $i = 1$ to n of the a_i**, denoted $\prod_{i=1}^{n} a_i$, is defined by

$$\prod_{i=1}^{1} a_i = a_1 \quad \text{and} \quad \prod_{i=1}^{n} a_i = \left(\prod_{i=1}^{n-1} a_i\right) \cdot a_n, \quad \text{if } n > 1.$$

Example 5.6.9 – *A Sum of Sums*

Prove that for any positive integer n, if $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are real numbers, then

$$\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i.$$

Example 5.6.9 – *Solution (1/5)*

The proof is by mathematical induction. Let the property $P(n)$ be the equation

$$\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i. \qquad \leftarrow P(n)$$

We must show that $P(n)$ is true for every integer $n \geq 1$. We do this by mathematical induction on n.

Show that P*(1) *is true: To establish $P(1)$, we must show that

$$\sum_{i=1}^{1}(a_i + b_i) = \sum_{i=1}^{1} a_i + \sum_{i=1}^{1} b_i. \qquad \leftarrow P(1)$$

Example 5.6.9 – *Solution (2/5)* continued

Now

$$\sum_{i=1}^{1}(a_i + b_i) = a_1 + b_1 \qquad \text{by definition of } \Sigma$$

$$= \sum_{i=1}^{1} a_i + \sum_{i=1}^{1} b_i \qquad \text{also by definition of } \Sigma.$$

Hence $P(1)$ is true.

Example 5.6.9 – *Solution (3/5)* continued

Show that for every integer k ≥ 1, if P(k) is true then P(k + 1) is also true:

Suppose that k is any integer with $k \geq 1$ and that $a_1, a_2, \ldots, a_k, a_{k+1}$ and $b_1, b_2, \ldots, b_k, b_{k+1}$ are real numbers such that

$$\sum_{i=1}^{k}(a_i + b_i) = \sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i. \qquad \leftarrow P(k) \text{ inductive hypothesis}$$

We must show that

$$\sum_{i=1}^{k+1}(a_i + b_i) = \sum_{i=1}^{k+1} a_i + \sum_{i=1}^{k+1} b_i. \qquad \leftarrow P(k+1)$$

Example 5.6.9 – *Solution (4/5)* continued

[We will show that the left-hand side of this equation equals the right-hand side.]

Now the left-hand side of the equation is

$$\sum_{i=1}^{k+1}(a_i + b_i) = \sum_{i=1}^{k}(a_i + b_i) + (a_{k+1} + b_{k+1}) \qquad \text{by definition of } \Sigma$$

$$= \left(\sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i\right) + (a_{k+1} + b_{k+1}) \qquad \text{by inductive hypothesis}$$

Example 5.6.9 – *Solution (5/5)* continued

$$= \left(\sum_{i=1}^{k} a_i + a_{k+1}\right) + \left(\sum_{i=1}^{k} b_i + b_{k+1}\right)$$ by the associative and commutative laws of algebra

$$= \sum_{i=1}^{k+1} a_i + \sum_{i=1}^{k+1} b_i$$ by definition of Σ

which equals the right-hand side of the equation. *[This is what was to be shown.]*

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.7 Solving Recurrence Relations by Iteration

Copyright © Cengage Learning. All rights reserved.

Solving Recurrence Relations by Iteration

Suppose you have a sequence that satisfies a certain recurrence relation and initial conditions. It is often helpful to know an explicit formula for the sequence, especially if you need to compute terms with very large subscripts or if you need to examine general properties of the sequence. Such an explicit formula is called a **solution** to the recurrence relation. In this section, we discuss methods for solving recurrence relations.

The Method of Iteration

The Method of Iteration (1/4)

The most basic method for finding an explicit formula for a recursively defined sequence is **iteration**. Iteration works as follows: Given a sequence $a_0, a_1, a_2, \cdots$ defined by a recurrence relation and initial conditions, you start from the initial conditions and calculate successive terms of the sequence until you see a pattern developing. At that point you guess an explicit formula.

Example 5.7.1 – *Finding an Explicit Formula*

Let $a_0, a_1, a_2, \cdots$ be the sequence defined recursively as follows: For each integer $k \geq 1$,

(1) $a_k = a_{k-1} + 2$ recurrence relation

(2) $a_0 = 1$ initial condition.

Use iteration to guess an explicit formula for the sequence.

Example 5.7.1 – *Solution (1/5)*

Recall that to say

$$a_k = a_{k-1} + 2 \qquad \text{for each integer } k \geq 1$$

means

$$a_{\square} = a_{\square - 1} + 2 \qquad \text{no matter what positive integer is placed into the box } \square.$$

In particular,

$$a_1 = a_0 + 2,$$

$$a_2 = a_1 + 2,$$

$$a_3 = a_2 + 2,$$

and so forth.

Example 5.7.1 – *Solution (2/5)* continued

Now use the initial condition to begin a process of successive substitutions into these equations, not just of numbers but of *numerical expressions*.

Example 5.7.1 – *Solution (3/5)* continued

Here's how the process works for the given sequence:

$a_0 = 1$	the initial condition
$a_1 = a_0 + 2 = 1 + 2$	by substitution
$a_2 = a_1 + 2 = (1 + 2) + 2 \quad = 1 + 2 + 2$	eliminate parentheses
$a_3 = a_2 + 2 = (1 + 2 + 2) + 2 \quad = 1 + 2 + 2 + 2$	eliminate parentheses again; write $3 \cdot 2$ instead of $2 + 2 + 2$?
$a_4 = a_3 + 2 = (1 + 2 + 2 + 2) + 2 = 1 + 2 + 2 + 2 + 2$	eliminate parentheses again; definitely write $4 \cdot 2$ instead of $2 + 2 + 2 + 2$—the length of the string of 2's is getting out of hand.

Example 5.7.1 – *Solution (4/5)* continued

Since it appears helpful to use the shorthand $k \cdot 2$ in place of $2 + 2 + \cdots + 2$ (k times), we do so, starting again from a_0.

Example 5.7.1 – *Solution (5/5)*

continued

$$a_0 = 1 \qquad = 1 + 0 \cdot 2$$ the initial condition

$$a_1 = a_0 + 2 = 1 + 2 \qquad = 1 + 1 \cdot 2$$ by substitution

$$a_2 = a_1 + 2 = (1 + 2) + 2 \qquad = 1 + 2 \cdot 2$$

$$a_3 = a_2 + 2 = (1 + 2 \cdot 2) + 2 = 1 + 3 \cdot 2$$

$$a_4 = a_3 + 2 = (1 + 3 \cdot 2) + 2 = 1 + 4 \cdot 2$$

At this point it certainly seems likely that the general pattern is $1 + n \cdot 2$; check whether the next calculation supports this.

$$a_5 = a_4 + 2 = (1 + 4 \cdot 2) + 2 = 1 + 5 \cdot 2$$

It does! So go ahead and write an answer. It's only a guess, after all.

$$\vdots$$

Guess: $a_n = 1 + n \cdot 2 = 1 + 2n$ for every integer n.

The Method of Iteration (2/4)

A sequence like the one in which each term equals the previous term plus a fixed constant, is called an *arithmetic sequence*.

Definition

A sequence $a_0, a_1, a_2, \ldots$ is called an **arithmetic sequence** if, and only if, there is a constant d such that

$$a_k = a_{k-1} + d \quad \text{for each integer } k \geq 1.$$

It follows that

$$a_n = a_0 + dn \quad \text{for every integer } n \geq 0.$$

Example 5.7.2 – *An Arithmetic Sequence*

Under the force of gravity, an object falling in a vacuum falls about 9.8 meters per second (m/sec) faster each second than it fell the second before. Thus, neglecting air resistance, a skydiver's speed upon leaving an airplane is approximately 9.8 m/sec one second after departure, 9.8 + 9.8 = 19.6 m/sec two seconds after departure, and so forth. If air resistance is neglected, how fast would the skydiver be falling 60 seconds after leaving the airplane?

Example 5.7.2 – *Solution (1/2)*

Let s_n be the skydiver's speed in m/sec n seconds after exiting the airplane, assuming there is no air resistance. Then s_0 is the initial speed, and since the diver would travel 9.8 m/sec faster each second than the second before,

$$s_k = s_{k-1} + 9.8 \text{ m/sec} \quad \text{for every integer } k \geq 1.$$

It follows that $s_0, s_1, s_2, \cdots$ is an arithmetic sequence with a fixed constant of 9.8, and thus

$$s_n = s_0 + (9.8)n \quad \text{for each integer } n \geq 0.$$

Example 5.7.2 – *Solution (2/2)*

continued

Hence sixty seconds after exiting and neglecting air resistance, the skydiver would travel at a speed of

$$s_{60} = 0 + (9.8)(60) = 588 \text{ m/sec}$$

Now 588 m/sec is over half a kilometer per second or over a third of a mile per second, which is very fast for a human being to travel. Fortunately for the skydiver, taking air resistance into account reduces the speed considerably.

Example 5.7.3 – *The Explicit Formula for a Geometric Sequence*

Let r be a fixed nonzero constant, and suppose a sequence $a_0, a_1, a_2, \cdots$ is defined recursively as follows:

$$a_k = ra_{k-1} \text{ for each integer } k \geq 1,$$
$$a_0 = a.$$

Use iteration to guess an explicit formula for this sequence.

Example 5.7.3 – *Solution*

$$a_0 = a$$

$$a_1 = ra_0 = ra$$

$$a_2 = ra_1 = r(ra) = r^2a$$

$$a_3 = ra_2 = r(r^2a) = r^3a$$

$$a_4 = ra_3 = r(r^3a) = r^4a$$

$$\vdots$$

Guess: $a_n = r^n a = ar^n$ for any arbitrary integer $n \geq 0$

The Method of Iteration (3/4)

Definition

A sequence $a_0, a_1, a_2, \ldots$ is called a **geometric sequence** if, and only if, there is a constant r such that

$$a_k = ra_{k-1} \quad \text{for each integer } k \geq 1.$$

It follows that

$$a_n = a_0 r^n \quad \text{for each integer } n \geq 0.$$

Example 5.7.4 – *A Geometric Sequence*

As shown in Example 5.6.7, if a bank pays interest at a rate of 4% per year compounded annually and A_n denotes the amount in the account at the end of year n, then

$$A_k = (1.04)\, A_{k-1}, \text{ for each integer } k \geq 1,$$

assuming no deposits or withdrawals during the year. Suppose the initial amount deposited is \$100,000, and assume that no additional deposits or withdrawals are made.

a. How much will the account be worth at the end of 21 years?

b. In how many years will the account be worth \$1,000,000?

Example 5.7.4(a) – *Solution*

$A_0, A_1, A_2, \cdots$ is a geometric sequence with initial value 100,000 and constant multiplier 1.04. Hence,

$$A_n = \$100{,}000 \cdot (1.04)^n \quad \text{for every integer } n \geq 0.$$

After 21 years, the amount in the account will be

$$A_{21} = \$100{,}000 \cdot (1.04)^{21} \cong \$227{,}876.81.$$

This is the same answer as that obtained in Example 5.6.7 but is computed much more easily (at least if a calculator with a powering key, such $\boxed{\wedge}$ or $\boxed{x^y}$, is used.

Example 5.7.4(b) – *Solution (1/2)*

Let *t* be the number of years needed for the account to grow to $1,000,000. Then

$$\$1{,}000{,}000 = \$100{,}000 \cdot (1.04)^t.$$

Dividing both sides by 100,000 gives

$$10 = (1.04)^t,$$

and taking natural logarithms of both sides results in

$$\ln(10) = \ln(1.04)^t.$$

Example 5.7.4(b) – *Solution (2/2)*

Then

$$\ln(10) \cong t \ln(1.04) \qquad \text{because } \log_b(x^a) = a \log_b(x)$$

and so

$$t = \frac{\ln(10)}{\ln(1.04)} \cong 58.7.$$

Hence the account will grow to $1,000,000 in approximately 58.7 years.

The Method of Iteration (4/4)

The following box indicates some quantities that are approximately equal to certain powers of 10.

$10^7 \cong$ number of seconds in a year

$10^9 \cong$ number of bytes of memory in a personal computer

$10^{11} \cong$ number of neurons in a human brain

$10^{17} \cong$ age of the universe in seconds (according to one theory)

$10^{31} \cong$ number of seconds to process all possible positions of a checkers game if moves are processed at a rate of 1 per billionth of a second

$10^{81} \cong$ number of atoms in the universe

$10^{111} \cong$ number of seconds to process all possible positions of a chess game if moves are processed at a rate of 1 per billionth of a second

Using Formulas to Simplify Solutions Obtained by Iteration

Example 5.7.5 – *An Explicit Formula for the Tower of Hanoi Sequence*

Recall that the Tower of Hanoi sequence $m_1, m_2, m_3, \cdots$ satisfies the recurrence relation

$$m_k = 2m_{k-1} \text{ for each integer } k \geq 2$$

and has the initial condition

$$m_1 = 1.$$

Use iteration to guess an explicit formula for this sequence, and make use of a formula for the Sum of a Geometric Sequence to simplify the answer.

Example 5.7.5 – *Solution (1/3)*

By iteration,

$$m_1 = 1$$

$$m_{2} = 2m_1 + 1 = 2 \cdot 1 + 1 = 2^{1} + 1,$$

$$m_{3} = 2m_2 + 1 = 2(2 + 1) + 1 = 2^{2} + 2 + 1,$$

$$m_{4} = 2m_3 + 1 = 2(2^2 + 2 + 1) + 1 = 2^{3} + 2^2 + 2 + 1,$$

$$m_{5} = 2m_4 + 1 = 2(2^3 + 2^2 + 2 + 1) + 1 = 2^{4} + 2^3 + 2^2 + 2 + 1.$$

Example 5.7.5 – *Solution (2/3)* continued

These calculations show that each term up to m_5 is a sum of successive powers of 2, starting with $2^0 = 1$ and going up to 2^k, where k is 1 less than the subscript of the term. The pattern would seem to continue to higher terms because each term is obtained from the preceding one by multiplying by 2 and adding 1; multiplying by 2 raises the exponent of each component of the sum by 1, and adding 1 adds back the 1 that was lost when the previous 1 was multiplied by 2. For instance, for $n = 6$,

$$\begin{aligned} m_6 = 2m_5 + 1 &= 2(2^4 + 2^3 + 2^2 + 2 + 1) + 1 \\ &= 2^5 + 2^4 + 2^3 + 2^2 + 2 + 1. \end{aligned}$$

Example 5.7.5 – *Solution (3/3)*

continued

Thus it seems that, in general,

$$m_n = 2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2 + 1.$$

By the formula for the sum of a geometric sequence (Theorem 5.2.2),

$$2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2 + 1 = \frac{2^n - 1}{2 - 1} = 2^n - 1.$$

Hence the explicit formula seems to be

$$m_n = 2^n - 1 \quad \text{for every integer } n \geq 1.$$

Checking the Correctness of a Formula by Mathematical Induction

Checking the Correctness of a Formula by Mathematical Induction (1/1)

As you can see from some of the previous examples, the process of solving a recurrence relation by iteration can involve complicated calculations. It is all too easy to make a mistake and come up with the wrong formula. That is why it is important to confirm your calculations by checking the correctness of your formula. The most common way to do this is to use mathematical induction.

Example 5.7.7 – *Using Mathematical Induction to Verify the Correctness of a Solution to a Recurrence Relation*

We have obtained a formula for the Tower of Hanoi sequence. Use mathematical induction to show that this formula is correct.

Example 5.7.7 – *Solution (1/6)*

What does it mean to show the correctness of a formula for a recursively defined sequence? Given a sequence of numbers that satisfies a certain recurrence relation and initial condition, the job is to show that each term of the sequence satisfies the proposed explicit formula. In this case, you need to prove the following statement:

If $m_1, m_2, m_3, \ldots$ is the sequence defined by

$$m_k = 2m_{k-1} + 1 \text{ for each integer } k \geq 2, \text{ and}$$
$$m_1 = 1,$$

then $m_n = 2^n - 1$ for every integer $n \geq 1$.

Example 5.7.7 – *Solution (2/6)*

continued

Proof of Correctness: Let $m_1, m_2, m_3, \cdots$ be the sequence defined by specifying that $m_1 = 1$ and $m_k = 2m_{k-1} + 1$ for each integer $k \geq 2$, and let the property $P(n)$ be the equation

$$m_n = 2^n - 1. \qquad \leftarrow P(n)$$

We will use mathematical induction to prove that for every integer $n \geq 1$, $P(n)$ is true.

Show that P*(1) *is true:

To establish $P(1)$, we must show that

$$m_1 = 2^1 - 1. \qquad \leftarrow P(1)$$

Example 5.7.7 – *Solution (3/6)* continued

Now the left-hand side of $P(1)$ is

$$m_1 = 1 \qquad \text{by definition of } m_1, m_2, m_3, \cdots,$$

and the right-hand side of $P(1)$ is

$$2^1 - 1 = 2 - 1 = 1.$$

Thus the two sides of $P(1)$ equal the same quantity, and hence $P(1)$ is true.

Example 5.7.7 – *Solution (4/6)* continued

Show that for every integer k ≥ 1, if P(k) is true then P(k + 1) is also true:

[Suppose that P(k) is true for a particular but arbitrarily chosen integer k ≥ 1. That is:]

Suppose that k is any integer with $k \geq 1$ such that

$$m_k = 2^k - 1. \qquad \leftarrow P(k) \text{ inductive hypothesis}$$

[We must show that P(k + 1) is true. That is:] We must show that

$$m_{k+1} = 2^{k+1} - 1. \qquad \leftarrow P(k+1)$$

Example 5.7.7 – *Solution (5/6)* continued

But the left-hand side of $P(k + 1)$ is

$$m_{k+1} = 2m_{(k+1)-1} + 1$$ by definition of $m_1, m_2, m_3, \cdots,$

$$= 2m_k + 1$$

$$= 2(2^k - 1) + 1$$ by substitution from the inductive hypothesis

$$= 2^{k+1} - 2 + 1$$ by the distributive law and the fact that $2 \cdot 2^k = 2^{k+1}$

$$= 2^{k+1} - 1$$ by basic algebra

Example 5.7.7 – *Solution (6/6)* continued

which equals the right-hand side of $P(k + 1)$. *[Since the basis and inductive steps have been proved, it follows by mathematical induction that the given formula holds for every integer* $n \geq 1$*.]*

Discovering That an Explicit Formula Is Incorrect

Example 5.7.8 – *Using Verification by Mathematical Induction to Find a Mistake*

Let $c_0, c_1, c_2, \cdots$ be the sequence defined as follows:

$$c_k = 2c_{k-1} + k \quad \text{for each integer } k \geq 1,$$

$$c_0 = 1.$$

Suppose your calculations suggest that $c_0, c_1, c_2, \cdots$ satisfies the following explicit formula:

$$c_n = 2^n + n \qquad \text{for every integer } n \geq 0.$$

Is this formula correct?

Example 5.7.8 – *Solution (1/4)*

Start to prove the statement by mathematical induction and see what develops. The proposed formula satisfies the basis step of the inductive proof since on the one hand, $c_0 = 1$ by definition and on the other hand, $2^0 + 0 = 1 + 0 = 1.$

In the inductive step, you suppose that k is any integer with $k \geq 0$ such that

$$c_k = 2^k + k,$$ This is the inductive hypothesis.

and then you must show that

$$c_{k+1} = 2^{k+1} + (k+1).$$

Example 5.7.8 – *Solution (2/4)* continued

To do this, you start with c_{k+1}, substitute from the recurrence relation, and use the inductive hypothesis:

$$c_{k+1} = 2c_k + (k+1) \quad \text{by the recurrence relation}$$

$$= 2(2^k + k) + (k+1) \quad \text{by substitution from the inductive hypothesis}$$

$$= 2^{(k+1)} + 3k + 1 \quad \text{by basic algebra.}$$

To finish the verification, therefore, you need to show that

$$2^{k+1} + 3k + 1 = 2^{k+1} + (k+1).$$

Example 5.7.8 – *Solution (3/4)* continued

Now this equation is equivalent to

$2k = 0$ by subtracting $2^{k+1} + k + 1$ from both sides

which is equivalent to

$k = 0$ by dividing both sides by 2.

But this is false since k may be *any* nonnegative integer. For instance, when $k = 1$, then $k + 1 = 2$, and

$$c_2 = 2 \cdot 3 + 2 = 8 \quad \text{whereas} \quad 2^2 + 2 = 4 + 2 = 6.$$

Example 5.7.8 – *Solution (4/4)*

continued

So the formula does not give the correct value for c_2. Hence the sequence $c_0, c_1, c_2, \cdots$ does not satisfy the proposed formula.

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.8 Second-Order Linear Homogeneous Recurrence Relations with Constant Coefficients

Copyright © Cengage Learning. All rights reserved.

Second-Order Linear Homogeneous Recurrence Relations with Constant Coefficients

Definition

A second-order linear homogeneous recurrence relation with constant coefficients is a recurrence relation of the form

$$a_k = Aa_{k-1} + Ba_{k-2} \quad \text{for every integer } k \geq \text{some fixed integer,}$$

where A and B are fixed real numbers with $B \neq 0$.

Example 5.8.1 – *Second-Order Linear Homogeneous Recurrence Relations with Constant Coefficients*

State whether each of the following is a second-order linear homogeneous recurrence relation with constant coefficients:

a. $a_k = 3a_{k-1} + 2a_{k-2}$

b. $b_k = b_{k-1} + b_{k-2} + b_{k-3}$

c. $c_k = \frac{1}{2}c_{k-1} - \frac{3}{7}c_{k-2}$

d. $d_k = d_{k-1}^2 + d_{k-1} \cdot d_{k-2}$

e. $e_k = 2e_{k-2}$

f. $f_k = 2f_{k-1} + 1$

g. $g_k = g_{k-1} + g_{k-2}$

h. $h_k = (-1)h_{k-1} + (k-1)h_{k-2}$

Example 5.8.1 – *Solution (1/2)*

a. Yes; $A = 3$ and $B = 2$

b. No; not second-order

c. Yes; $A = \frac{1}{2}$ and $B = -\frac{3}{7}$

d. No; not linear

e. Yes; $A = 0$ and $B = 2$

f. No; not homogeneous

Example 5.8.1 – *Solution (2/2)*

continued

g. Yes; $A = 1$ and $B = 1$

h. No; nonconstant coefficients

The Distinct-Roots Case

The Distinct-Roots Case (1/4)

Lemma 5.8.1

Let A and B be real numbers. A recurrence relation of the form

$$a_k = Aa_{k-1} + Ba_{k-2} \quad \text{for every integer } k \geq 2 \tag{5.8.1}$$

is satisfied by the sequence

$$1, t, t^2, t^3, \ldots, t^n, \ldots,$$

where t is a nonzero real number, if, and only if, t satisfies the equation

$$t^2 - At - B = 0. \tag{5.8.2}$$

The Distinct-Roots Case (2/4)

Definition

Given a second-order linear homogeneous recurrence relation with constant coefficients

$$a_k = Aa_{k-1} + Ba_{k-2} \quad \text{for every integer } k \geq 2, \tag{5.8.1}$$

the **characteristic equation of the relation** is

$$t^2 - At - B = 0. \tag{5.8.2}$$

Example 5.8.2 – *Using the Characteristic Equation to Find Solutions to a Recurrence Relation*

Consider the recurrence relation that specifies that the kth term of a sequence equals the sum of the $(k - 1)$st term plus twice the $(k - 2)$nd term. That is,

$$a_k = a_{k-1} + 2a_{k-2} \text{ for each integer } k \geq 2. \qquad 5.8.3$$

Find all sequences that satisfy relation (5.8.3) and have the form

$$1, t, t^2, t^3, \ldots, t^n, \ldots,$$

where t is nonzero.

Example 5.8.2 – *Solution (1/2)*

By Lemma 5.8.1, relation (5.8.3) is satisfied by a sequence $1, t, t^2, t^3, \ldots, t^n, \ldots$ if, and only if, t satisfies the characteristic equation

$$t^2 - t - 2 = 0.$$

Since

$$t^2 - t - 2 = (t-2)(t+1),$$

the only possible values of t are 2 and −1. It follows that the sequences

$$1, 2, 2^2, 2^3, \ldots, 2^n, \ldots \quad \text{and} \quad 1, -1, (-1)^2, (-1)^3, \ldots, (-1)^n, \ldots.$$

are both solutions for relation (5.8.3) and there are no other solutions of this form.

Example 5.8.2 – *Solution (2/2)* continued

It follows that the sequences

$$1, 2, 2^2, 2^3, \ldots, 2^n, \ldots \quad \text{and} \quad 1, -1, (-1)^2, (-1)^3, \ldots, (-1)^n, \ldots.$$

are both solutions for relation (5.8.3) and there are no other solutions of this form.

Note that these sequences can be rewritten more simply as

$$1, 2, 2^2, 2^3, \ldots, 2^n, \ldots \quad \text{and} \quad 1, -1, 1, -1, \ldots, (-1)^n, \ldots.$$

The Distinct-Roots Case (3/4)

Lemma 5.8.2

If $r_0, r_1, r_2, \ldots$ and $s_0, s_1, s_2, \ldots$ are sequences that satisfy the same second-order linear homogeneous recurrence relation with constant coefficients, and if C and D are *any* numbers, then the sequence $a_0, a_1, a_2, \ldots$ defined by the formula

$$a_n = Cr_n + Ds_n \quad \text{for every integer } n \geq 0$$

also satisfies the same recurrence relation.

Example 5.8.3 – *Finding the Linear Combination That Satisfies the Initial Conditions*

Consider the recurrence relation that specifies that the kth term of a sequence equals the sum of the $(k - 1)$st term plus twice the $(k - 2)$nd term. That is,

$$a_k = a_{k-1} + 2a_{k-2} \text{ for each integer } k \geq 2. \qquad 5.8.3$$

Find a sequence that satisfies the recurrence relation (5.8.3) and that also satisfies the initial conditions

$$a_0 = 1 \text{ and } a_1 = 8.$$

Example 5.8.3 – *Solution (1/3)*

Consider the following sequences from Example 5.8.2.

$$1, 2, 2^2, 2^3, \ldots, 2^n, \ldots \quad \text{and} \quad 1, -1, 1, -1, \ldots, (-1)^n, \ldots$$

Both satisfy relation (5.8.3) although neither satisfies the given initial conditions. However, by Lemma 5.8.2, any sequence a_0, a_1, a_2, … that satisfies the explicit formula

$$a_n = C \cdot 2^n + D(-1)^n, \qquad 5.8.6$$

where C and D are numbers, also satisfies relation (5.8.3).

Example 5.8.3 – *Solution (2/3)*

continued

You can find C and D so that $a_0, a_1, a_2, \ldots$ satisfies the initial conditions specified in this example by substituting $n = 0$ and $n = 1$ into equation (5.8.6) and solving for C and D:

$$a_0 = 1 = C \cdot 2^0 + D(-1)^0$$
$$a_1 = 8 = C \cdot 2^1 + D(-1)^1.$$

When you simplify, you obtain the system

$$1 = C + D$$
$$8 = 2C - D,$$

which can be solved in various ways.

Example 5.8.3 – *Solution (3/3)* continued

For instance, if you add the two equations, you get

$$9 = 3C,$$

and so $$C = 3.$$

Then, by substituting into $1 = C + D$, you get

$$D = -2.$$

It follows that the sequence $a_0, a_1, a_2, \ldots$ given by

$$a_n = 3 \cdot 2^n + (-2)(-1)^n = 3 \cdot 2^n - 2(-1)^n,$$

for each integer $n \geq 0$, satisfies both the recurrence relation and the given initial conditions.

The Distinct-Roots Case (4/4)

Theorem 5.8.3 Distinct-Roots Theorem

Suppose a sequence $a_0, a_1, a_2, \ldots$ satisfies a recurrence relation

$$a_k = Aa_{k-1} + Ba_{k-2} \qquad 5.8.1$$

for some real numbers A and B with $B \neq 0$ and every integer $k \geq 2$. If the characteristic equation

$$t^2 - At - B = 0 \qquad 5.8.2$$

has two distinct roots r and s, then $a_0, a_1, a_2, \ldots$ is given by the explicit formula

$$a_n = Cr^n + Ds^n,$$

where C and D are the numbers whose values are determined by the values a_0 and a_1.

Example 5.8.4 – *A Formula for the Fibonacci Sequence*

The Fibonacci sequence $F_0, F_1, F_2, \ldots$ satisfies the recurrence relation

$$F_k = F_{k-1} + F_{k-2} \text{ for every integer } k \geq 2$$

with initial conditions

$$F_0 = F_1 = 1.$$

Find an explicit formula for this sequence.

Example 5.8.4 – *Solution (1/6)*

The Fibonacci sequence satisfies the first part of the hypothesis of the distinct-roots theorem since the Fibonacci relation is a second-order linear homogeneous recurrence relation with constant coefficients (A = 1 and B = 1).

To check that it satisfies the second part of the hypothesis, examine the characteristic equation

$$t^2 - t - 1 = 0.$$

Example 5.8.4 – *Solution (2/6)*

continued

By the quadratic formula,

$$t = \frac{1 \pm \sqrt{1 - 4(-1)}}{2} = \begin{cases} \dfrac{1 + \sqrt{5}}{2} \\ \dfrac{1 - \sqrt{5}}{2} \end{cases}$$

and so the roots are distinct.

Example 5.8.4 – *Solution (3/6)* continued

Thus it follows from the distinct-roots theorem that the Fibonacci sequence is given by the explicit formula

$$F_n = C\left(\frac{1+\sqrt{5}}{2}\right)^n + D\left(\frac{1-\sqrt{5}}{2}\right)^n \quad \text{for each integer } n \geq 0, \qquad 5.8.7$$

where C and D are the numbers whose values are determined by the fact that $F_0 = F_1 = 1$.

Example 5.8.4 – *Solution (4/6)*

continued

To find *C* and *D*, write

$$F_0 = 1 = C\left(\frac{1+\sqrt{5}}{2}\right)^0 + D\left(\frac{1+\sqrt{5}}{2}\right)^0 = C\cdot 1 + D\cdot 1 = C + D$$

and

$$\begin{aligned} F_1 = 1 &= C\left(\frac{1+\sqrt{5}}{2}\right)^1 + D\left(\frac{1-\sqrt{5}}{2}\right)^1 \\ &= C\left(\frac{1+\sqrt{5}}{2}\right) + D\left(\frac{1-\sqrt{5}}{2}\right). \end{aligned}$$

Example 5.8.4 – *Solution (5/6)* continued

Thus the problem is to find numbers C and D such that

$$C + D = 1$$

and

$$C\left(\frac{1+\sqrt{5}}{2}\right) + D\left(\frac{1-\sqrt{5}}{2}\right) = 1.$$

This may look complicated, but in fact it is just a system of two equations in two unknowns. By solving this two system of equations we get:

$$C = \frac{1+\sqrt{5}}{2\sqrt{5}} \quad \text{and} \quad D = \frac{-(1-\sqrt{5})}{2\sqrt{5}}.$$

Example 5.8.4 – *Solution (6/6)* continued

Substituting these values for C and D into formula (5.8.7) gives

$$F_n = \left(\frac{1+\sqrt{5}}{2\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{-(1-\sqrt{5})}{2\sqrt{5}}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n,$$

or, simplifying,

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n+1} \qquad 5.8.8$$

for each integer $n \geq 0$. Remarkably, even though the formula for F_n involves $\sqrt{5}$, all of the values of the Fibonacci sequence are integers.

The Single-Root Case

The Single-Root Case (1/2)

Lemma 5.8.4

Let A and B be real numbers and suppose the characteristic equation

$$t^2 - At - B = 0$$

has a single root r. Then the sequences $1, r^1, r^2, r^3, \ldots, r^n, \ldots$ and $0, r, 2r^2, 3r^3, \ldots, nr^n, \ldots$ both satisfy the recurrence relation

$$a_k = Aa_{k-1} + Ba_{k-2}$$

for each integer $k \geq 2$.

The Single-Root Case (2/2)

Theorem 5.8.5 Single-Root Theorem

Suppose a sequence $a_0, a_1, a_2, \ldots$ satisfies a recurrence relation

$$a_k = Aa_{k-1} + Ba_{k-2}$$

for some real numbers A and B with $B \neq 0$ and for every integer $k \geq 2$. If the characteristic equation $t^2 - At - B = 0$ has a single (real) root r, then $a_0, a_1, a_2, \ldots$ is given by the explicit formula

$$a_n = Cr^n + Dnr^n,$$

where C and D are the real numbers whose values are determined by the values of a_0 and any other known value of the sequence.

Example 5.8.5 – *Single-Root Case*

Suppose a sequence $b_0, b_1, b_2, \ldots$ satisfies the recurrence relation

$$b_k = 4b_{k-1} - 4b_{k-2} \text{ for every integer } k \geq 2, \qquad 5.8.11$$

with initial conditions

$$b_0 = 1 \text{ and } b_1 = 3.$$

Find an explicit formula for $b_0, b_1, b_2, \ldots$.

Example 5.8.5 – *Solution (1/4)*

This sequence satisfies part of the hypothesis of the single-root theorem because it satisfies a second-order linear homogeneous recurrence relation with constant coefficients ($A = 4$ and $B = -4$).

The single-root condition is also met because the characteristic equation

$$t^2 - 4t + 4 = 0$$

has the unique root $r = 2$ [since $t^2 - 4t + 4 = (t-2)^2$].

Example 5.8.5 – *Solution (2/4)* continued

It follows from the single-root theorem that $b_0, b_1, b_2, \ldots$ is given by the explicit formula

$$b_n = C \cdot 2^n + Dn2^n \qquad \text{for each integer } n \geq 0, \qquad 5.8.12$$

where C and D are the real numbers whose values are determined by the fact that $b_0 = 1$ and $b_1 = 3$. To find C and D, write

$$b_0 = 1 = C \cdot 2^0 + D \cdot 0 \cdot 2^0 = C$$

and

$$b_1 = 3 = C \cdot 2^1 + D \cdot 1 \cdot 2^1 = 2C + 2D.$$

Example 5.8.5 – *Solution (3/4)* continued

Hence the problem is to find numbers C and D such that

$$C = 1$$

and $$2C + 2D = 3.$$

Substitute $C = 1$ into the second equation to obtain

$$2 + 2D = 3,$$

and so

$$D = \frac{1}{2}.$$

Example 5.8.5 – *Solution (4/4)*

continued

Now substitute $C = 1$ and $D = \frac{1}{2}$ into formula (5.8.12) to conclude that

$$b_n = 2^n + \frac{1}{2}n2^n = 2^n\left(1 + \frac{n}{2}\right) \qquad \text{for each integer } n \geq 0.$$

CHAPTER 5

SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Copyright © Cengage Learning. All rights reserved.

5.9 General Recursive Definitions and Structural Induction

Copyright © Cengage Learning. All rights reserved.

General Recursive Definitions and Structural Induction

Sequences of numbers are not the only objects that can be defined recursively.

In this section we discuss recursive definitions for sets and functions. We also introduce *structural induction*, which is a version of mathematical induction that is used to prove properties of recursively defined sets.

Recursively Defined Sets

General Recursive Definitions and Structural Induction (1/4)

To define a set of objects recursively, you identify a few core objects as belonging to the set and give rules showing how to build new set elements from old. More formally, a recursive definition for a set consists of the following three components:

I. **Base:** A statement that certain objects belong to the set.

II. **Recursion:** A collection of rules indicating how to form new set objects from those already known to be in the set.

General Recursive Definitions and Structural Induction (2/4)

III. **Restriction:** A statement that no objects belong to the set other than those coming from the base and the recursion.

Example 5.9.1 – *Recursive Definition of Boolean Expressions (1/2)*

The set of Boolean expressions was introduced in Section 2.4 as "legal" expressions involving letters from the English alphabet such as *p*, *q*, and *r*, and the symbols $\wedge$, $\vee$ $\sim$, and ( ). To make precise which expressions are legal, the set of Boolean expressions over a general alphabet is defined recursively.

I. Base: Each symbol of the alphabet is a Boolean expression.

II. Recursion: If *P* and *Q* are Boolean expressions, then the following are also Boolean expressions:

II(a) $P \wedge Q$ II(b) $P \vee Q$ II(c) $\sim P$ II(d) (P)

Example 5.9.1 – *Recursive Definition of Boolean Expressions (2/2)*

III. Restriction: There are no Boolean expressions over the alphabet other than those obtained from the base and the recursion.

Derive the fact that the following is a Boolean expression over the English alphabet

{*a*, *b*, *c*,…, *x*, *y*, *z*}:

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

Example 5.9.1 – *Solution (1/2)*

(1) By I, p, q, and r are Boolean expressions.

(2) By (1), II(a), and II(c), $p \wedge q$ and $\sim r$ are Boolean expressions.

(3) By (2), II(d), and II(a), $(p \wedge q)$ and $\sim r \wedge p$ are Boolean expressions.

(4) By (3), II(c), and II(d), $\sim(p \wedge q)$ and $(\sim r \wedge p)$ are Boolean expressions.

Example 5.9.1 – *Solution (2/2)*

continued

(5) By (4) and II(b), $\sim(p \wedge q) \vee (\sim r \wedge p)$ is a Boolean expression.

Example 5.9.2 – *Recursive Definition of Parenthesis Structures (1/2)*

Certain configurations of parentheses in algebraic expressions are "legal" *[such as* (())() *and* ()()()*]*, whereas others are not *[such as*)( ))) *and* ()))(((*]*. Here is a recursive definition to generate the set *C* of legal parenthesis structures.

I. Base: ( ) is in *C*.

II. Recursion:

II(a) If *E* is in *C*, so is (*E*).

II(b) If *E* and *F* are in *C*, so is *EF*.

Example 5.9.2 – *Recursive Definition of Parenthesis Structures (2/2)*

III. Restriction: No parenthesis structures are in *P* other than those obtained from the base and the recursion.

Show that (( ))( ) is a parenthesis structure in *C*.

Example 5.9.2 – *Solution*

(1) By I, ( ) is in *C*.

(2) By (1) and II(a), (( )) is in *C*.

(3) By (2), (1), and II(b), (( ))( ) is in *C*.

Recursion is used to give a formal definition for the set of all strings over a finite set.

General Recursive Definitions and Structural Induction (3/4)

Recursive Definition for the Set of All Strings over a Finite Set

Let A be any finite set. Call the elements of A **characters**, and define the **set S of all strings over A** as follows:

I. Base: λ is a string in S, where λ denotes the **null string**, the "string" with no characters.

II. Recursion: New strings are formed according to the following rules:

II(a) If u is any string in S and if c is any character in A, then

$$uc \text{ is a string in } S,$$

where uc is called the **concatenation of u and c**, and is obtained by appending c on the right of u.

II(b) If u is any string in S, then both the concatenation of λ and u, denoted λu, and the concatenation of u and λ, denoted $u\lambda$, are defined to equal u. Symbolically:

$$\lambda u = u\lambda = u.$$

II(c) If u and v are any strings in S, and if c is any character in A, then the concatenation of u and vc is defined to equal the concatenation of uv and c. Symbolically:

$$u(vc) = (uv)c.$$

III. Restriction: Nothing is a string in S other than objects obtained from the base and the recursion.

General Recursive Definitions and Structural Induction (4/4)

The base for the recursive definition of strings only indicates that one character, namely λ, is a string. The next theorem states that each individual character in the underlying set is a string.

Theorem 5.9.1 Characters Are Strings

If A is a finite set and S is the set of all strings over A, then every character in A is a string in S.

Example 5.9.3 – *Proving a Property of Strings*

Suppose A is a finite set, S is the set of all strings over A, and a and b are in A. Because of the way the definition of string is stated, it cannot be deduced immediately that $b(aa)$ is a string in S.

To prove that this is true, first show that ba and $(ba)a$ are strings in S and use the fact that $b(aa) = (ba)a$.

Example 5.9.3 – *Solution*

(1) By Theorem 5.9.1, a and b are strings in S because both a and b are in A.

(2) By (1) and II(a), ba is a string in S because b is in S and $a \in A$.

(3) By (2) and II(a), $(ba)a$ is a string in S because $ba \in S$ and $a \in A$.

(4) By II(c), $b(aa) = (ba)a$ because b and a are strings in S and $a \in A$.

(5) By (3) and (4), $b(aa)$ is a string in S because it equals $(ba)a$, which is a string in S.

Proving Properties about Recursively Defined Sets

Proving Properties about Recursively Defined Sets (1/6)

When a set has been defined recursively, a version of mathematical induction, called **structural induction**, can be used to prove that every object in the set satisfies a given property.

Proving Properties about Recursively Defined Sets (2/6)

Structural Induction for a Recursively Defined Set

Let S be a set that has been defined recursively, and let $P(x)$ be a property that objects in S may or may not satisfy. To prove that every object in S satisfies $P(x)$, perform the following two steps:

Step 1 (basis step): Show that $P(a)$ is true for each object a in the base for S.
Step 2 (inductive step): Show that for each x in S, if $P(x)$ is true and if y is obtained from x by applying a rule from the recursion, then $P(y)$ is true. To perform this step,

> **suppose** that x is an arbitrarily chosen element of S for which $P(x)$ is true.
>
> *[This supposition is the **inductive hypothesis**.]*

Then

> **show** that if y is obtained from x by applying a rule from the recursion for S, then $P(y)$ is true.

Conclusion: Because no objects other than those obtained from the base and recursion are contained in S, steps 1 and 2 prove that $P(x)$ is true for every object x in S.

Example 5.9.5 – *A Property of the Set of Integers*

Let S be the set of all integers defined recursively as follows:

I. Base: 4 is in S.

II. Recursion: Given any integer n in S, $n + 3$ is in S.

III. Restriction: No integers are in S other than those derived from the base and the recursion.

Use structural induction to prove that for every integer n in S, $n \bmod 3 = 1$.

Example 5.9.5 – *Solution (1/4)*

Proof (by structural induction): Given any integer n in S, let property $P(n)$ be the sentence "n *mod* 3 = 1."

Show that P(n) is true for each integer n in the base for S:

The only object in the base for S is 4, and $P(4)$ is true because 4 *mod* 3 = 1 since $4 = 3 \cdot 1 + 1$.

Show that for each integer n in S, if P(n) is true and if m is obtained from n by applying a rule from the recursion for S, then P(m) is true:

Suppose n is any integer in S such that $P(n)$ is true. Then n *mod* 3 = 1.

Example 5.9.5 – *Solution (2/4)* continued

[This is the inductive hypothesis.] The recursion for S consists only of one rule, and when the rule is applied to n, the result is $n + 3$. To complete the inductive step, we must show that $P(n + 3)$ is true.

By inductive hypothesis,

$$n = 3k + 1 \text{ for some integer } k.$$

Example 5.9.5 – *Solution (3/4)* continued

It follows that

$(n + 3)\ mod\ 3 = [(3k + 1) + 3]\ mod\ 3$ by substitution

$= (3k + 4)\ mod\ 3$ by basic algebra

$= [(3k + 1) + 1]\ mod\ 3$ because k + 1 is an integer,

$= 1$

which means that $P(n+3)$ is true *[as was to be shown].*

Example 5.9.5 – *Solution (4/4)* continued

Conclusion:

Because there are no integers in S other than those obtained from the base and the recursion for S, every integer n in S satisfies the equation $n \bmod 3 = 1$.

$(n + 3) \bmod 3 = [(3k + 1) + 3] \bmod 3$

Proving Properties about Recursively Defined Sets (3/6)

A recursive definition can also be given for the length of a string.

Definition Length of a String

Given the set of all strings S over a finite set A, the **length L of a string in S** in defined as follows:

1. $L(\lambda) = 0$.
2. For every string u in S and for every character a in A, the length of ua is one more than the length of u. Symbolically:

$$L(ua) = L(u) + 1 \quad \text{where} \quad u \in S \text{ and } a \in A.$$

Proving Properties about Recursively Defined Sets (4/6)

The following theorem states that the length of a concatenation of two strings is the sum of the lengths of the strings.

Theorem 5.9.2 Additive Property of String Length

If S is the set of all strings over a finite set A, then for all strings u and v in S, $L(uv) = L(u) + L(v)$.

Proving Properties about Recursively Defined Sets (5/6)

The next theorem extends the operation to pairs of strings.

Theorem 5.9.3 The Concatenation of Any Two Strings Is a String

If S is the set of all strings over a finite set A and u and v are any strings in S, then uv is a string in S.

Proving Properties about Recursively Defined Sets (6/6)

The next theorem generalizes the associativity of concatenation to any three strings.

Theorem 5.9.4 Concatenation of Strings Is Associative

If S is the set of all strings over a finite set A and u, v, and w are any strings in S, then $u(vw) = (uv)w$.

Recursive Functions

Recursive Functions (1/1)

A function is said to be **defined recursively** or to be a **recursive function** if its rule of definition refers to itself. Because of this self-reference, it is sometimes difficult to tell whether a given recursive function is well defined.

Recursive functions are of great importance in the theory of computation in computer science.

Example 5.9.7 – *McCarthy's 91 Function*

The following function $M : \mathbf{Z}^+ \to \mathbf{Z}$ was defined by John McCarthy, a pioneer in the theory of computation and in the study of artificial intelligence:

$$M(n) = \begin{cases} n - 10 & \text{if } n > 100 \\ M(M(n+11)) & \text{if } n \leq 100 \end{cases}$$

for all positive integers *n*. Find *M*(99).

Example 5.9.7 – *Solution (1/2)*

By repeated use of the definition of M,

$$M(99) = M(M(110)) \quad \text{since } 99 \le 100$$

$$= M(100) \quad \text{since } 110 > 100$$

$$= M(M(111)) \quad \text{since } 100 \le 100$$

$$= M(101) \quad \text{since } 111 > 100$$

$$= 91 \quad \text{since } 101 > 100.$$

Example 5.9.7 – *Solution (2/2)*

continued

The remarkable thing about this function is that it takes the value 91 for all positive integers less than or equal to 101. (You are asked to show this in exercise 24 at the end of this section.)

For $n > 101$, $M(n)$ is well defined because it equals $n - 10$.

Example 5.9.8 – *The Ackermann Function (1/2)*

In the 1920s the German logician and mathematician Wilhelm Ackermann first defined a version of the function that now bears his name.

This function is important because its values are computable but cannot be evaluated using only for-next loops. The function is defined on the set of all pairs of nonnegative integers as follows:

$A(0, n) = n + 1$ for all nonnegative integers n 5.9.1

$A(m, 0) = A(m - 1, 1)$ for all positive integers m 5.9.2

Example 5.9.8 – *The Ackermann Function (2/2)*

$A(m, n) = A(m - 1, A(m, n - 1))$ for all positive integers m and n 5.9.3

Find $A(1, 2)$.

Example 5.9.8 – *Solution (1/2)*

$A(1, 2) = A(0, A(1, 1))$	by (5.9.3) with $m = 1$ and $n = 2$
$= A(0, A(0, A(1, 0)))$	by (5.9.3) with $m = 1$ and $n = 1$
$= A(0, A(0, A(0, 1)))$	by (5.9.2) with $m = 1$
$= A(0, A(0, 2))$	by (5.9.1) with $n = 1$
$= A(0, 3)$	by (5.9.1) with $n = 2$
$= 4$	by (5.9.1) with $n = 3$.

Example 5.9.8 – *Solution (2/2)*

continued

The special properties of the Ackermann function are a consequence of its phenomenal rate of growth. While the values of $A(0, 0) = 1$, $A(1, 1) = 3$, $A(2, 2) = 7$, and $A(3, 3) = 61$ are not especially impressive,

$$A(4, 4) \cong 2^{2^{2^{65536}}}$$

and the values of $A(n, n)$ continue to increase with extraordinary rapidity thereafter.

Example 5.9.9 – *A Recursive "Function" That Is Not Well Defined*

Consider the following attempt to define a recursive function G from $\mathbf{Z}^+$ to $\mathbf{Z}$. For each integer $n \geq 1$,

$$G(n) = \begin{cases} 1 & \text{if } n \text{ is } 1 \\ 1 + G\left(\frac{n}{2}\right) & \text{if } n \text{ is even} \\ G(3n-1) & \text{if } n \text{ is odd and } n > 1. \end{cases}$$

Is G well defined? Why?

Example 5.9.9 – *Solution (1/2)*

Suppose G is a function. Then by definition of G,

$$G(1) = 1,$$

$$G(2) = 1 + G(1) = 1 + 1 = 2,$$

$$G(3) = G(8) = 1 + G(4) = 1 + (1 + G(2)) = 1 = (1 + 2) = 4,$$

$$G(4) = 1 + G(2) = 1 + 2 = 3.$$

Example 5.9.9 – *Solution (2/2)*

continued

However,

$$G(5) = G(14) = 1 + G(7) = 1 + G(20)$$

$$= 1 + (1 + G(10)) = 1 + (1 + (1 + G(5))) = 3 + G(5).$$

Subtracting $G(5)$ from both sides gives $0 = 3$, which is false. Since the supposition that G is a function leads logically to a false statement, it follows that G is not a function.

CHAPTER 6

SET THEORY

Copyright © Cengage Learning. All rights reserved.

6.1 Set Theory: Definitions and the Element Method of Proof

Copyright © Cengage Learning. All rights reserved.

Set Theory: Definitions and the Element Method of Proof (1/3)

The words *set* and *element* are undefined terms of set theory just as *sentence*, *true*, and *false* are undefined terms of logic.

The founder of set theory, Georg Cantor, suggested imagining a set as a “collection into a whole *M* of definite and separate objects of our intuition or our thought.

These objects are called the elements of *M*.” Cantor used the letter *M* because it is the first letter of the German word for set: *Menge*.

Set Theory: Definitions and the Element Method of Proof (2/3)

Following the spirit of Cantor's notation (though not the letter), let S denote a set and a an element of S.

Then, as indicated in Section 1.2, $a \in S$ means that a is an element of S, $a \notin S$ means that a is not an element of S, {1, 2, 3} refers to the set whose elements are 1, 2, and 3, and {1, 2, 3, … } refers to the set of all positive integers.

Set Theory: Definitions and the Element Method of Proof (3/3)

If S is a set and $P(x)$ is a property that elements of S may or may not satisfy, then a set A may be defined by writing

$$A = \{x \in S \mid P(x)\},$$

the set of all such that

which is read "A is the set of all x in S such that P of x."

Subsets: Proof and Disproof

Subsets: Proof and Disproof (1/3)

We defined what it means for a set *A* to be a subset of the set *B*. Here we rewrite the definition as a formal universal conditional statement:

$$A \subseteq B \quad \Leftrightarrow \quad \forall x, \text{ if } x \in A \text{ then } x \in B.$$

The negation is, therefore, existential:

$$A \not\subseteq B \quad \Leftrightarrow \quad \exists x \text{ such that } x \in A \text{ and } x \notin B.$$

Subsets: Proof and Disproof (2/3)

We know that a *proper subset* of a set is a subset that is not equal to its containing set. That is:

A is a **proper subset** of B $\Leftrightarrow$
(1) $A \subseteq B$, and
(2) there is at least one element in B that is not in A.

Example 6.1.1 – *Testing Whether One Set Is a Subset of Another*

Let $A = \{1\}$ and $B = \{1, \{1\}\}$.

a. Is $A \subseteq B$?

b. If so, is A a proper subset of B?

Example 6.1.1 – *Solution*

a. Because $A = \{1\}$, A has only one element—namely, the symbol 1. This element is also one of the elements in set B. Hence every element in A is in B, and so $A \subseteq B$.

Note A set like {1}, with just one element, is called a **singleton set**.

b. B has two distinct elements, the symbol 1 and the set {1} whose only element is 1. Since $1 \neq \{1\}$, the set {1} is not an element of A, and so there is an element of B that is not an element of A. Hence A is a proper subset of B.

Subsets: Proof and Disproof (3/3)

Element Argument: The Basic Method for Proving That One Set Is a Subset of Another

Let sets X and Y be given. To prove that $X \subseteq Y$,

1. **suppose** that x is a particular but arbitrarily chosen element of X,
2. **show** that x is an element of Y.

Example 6.1.2 – Proving and Disproving Subset Relations

Define sets A and B as follows:

$$A = \{m \in \mathbf{Z} \mid m = 6r + 12 \text{ for some } r \in \mathbf{Z}\}$$

$$B = \{n \in \mathbf{Z} \mid n = 3s \text{ for some } s \in \mathbf{Z}\}.$$

a. Outline a proof that $A \subseteq B$.

b. Prove that $A \subseteq B$.

c. Disprove that $B \subseteq A$.

Example 6.1.2 – *Solution (1/6)*

a. **Proof Outline:**

Starting Point: Suppose x is a particular but arbitrarily chosen element of A.

To Show: Therefore, x is an element of B.

b. **Proof:**

Suppose x is a particular but arbitrarily chosen element of A.

[We must show that $x \in B$. By definition of B, this means we must show that $x = 3 \cdot$ (some integer).]

Example 6.1.2 – *Solution (2/6)* continued

By definition of A, there is an integer, say r, such that $x = 6r + 12$.

[Given that $x = 6r + 12$, *can we express* x *as* $3 \cdot$ (*some integer*)? *That is, does* $6r + 12 = 3 \cdot$ (*some integer*)? *Yes,* $6r + 12 = 3 \cdot (2r + 4)$.*]*

Let $s = 2r + 4$.

[We must check that s *is an integer.]*

Example 6.1.2 – *Solution (3/6)* continued

Then s is an integer because products and sums of integers are integers, and so $3s \in B$ by definition of B.

[Now we must check that $x = 3s$.]

Also $3s = 3(2r + 4) = 6r + 12 = x$,

Thus, by definition of B, x is an element of B,

[as was to be shown].

Example 6.1.2 – *Solution (4/6)* continued

c. To disprove a statement means to show that it is false. And to show that $B \subseteq A$ is false, you must find an element of B that is not an element of A.

By the definitions of A and B, this means that you must find an integer x of the form $3 \cdot$ (some integer) that cannot be written in the form $6 \cdot$ (some integer) + 12.

A little experimentation reveals that various numbers work. For instance, you could let $x = 3$.

Example 6.1.2 – *Solution (5/6)* continued

Then $x \in B$ because $3 = 3 \cdot 1$, but $x \notin A$ because there is no integer r such that $3 = 6r + 12$. For if there were such an integer, then

	$6r + 12 = 3$	by assumption
$\Longrightarrow$	$2r + 4 = 1$	by dividing both sides by 3
$\Longrightarrow$	$2r = -3$	by subtracting 4 from both sides
$\Longrightarrow$	$r = -3/2$	by dividing both sides by 2.

Example 6.1.2 – *Solution (6/6)* continued

But $-3/2$ is not an integer. Thus $3 \in B$ whereas $3 \notin A$, and so $B \not\subseteq A$.

Set Equality

Set Equality (1/1)

Definition

Given sets A and B, A **equals** B, written $\boldsymbol{A = B}$, if, and only if, every element of A is in B and every element of B is in A.

Symbolically:

$$A = B \quad \Leftrightarrow \quad A \subseteq B \text{ and } B \subseteq A.$$

This version of the definition of equality implies the following:

To know that a set A equals a set B, you must know that $A \subseteq B$ and you must also know that $B \subseteq A$.

Example 6.1.3 – *Set Equality*

Define sets *A* and *B* as follows:

$$A = \{m \in \mathbf{Z} \mid m = 2a \text{ for some integer } a\}$$

$$B = \{n \in \mathbf{Z} \mid n = 2b - 2 \text{ for some integer } b\}.$$

Is $A = B$?

Example 6.1.3 – *Solution (1/5)*

Yes. To prove this, both subset relations $A \subseteq B$ and $B \subseteq A$ must be proved.

Part 1, Proof That $A \subseteq B$:

Suppose x is a particular but arbitrarily chosen element of A.

[We must show that $x \in B$. By definition of B, this means we must show that $x = 2 \cdot$ (some integer) $- 2$.]

By definition of A, there is an integer, say a, such that $x = 2a$.

Example 6.1.3 – *Solution (2/5)*

[Given that $x = 2a$, can x also be expressed as $2 \cdot$ (some integer) $- 2$*? In other words, is there an integer—say, b—such that $2a = 2b - 2$? Solve for b to obtain $b = (2a + 2)/2 = a + 1$. Check to see if this works.]*

Let $b = a + 1$.

[First check that b is an integer.]

Then b is an integer because it is a sum of integers.

[Then check that $x = 2b - 2$.]

Also, $2b - 2 = 2(a + 1) - 2 = 2a + 2 - 2 = 2a = x$.

Example 6.1.3 – *Solution (3/5)* continued

Thus, by definition of B, x is an element of B

[as was to be shown].

Part 2*, *Proof That B* $\subseteq$ *A:

Suppose x is a particular but arbitrarily chosen element of B.

[We must show that $x \in A$. By definition of A, this means we must show that $x = 2 \cdot$ (some integer).]

By definition of B, there is an integer b such that $x = 2b - 2$.

Example 6.1.3 – *Solution (4/5)* continued

[Given that $x = 2b - 2$, can x also be expressed as $2 \cdot$ (some integer)? That is, is there an integer— say, a—such that $2b - 2 = 2a$? Solve for a to obtain $a = b - 1$. Check to see if this works.]

Let $a = b - 1$.

[First check that a is an integer.]

We know that a is an integer because it is a difference of integers.

[Then check that $x = 2a$.]

Example 6.1.3 – *Solution (5/5)* continued

By substitution, $2a = 2(b - 1) = 2b - 2 = x$.

Thus, by definition of A, x is an element of A,
[as was to be shown].

Venn Diagrams

Venn Diagrams (1/2)

If sets A and B are represented as regions in the plane, relationships between A and B can be represented by pictures called **Venn diagrams**.

For instance, the relationship $A \subseteq B$ can be pictured in one of two ways, as shown in Figure 6.1.1.

$A \subseteq B$

Figure 6.1.1

Venn Diagrams (2/2)

The relationship $A \not\subset B$ can be represented in three different ways, as shown in Figure 6.1.2.

$A \not\subset B$

Figure 6.1.2

Example 6.1.4 – *Relations among Sets of Numbers (1/2)*

Since **Z**, **Q**, and **R** denote the sets of integers, rational numbers, and real numbers, respectively, then **Z** is a subset of **Q** because every integer is rational (any integer *n* can be written in the form $\frac{n}{1}$), and **Q** is a subset of **R** because every rational number is real (any rational number can be represented as a length on the number line).

Z is a proper subset of **Q** because there are rational numbers that are not integers (for example, $\frac{1}{2}$), and **Q** is a proper subset of **R** because there are real numbers that are not rational (for example, $\sqrt{2}$).

Example 6.1.4 – *Relations among Sets of Numbers (2/2)*

These relationships are shown diagrammatically in Figure 6.1.3.

Figure 6.1.3

Operations on Sets

Operations on Sets (1/4)

Definition

Let A and B be subsets of a universal set U.

1. The **union** of A and B, denoted $\boldsymbol{A \cup B}$, is the set of all elements that are in at least one of A or B.
2. The **intersection** of A and B, denoted $\boldsymbol{A \cap B}$, is the set of all elements that are common to both A and B.
3. The **difference** of B minus A (or **relative complement** of A in B), denoted $\boldsymbol{B - A}$, is the set of all elements that are in B and not A.
4. The **complement** of A, denoted A^c, is the set of all elements in U that are not in A.

Symbolically:

$$A \cup B = \{x \in U | x \in A \text{ or } x \in B\}$$
$$A \cap B = \{x \in U | x \in A \text{ and } x \in B\}$$
$$B - A = \{x \in U | x \in B \text{ and } x \notin A\}$$
$$A^c = \{x \in U | x \notin A\}.$$

Operations on Sets (2/4)

Venn diagram representations for union, intersection, difference, and complement are shown in Figure 6.1.4.

Shaded region represents $A \cup B$.

Shaded region represents $A \cap B$.

Shaded region represents $B - A$.

Shaded region represents A^c.

Figure 6.1.4

Example 6.1.5 – *Unions, Intersections, Differences, and Complements*

Let the universal set be the set $U = \{a, b, c, d, e, f, g\}$, and let $A = \{a, c, e, g\}$ and $B = \{d, e, f, g\}$.

Find $A \cup B$, $A \cap B$, $B - A$, and A^c.

Example 6.1.5 – *Solution*

$A \cup B = \{a, c, d, e, f, g\}$

$A \cap B = \{e, g\}$

$B - A = \{d, f\}$

$A^c = \{b, d, f\}$

Operations on Sets (3/4)

There is a convenient notation for subsets of real numbers that are intervals.

Interval Notation

Given real numbers a and b with $a \leq b$:

$$(a, b) = \{x \in \mathbf{R} \mid a < x < b\} \qquad [a, b] = \{x \in \mathbf{R} \mid a \leq x \leq b\}$$
$$(a, b] = \{x \in \mathbf{R} \mid a < x \leq b\} \qquad [a, b) = \{x \in \mathbf{R} \mid a \leq x < b\}.$$

The symbols ∞ and $-\infty$ are used to indicate intervals that are unbounded either on the right or on the left:

$$(a, \infty) = \{x \in \mathbf{R} \mid x > a\} \quad [a, \infty) = \{x \in \mathbf{R} \mid x \geq a\}$$
$$(-\infty, b) = \{x \in \mathbf{R} \mid x < b\} \quad (-\infty, b] = \{x \in \mathbf{R} \mid x \leq b\}.$$

Example 6.1.6 – *An Example with Intervals*

Let the universal set be **R**, the set of all real numbers, and let

$$A = (-1, 0] = \{x \in \mathbf{R} \mid -1 < x \leq 0\} \quad \text{and} \quad B = [0, 1) = \{x \in \mathbf{R} \mid 0 \leq x < 1\}.$$

These sets are shown on the number lines below.

Find $A \cup B$, $A \cap B$, $B - A$, and A^c.

Example 6.1.6 – *Solution (1/2)*

$$A \cup B = \{x \in \mathbf{R} \mid x \in (-1, 0] \text{ or } x \in [0, 1)\} = \{x \in \mathbf{R} \mid x \in (-1, 1)\} = (-1, 1).$$

$$A \cap B = \{x \in \mathbf{R} \mid x \in (-1, 0] \text{ and } x \in [0, 1)\} = \{0\}.$$

$$B - A = \{x \in \mathbf{R} \mid x \in [0, 1) \text{ and } x \notin (-1, 0]\} = \{x \in \mathbf{R} \mid 0 < x < 1\} = (0, 1)$$

Example 6.1.6 – *Solution (2/2)*

continued

$$A^c = \{x \in \mathbf{R} \mid \text{it is not the case that } x \in (-1, 0]\}$$

$$= \{x \in \mathbf{R} \mid \text{it is not the case that } (-1 < x \text{ and } x \leq 0)\}$$

by definition of the double inequality

$$= \{x \in \mathbf{R} \mid x \leq -1 \text{ or } x > 0\} = (-\infty, -1] \cup (0, \infty)$$

by De Morgan's law

Operations on Sets (4/4)

Definition

Unions and Intersections of an Indexed Collection of Sets

Given sets $A_0, A_1, A_2, \ldots$ that are subsets of a universal set U and given a nonnegative integer n,

$$\bigcup_{i=0}^{n} A_i = \{x \in U \mid x \in A_i \text{ for at least one } i = 0, 1, 2, \ldots, n\}$$

$$\bigcup_{i=0}^{\infty} A_i = \{x \in U \mid x \in A_i \text{ for at least one nonnegative integer } i\}$$

$$\bigcap_{i=0}^{n} A_i = \{x \in U \mid x \in A_i \text{ for every } i = 0, 1, 2, \ldots, n\}$$

$$\bigcap_{i=0}^{\infty} A_i = \{x \in U \mid x \in A_i \text{ for every nonnegative integer } i\}.$$

Example 6.1.7 – *Finding Unions and Intersections of More than Two Sets*

For each positive integer i , let $A_i = \left\{x \in \mathbf{R} \mid -\frac{1}{i} < x < \frac{1}{i}\right\} = \left(-\frac{1}{i}, \frac{1}{i}\right)$.

a. Find $A_1 \cup A_2 \cup A_3$ and $A_1 \cap A_2 \cap A_3$.

b. Find $\bigcup_{i=1}^{\infty} A_i$ and $\bigcap_{i=1}^{\infty} A_i$.

Example 6.1.7 – *Solution (1/3)*

a. $A_1 \cup A_2 \cup A_3 = \{x \in \mathbf{R} \mid x \text{ is in at least one of the intervals } (-1, 1),$

$\text{or } \left(-\frac{1}{2}, \frac{1}{2}\right), \text{ or } \left(-\frac{1}{3}, \frac{1}{3}\right)\}$

$= \{x \in \mathbf{R} \mid -1 < x < 1\}$ because all the elements in $\left(-\frac{1}{2}, \frac{1}{2}\right)$

$= (-1, 1)$ and $\left(-\frac{1}{3}, \frac{1}{3}\right)$ are in $(-1, 1)$

Example 6.1.7 – *Solution (2/3)*

continued

$$A_1 \cap A_2 \cap A_3 = \left\{x \in \mathbf{R} \,\middle|\, x \text{ is in all of the intervals } (-1, 1), \text{ and } \left(-\tfrac{1}{2}, \tfrac{1}{2}\right), \text{ and } \left(-\tfrac{1}{3}, \tfrac{1}{3}\right)\right\}$$

$$= \left\{x \in \mathbf{R} \,\middle|\, -\tfrac{1}{3} < x < \tfrac{1}{3}\right\} \qquad \text{because } \left(-\tfrac{1}{3}, \tfrac{1}{3}\right) \subseteq \left(-\tfrac{1}{2}, \tfrac{1}{2}\right) \subseteq (-1, 1)$$

$$= \left(-\tfrac{1}{3}, \tfrac{1}{3}\right)$$

Example 6.1.7 – *Solution (3/3)* continued

b. $\bigcup_{i=1}^{\infty} A_i = \left\{x \in \mathbf{R} \,\middle|\, x \text{ is in at least one of the intervals } \left(-\frac{1}{i}, \frac{1}{i}\right), \text{ where } i \text{ is a positive integer}\right\}$

$= \{x \in \mathbf{R} \mid -1 < x < 1\}$ because all the elements in every interval $\left(-\frac{1}{i}, \frac{1}{i}\right)$ are in $(-1, 1)$

$= (-1, 1)$

$\bigcap_{i=1}^{\infty} A_i = \left\{x \in \mathbf{R} \,\middle|\, x \text{ is in all of the intervals } \left(-\frac{1}{i}, \frac{1}{i}\right), \text{ where } i \text{ is a positive integer}\right\}$

$= \{0\}$ because the only element in every interval is 0

The Empty Set

The Empty Set (1/1)

There is only one set with no elements. Because it is unique, we can give it a special name. We call it the **empty set** (or **null set**) and denote it by the symbol $\varnothing$.

Thus $\{1, 3\} \cap \{2, 4\} = \varnothing$ and $\{x \in \mathbf{R} | x^2 = -1\} = \varnothing$.

Example 6.1.8 – *A Set with No Elements*

Describe the following sets.

a. $D = \{x \in \mathbf{R} \mid 3 < x < 2\}$.

b. $E = \{x \in \mathbf{Z} \mid 2 < x < 3\}$.

Example 6.1.8 – *Solution*

a. We know that $a < x < b$ means that $a < x$ and $x < b$. So D consists of all real numbers that are both greater than 3 and less than 2.

Since there are no such numbers, D has no elements and thus $D = \varnothing$.

b. E is the set of all integers that are both greater than 2 and less than 3. Since no integers satisfy this condition, E has no elements, and so $E = \varnothing$.

Partitions of Sets

Partitions of Sets (1/4)

In many applications of set theory, sets are divided into nonoverlapping (or *disjoint*) pieces. Such a division is called a *partition*.

Definition

Two sets are called **disjoint** if, and only if, they have no elements in common.

Symbolically:

$$A \text{ and } B \text{ are disjoint} \quad \Leftrightarrow \quad A \cap B = \varnothing.$$

Example 6.1.9 – *Disjoint Sets*

Let $A = \{1, 3, 5\}$ and $B = \{2, 4, 6\}$. Are A and B disjoint?

Example 6.1.9 – *Solution*

Yes. By inspection A and B have no elements in common, or, in other words, $\{1, 3, 5\} \cap \{2, 4, 6\} = \varnothing$.

Partitions of Sets (2/4)

Definition

Sets $A_1, A_2, A_3, \ldots$ are **mutually disjoint** (or **pairwise disjoint** or **nonoverlapping**) if, and only if, no two sets A_i and A_j with distinct subscripts have any elements in common. More precisely, for all integers i and $j = 1, 2, 3, \ldots$

$$A_i \cap A_j = \varnothing \quad \text{whenever } i \neq j.$$

Example 6.1.10 – *Mutually Disjoint Sets*

a. Let $A_1 = \{3, 5\}$, $A_2 = \{1, 4, 6\}$, and $A_3 = \{2\}$. Are A_1, A_2, and A_3 mutually disjoint?

b. Let $B_1 = \{2, 4, 6\}$, $B_2 = \{3, 7\}$, and $B_3 = \{4, 5\}$. Are B_1, B_2, and B_3 mutually disjoint?

Example 6.1.10 – *Solution*

a. Yes. A_1 and A_2 have no elements in common, A_1 and A_3 have no elements in common, and A_2 and A_3 have no elements in common.

b. No. B_1 and B_3 both contain 4.

Partitions of Sets (3/4)

Suppose A, A_1, A_2, A_3, and A_4 are the sets of points represented by the regions shown in Figure 6.1.5.

A Partition of a Set

Figure 6.1.5

Partitions of Sets (4/4)

Then A_1, A_2, A_3, and A_4 are subsets of A, and $A = A_1 \cup A_2 \cup A_3 \cup A_4$. Suppose further that boundaries are assigned to the regions representing A_1, A_2, A_3, and A_4 in such a way that these sets are mutually disjoint. Then A is called a *union of mutually disjoint subsets*, and the collection of sets $\{A_1, A_2, A_3, A_4\}$ is said to be a *partition* of A.

Definition

A finite or infinite collection of nonempty sets $\{A_1, A_2, A_3, \ldots\}$ is a **partition** of a set A if, and only if,

1. A is the union of all the A_i;
2. the sets $A_1, A_2, A_3, \ldots$ are mutually disjoint.

Example 6.1.11 – *Partitions of Sets*

a. Let $A = \{1, 2, 3, 4, 5, 6\}$, $A_1 = \{1, 2\}$, $A_2 = \{3, 4\}$, and $A_3 = \{5, 6\}$. Is $\{A_1, A_2, A_3\}$ a partition of A?

b. Let **Z** be the set of all integers and let

$$T_0 = \{n \in \mathbf{Z} \mid n = 3k, \text{ for some integer } k\},$$
$$T_1 = \{n \in \mathbf{Z} \mid n = 3k + 1, \text{ for some integer } k\}, \text{ and}$$
$$T_2 = \{n \in \mathbf{Z} \mid n = 3k + 2, \text{ for some integer } k\}.$$

Is $\{T_0, T_1, T_2\}$ a partition of **Z**?

Example 6.1.11 – *Solution*

a. Yes. By inspection, $A = A_1 \cup A_2 \cup A_3$ and the sets A_1, A_2, and A_3 are mutually disjoint.

b. Yes. By the quotient-remainder theorem, every integer n can be represented in exactly one of the three forms

$$n = 3k \quad \text{or} \quad n = 3k + 1 \quad \text{or} \quad n = 3k + 2,$$

for some integer k. This implies that no integer can be in any two of the sets T_0, T_1, or T_2. So T_0, T_1, and T_2 are mutually disjoint. The theorem also implies that every integer is in one of the sets T_0, T_1, or T_2. So $\mathbf{Z} = T_0 \cup T_1 \cup T_2$.

Power Sets

Power Sets (1/1)

There are various situations in which it is useful to consider the set of all subsets of a particular set. The **power set axiom** guarantees that this is a set.

Definition

Given a set A, the **power set** of A, denoted $\mathscr{P}(A)$, is the set of all subsets of A.

Example 6.1.12 – *Power Set of a Set*

Find the power set of the set $\{x, y\}$. That is, find $\mathcal{P}(\{x, y\})$.

Example 6.1.12 – *Solution*

$\mathcal{P}(\{x, y\})$ is the set of all subsets of {*x*, *y*}. In the next section we will show that $\varnothing$ is a subset of every set, and so $\varnothing \in \mathcal{P}(\{x, y\})$. Also any set is a subset of itself, so $\{x, y\} \in \mathcal{P}(\{x, y\})$.

The only other subsets of {*x*, *y*} are {*x*} and {*y*}, so

$$\mathcal{P}(\{x, y\}) = \{\varnothing, \{x\}, \{y\}, \{x, y\}\}$$

An Algorithm to Check Whether One Set Is a Subset of Another (Optional)

An Algorithm to Check Whether One Set Is a Subset of Another (Optional) (1/2)

Order the elements of both sets and successively compare each element of the first set with each element of the second set. If some element of the first set is not found to equal any element of the second, then the first set is not a subset of the second.

But if each element of the first set is found to equal an element of the second set, then the first set is a subset of the second. The following algorithm formalizes this reasoning.

An Algorithm to Check Whether One Set Is a Subset of Another (Optional) (2/2)

Algorithm 6.1.1 Testing Whether $A \subseteq B$

[The input sets A and B are represented as one-dimensional arrays $a[1], a[2], \ldots, a[m]$ and $b[1], b[2], \ldots, b[n]$, respectively. Starting with $a[1]$ and for each successive $a[i]$ in A, a check is made to see whether $a[i]$ is in B. To do this, $a[i]$ is compared to successive elements of B. If $a[i]$ is not equal to any element of B, then the output string, called answer, is given the value "$A \not\subseteq B$." If $a[i]$ equals some element of B, the next successive element in A is checked to see whether it is in B. If every successive element of A is found to be in B, then the answer never changes from its initial value "$A \subseteq B$."]

Input: *m [a positive integer], $a[1], a[2], \ldots, a[m]$ [a one-dimensional array representing the set A], n [a positive integer], $b[1], b[2], \ldots, b[n]$ [a one-dimensional array representing the set B]*

Algorithm Body:

```
i := 1, answer := "A ⊆ B"
while (i ≤ m and answer = "A ⊆ B")
    j := 1, found := "no"
        while (j ≤ n and found = "no")
            if a[i] = b[j] then found := "yes"
            j := j + 1
        end while
        [If found has not been given the value "yes" when execution reaches this point,
        then a[i] ∉ B.]
        if found = "no" then answer := "A ⊈ B"
        i := i + 1
end while
```

Output: *answer [a string]*

Example 6.1.13 – *Tracing Algorithm 6.1.1*

Trace the action of Algorithm 6.1.1 on the variables i, j, *found*, and *answer* for $m = 3$, $n = 4$, and sets A and B represented as the arrays $a[1] = u$, $a[2] = v$, $a[3] = w$, $b[1] = w$, $b[2] = x$, $b[3] = y$, and $b[4] = u$.

Example 6.1.13 – *Solution*

i	1					2					3
j	1	2	3	4	5	1	2	3	4	5	
found	no			yes		no					
answer	$A \subseteq B$									$A \not\subseteq B$	

CHAPTER 6

SET THEORY

Copyright © Cengage Learning. All rights reserved.

6.2 Properties of Sets

Copyright © Cengage Learning. All rights reserved.

Properties of Sets (1/3)

It is possible to list many relations involving unions, intersections, complements, and differences of sets. Some of these are true for all sets, whereas others fail to hold in some cases.

In this section we show how to establish basic set properties using *element arguments*, and we discuss a variation used to prove that a set is empty.

We begin by listing some set properties that involve subset relations.

Properties of Sets (2/3)

As you read them, keep in mind that the operations of union, intersection, and difference take precedence over set inclusion. Thus, for example, $A \cap B \subseteq C$ means $(A \cap B) \subseteq C$.

Theorem 6.2.1 Some Subset Relations

1. *Inclusion of Intersection:* For all sets A and B,

$$\text{(a) } A \cap B \subseteq A \quad \text{and} \quad \text{(b) } A \cap B \subseteq B.$$

2. *Inclusion in Union:* For all sets A and B,

$$\text{(a) } A \subseteq A \cup B \quad \text{and} \quad \text{(b) } B \subseteq A \cup B.$$

3. *Transitive Property of Subsets:* For all sets A, B, and C,

$$\text{if } A \subseteq B \text{ and } B \subseteq C, \text{ then } A \subseteq C.$$

Properties of Sets (3/3)

Procedural versions of the definitions of the other set operations are derived similarly and are summarized below.

Procedural Versions of Set Definitions

Let X and Y be subsets of a universal set U and suppose x and y are elements of U.

1. $x \in X \cup Y \quad \Leftrightarrow \quad x \in X$ or $x \in Y$
2. $x \in X \cap Y \quad \Leftrightarrow \quad x \in X$ and $x \in Y$
3. $x \in X - Y \quad \Leftrightarrow \quad x \in X$ and $x \notin Y$
4. $x \in X^c \quad \Leftrightarrow \quad x \notin X$
5. $(x, y) \in X \times Y \quad \Leftrightarrow \quad x \in X$ and $y \in Y$

Proving a Subset Relation

Proving a Subset Relation (1/3)

Consider trying to prove Theorem 6.2.1(a): For all sets A and B, $A \cap B \subseteq A$. First notice that the statement is universal. It makes a claim about *all* sets A and B. So the proof has the following outline:

Starting Point: Suppose A and B are any *[particular but arbitrarily chosen]* sets.

To Show: $A \cap B \subseteq A$

Proving a Subset Relation (2/3)

Now to prove $A \cap B \subseteq A$ you must use the definition of subset. In other words, you must show that

$$\forall x, \text{ if } x \in A \cap B, \text{ then } x \in A.$$

This statement is also universal, and to prove it you use an element argument:

suppose x is any element in $A \cap B$

and

show that x is in A.

Proving a Subset Relation (3/3)

You can fill in the gap between “suppose” and the “show” by using the procedural version of the definition of intersection along with your knowledge of logic and the definition of subset.

Example 6.2.1 – *Fill in the Blanks for Proofs of Subset Relations (1/2)*

Fill in the blanks in the proofs shown below.

a. **Theorem 6.2.1(1)(a):** For all sets A and B, $A \cap B \subseteq A$.

Proof:

Statement	**Explanation**
Suppose A and B are any sets.	starting point
We must show that $A \cap B \subseteq A$.	conclusion to be shown
Let x be any element in $A \cap B$.	start of an element argument
Then x is in A and x is in B.	(*i*)
In particular, x is in A.	(*ii*)
Thus every element in $A \cap B$ is in A.	because x could be any element of $A \cap B$
Therefore, $A \cap B \subseteq A$.	(*iii*)

Example 6.2.1 – *Fill in the Blanks for Proofs of Subset Relations (2/2)*

b. **Theorem 6.2.1(2)(a):** For all sets A and B, $A \subseteq A \cup B$.

Proof:

Statement	**Explanation**
Suppose (*i*).	starting point
We must show that $A \subseteq A \cup B$.	conclusion to be shown
(*ii*)	start of an element argument
Then the following statement is true: "x is in A or x is in B."	For an *or* statement to be true only (*iii*) component needs to be true.
Thus x is in $A \cup B$.	by definition of union
Hence every element in A is in (*iv*).	because x could be any element of A
Therefore, (*v*).	by definition of subset

Example 6.2.1 – *Solution*

a. (*i*) by definition of intersection (*ii*) If an *and* statement is true, then each individual component is true (*iii*) by definition of subset

b. (*i*) A and B are any sets. (*ii*) Let x be any element in A. (*Or: Suppose x is any element in A.*) (*iii*) one (*iv*) $A \cup B$ (*v*) $A \subseteq A \cup B$

Set Identities

Set Identities (1/6)

An **identity** is an equation that is universally true for all elements in some set. For example, the equation $a + b = b + a$ is an identity for real numbers because it is true for all real numbers a and b.

The collection of set properties in the next theorem consists entirely of set identities. That is, they are equations that are true for all sets in some universal set.

Set Identities (2/6)

Theorem 6.2.2 Set Identities

Let all sets referred to below be subsets of a universal set U.

1. *Commutative Laws:* For all sets A and B,

$$\text{(a)}\ A \cup B = B \cup A \quad \text{and} \quad \text{(b)}\ A \cap B = B \cap A.$$

2. *Associative Laws:* For all sets A, B, and C,

$$\text{(a)}\ (A \cup B) \cup C = A \cup (B \cup C) \quad \text{and}$$
$$\text{(b)}\ (A \cap B) \cap C = A \cap (B \cap C).$$

3. *Distributive Laws:* For all sets A, B, and C,

$$\text{(a)}\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \quad \text{and}$$
$$\text{(b)}\ A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

4. *Identity Laws:* For every set A,

$$\text{(a)}\ A \cup \varnothing = A \quad \text{and} \quad \text{(b)}\ A \cap U = A.$$

5. *Complement Laws:* For every set A,

$$\text{(a)}\ A \cup A^c = U \quad \text{and} \quad \text{(b)}\ A \cap A^c = \varnothing.$$

6. *Double Complement Law:* For every set A,

$$(A^c)^c = A.$$

7. *Idempotent Laws:* For every set A,

$$\text{(a)}\ A \cup A = A \quad \text{and} \quad \text{(b)}\ A \cap A = A.$$

Set Identities (3/6)

continued

Theorem 6.2.2 Set Identities

8. *Universal Bound Laws:* For every set A,

$$\text{(a)}\ A \cup U = U \quad \text{and} \quad \text{(b)}\ A \cap \varnothing = \varnothing.$$

9. *De Morgan's Laws:* For all sets A and B,

$$\text{(a)}\ (A \cup B)^c = A^c \cap B^c \quad \text{and} \quad \text{(b)}\ (A \cap B)^c = A^c \cup B^c.$$

10. *Absorption Laws:* For all sets A and B,

$$\text{(a)}\ A \cup (A \cap B) = A \quad \text{and} \quad \text{(b)}\ A \cap (A \cup B) = A.$$

11. *Complements of U and $\varnothing$:*

$$\text{(a)}\ U^c = \varnothing \quad \text{and} \quad \text{(b)}\ \varnothing^c = U.$$

12. *Set Difference Law:* For all sets A and B,

$$A - B = A \cap B^c.$$

Set Identities (4/6)

Basic Method for Proving That Sets Are Equal

Let sets X and Y be given. To prove that $X = Y$:

1. Prove that $X \subseteq Y$.
2. Prove that $Y \subseteq X$.

Example 6.2.2 – *Proof of a Distributive Law*

Consider trying to prove that for all sets A, B, and C,

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

Example 6.2.2 – *Solution (1/6)*

The proof of this fact is somewhat more complicated than the proofs in Example 6.2.1, so we first derive its logical structure, then find the core arguments, and end with a formal proof as a summary.

As in the subset relation examples, the statement to be proved is universal.

Example 6.2.2 – *Solution (2/6)* continued

Thus, by the method of generalizing from the generic particular, the proof has the following outline:

Starting Point: Suppose A, B, and C are arbitrarily chosen sets.

To Show: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

Example 6.2.2 – *Solution (3/6)* continued

Now two sets are equal if, and only if, each is a subset of the other.

Hence, the following two statements must be proved:

$$A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$$

and

$$(A \cup B) \cap (A \cup C) \subseteq A \cup (B \cap C).$$

Example 6.2.2 – *Solution (4/6)* continued

Showing the first subset relation requires showing that

$$\forall x, \text{ if } x \in A \cup (B \cap C) \text{ then } x \in (A \cup B) \cap (A \cup C).$$

Showing the second containment requires showing that

$$\forall x, \text{ if } x \in (A \cup B) \cap (A \cup C) \text{ then } x \in A \cup (B \cap C).$$

Example 6.2.2 – *Solution (5/6)* continued

Note that both of these statements are universal. So to prove the first containment, you

suppose you have any element x in $A \cup (B \cap C)$,

and then you

show that $x \in (A \cup B) \cap (A \cup C)$.

And to prove the second containment, you

suppose you have any element x in $(A \cup B) \cap (A \cup C)$,

and then you

show that $x \in A \cup (B \cap C)$.

Example 6.2.2 – *Solution (6/6)*

continued

Theorem 6.2.2(3)(a) A Distributive Law for Sets

For all sets A, B, and C,

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

Set Identities (5/6)

In the study of artificial intelligence, the types of reasoning used previously to derive the proof of the distributive law are called *forward chaining* and *backward chaining*.

First what is to be shown is viewed as a goal to be reached starting from a certain initial position: the starting point.

Analysis of this goal leads to the realization that if a certain job is accomplished, then the goal will be reached.

Set Identities (6/6)

Call this job subgoal 1: SG_1. (For instance, if the goal is to show that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$, then SG_1 would be to show that each set is a subset of the other.) Analysis of SG_1 shows that when yet another job is completed, SG_1 will be reached.

Call this job subgoal 2: SG_2. Continuing in this way, a chain of argument leading backward from the goal is constructed.

$$\boxed{\text{starting point}} \qquad \rightarrow SG_3 \rightarrow SG_2 \rightarrow SG_1 \rightarrow \boxed{\text{goal}}$$

Example 6.2.3 – *Proof of a De Morgan's Law for Sets*

Prove that for all sets *A* and *B*, $(A \cup B)^c = A^c \cap B^c$.

Example 6.2.3 – *Solution (1/4)*

As in previous examples, the statement to be proved is universal, and so the starting point of the proof and the conclusion to be shown are as follows:

Starting Point: Suppose *A* and *B* are arbitrarily chosen sets.

To Show: $(A \cup B)^c = A^c \cap B^c$

To do this, you must prove both that $(A \cup B)^c \subseteq A^c \cap B^c$ and that $A^c \cap B^c \subseteq (A \cup B)^c$.

Example 6.2.3 – *Solution (2/4)* continued

To prove the first subset relation means to show that

$$\forall x, \text{ if } x \in (A \cup B)^c \text{ then } x \in A^c \cap B^c.$$

And to prove the second subset relation means to show that

$$\forall x, \text{ if } x \in A^c \cap B^c \text{ then } x \in (A \cup B)^c.$$

Since each of these statements is universal and conditional, for the first subset relation, you

$$\textbf{suppose } x \in (A \cup B)^c,$$

and then you

$$\textbf{show} \text{ that } x \in A^c \cap B^c.$$

Example 6.2.3 – *Solution (3/4)* continued

And for the second subset relation, you

$$\textbf{suppose } x \in A^c \cap B^c,$$

and then you

$$\textbf{show} \text{ that } x \in (A \cup B)^c.$$

To fill in the steps of these arguments, you use the procedural versions of the definitions of complement, union, and intersection, and at crucial points you use De Morgan's laws of logic.

Example 6.2.3 – *Solution (4/4)*

continued

Theorem 6.2.2(9)(a) A De Morgan's Law for Sets

For all sets A and B, $(A \cup B)^c = A^c \cap B^c$.

Theorem 6.2.3 Intersection and Union with a Subset

For any sets A and B, if $A \subseteq B$, then

$$\text{(a) } A \cap B = A \quad \text{and} \quad \text{(b) } A \cup B = B.$$

The Empty Set

The Empty Set (1/1)

Theorem 6.2.4 A Set with No Elements Is a Subset of Every Set

If E is a set with no elements and A is any set, then $E \subseteq A$.

Corollary 6.2.5 Uniqueness of the Empty Set

There is only one set with no elements.

Example 6.2.4 – *Proving That a Set Is Empty*

Prove that for any set *A*, $A \cap \varnothing = \varnothing$.

Example 6.2.4 – *Solution (1/2)*

Let *A* be a *[particular, but arbitrarily chosen]* set. To show that $A \cap \varnothing = \varnothing$, it suffices to show that $A \cap \varnothing$ has no elements *[by the element method for proving a set equals the empty set].*

Suppose not. That is, suppose there is at least one element —say *x*—such that $x \in A \cap \varnothing$.

Then, by definition of intersection, $x \in A$ and $x \in \varnothing$.

Example 6.2.4 – *Solution (2/2)*

continued

In particular, $x \in \varnothing$. But this is impossible since $\varnothing$ has no elements. *[This contradiction shows that the supposition that there is an element x in* $A \cap \varnothing$ *is false. So* $A \cap \varnothing$ *has no elements, as was to be shown.]* Thus $A \cap \varnothing = \varnothing$.

Example 6.2.5 – *A Proof for a Conditional Statement*

Prove that for all sets *A*, *B*, and *C*, if $A \subseteq B$ and $B \subseteq C^c$, then $A \cap C = \varnothing$.

Example 6.2.5 – *Solution (1/2)*

Because the statement to be proved is both universal and conditional, you start with the method of direct proof:

Suppose *A*, *B*, and *C* are arbitrarily chosen sets that satisfy the condition: $A \subseteq B$ and $B \subseteq C^c$.

Show that $A \cap C = \varnothing$.

Since the conclusion to be shown is that a certain set is empty, you can use the principle for proving that a set equals the empty set.

Example 6.2.5 – *Solution (2/2)*

continued

Proposition 6.2.6

For all sets A, B, and C, if $A \subseteq B$ and $B \subseteq C^c$, then $A \cap C = \varnothing$.

Example 6.2.6 – *A Generalized Distributive Law*

Prove that for all sets A and $B_1, B_2, B_3, \ldots, B_n$, where n is a positive integer,

$$A \cup \left(\bigcap_{i=1}^{n} B_i \right) = \bigcap_{i=1}^{n} (A \cup B_i).$$

Example 6.2.6 – *Solution (1/6)*

Compare this proof to the one given in Example 6.2.2. Although the notation is more complex, the basic ideas are the same.

Proof: Suppose A and $B_1, B_2, B_3, \ldots, B_n$ are any sets and n is a positive integer.

Example 6.2.6 – *Solution (2/6)* continued

***Part 1*, Proof that** $A \cup \left(\bigcap_{i=1}^{n} B_i\right) \subseteq \bigcap_{i=1}^{n} (A \cup B_i)$**:**

Suppose *x* is any element in $A \cup \left(\bigcap_{i=1}^{n} B_i\right)$. *[We must show that x is in* $\bigcap_{i=1}^{n} (A \cup B_i)$*.]*

By definition of union, $x \in A$ or $x \in \bigcap_{i=1}^{n} B_i$.

Example 6.2.6 – *Solution (3/6)* continued

Case 1, x $\in$ ***A:*** In this case, it is true by definition of union that for every integer $i = 1, 2, \ldots, n$, $x \in A \cup B_i$.

Hence $x \in \bigcap_{i=1}^{n} (A \cup B_i)$.

Case 2, $x \in \bigcap_{i=1}^{n} B_i$***:*** In this case, by definition of the general intersection, we have that for every integer $i = 1, 2, \ldots, n$, $x \in B_i$.

Example 6.2.6 – *Solution (4/6)* continued

Hence, by definition of union, for every integer $i = 1, 2, \ldots, n$, $x \in A \cup B_i$, and so, by definition of general intersection, $x \in \bigcap_{i=1}^{n} (A \cup B_i)$.

Thus, in either case, $x \in \bigcap_{i=1}^{n} (A \cup B_i)$ *[as was to be shown]*.

Part 2, Proof that $\bigcap_{i=1}^{n} (A \cup B_i) \subseteq A \cup \left(\bigcap_{i=1}^{n} B_i \right)$***:***

Suppose x is any element in $\bigcap_{i=1}^{n} (A \cup B_i)$ *[We must show that x is in* $A \cup \bigcap_{i=1}^{n} B_i$*.]*

Example 6.2.6 – *Solution (5/6)* continued

By definition of intersection, $x \in A \cup B_i$ for every integer i = 1, 2, . . . , n. Either $x \in A$ or $x \notin A$.

Case 1, x $\in$ ***A:*** In this case $x \in A \cup \left(\bigcap_{i=1}^{n} B_i \right)$ by definition of union.

Case 2, x $\notin$ ***A:*** By definition of intersection, $x \in A \cup B_i$ for every integer i = 1, 2, . . . , n.

Since $x \notin A$, x must be in each B_i for every integer i = 1, 2, . . . , n.

Example 6.2.6 – *Solution (6/6)*

continued

Hence, by definition of intersection, $x \in \bigcap_{i=1}^{n} B_i$, and so, by definition of union, $x \in A \cup \left(\bigcap_{i=1}^{n} B_i\right)$.

Conclusion: Since both set containments have been proved, it follows by definition of set equality that

$$A \cup \left(\bigcap_{i=1}^{n} B_i\right) = \bigcap_{i=1}^{n} (A \cup B_i).$$

CHAPTER 6

SET THEORY

Copyright © Cengage Learning. All rights reserved.

6.3 Disproofs and Algebraic Proofs

Copyright © Cengage Learning. All rights reserved.

Disproving an Alleged Set Property

Disproving an Alleged Set Property (1/2)

We know that to show a universal statement is false, it suffices to find one example (called a counterexample) for which it is false.

Example 6.3.1 – *Finding a Counterexample for a Set Identity*

Is the following set property true?

For all sets A, B, and C, $(A - B) \cup (B - C) = A - C$.

Example 6.3.1 – *Solution (1/5)*

Observe that the property is true if, and only if, the given equality holds for *all* sets A, B, and C.

So it is false if, and only if, there are sets A, B, and C for which the equality does *not* hold.

One way to solve this problem is to picture sets A, B, and C by drawing a Venn diagram such as that shown in Figure 6.3.1.

Figure 6.3.1

Example 6.3.1 – *Solution (2/5)* continued

If you assume that any of the eight regions of the diagram may be empty of points, then the diagram is quite general.

Find and shade the region corresponding to $(A - B) \cup (B - C)$.

Then shade the region Corresponding to $A - C$. These are shown in Figure 6.3.2.

Figure 6.3.2

Example 6.3.1 – *Solution (3/5)* continued

Comparing the shaded regions seems to indicate that the property is false. For instance, if there is an element in B that is not in either A or C then this element would be in $(A - B) \cup (B - C)$ (because of being in B and not C), but it would not be in $A - C$ since $A - C$ contains nothing outside A.

Similarly, an element that is in both A and C but not B would be in $(A - B) \cup (B - C)$ (because of being in A and not B), but it would not be in $A - C$ (because of being in both A and C).

Example 6.3.1 – *Solution (4/5)* continued

One way is to put one of the integers from 1 through 7 into each of the seven subregions enclosed by the circles representing A, B, and C.

If the proposed set property had involved set complements, it would also be helpful to label the region outside the circles, and so we place the number 8 there.

Example 6.3.1 – *Solution (5/5)* continued

(See Figure 6.3.3.) Then define discrete sets *A*, *B*, and *C* to consist of all the numbers in their respective subregions.

Figure 6.3.3

Disproving an Alleged Set Property (2/2)

Counterexample 1: Let $A = \{1, 2, 4, 5\}$, $B = \{2, 3, 5, 6\}$, and $C = \{4, 5, 6, 7\}$. Then

$$A - B = \{1, 4\}, \quad B - C = \{2, 3\}, \quad \text{and} \quad A - C = \{1, 2\}.$$

Hence

$$(A - B) \cup (B - C) = \{1, 4\} \cup \{2, 3\} = \{1, 2, 3, 4\}, \quad \text{whereas} \quad A - C = \{1, 2\}.$$

Since $\{1, 2, 3, 4\} \neq \{1, 2\}$, we have that $(A - B) \cup (B - C) \neq A - C$.

Problem-Solving Strategy

Problem-Solving Strategy (1/2)

How can you discover whether a given universal statement about sets is true or false? There are two basic approaches.

Either you plunge in and start trying to prove the statement, asking yourself, “What do I need to show?” and “How do I show it?” or you try to find a set of conditions that must be fulfilled to construct a counterexample.

Problem-Solving Strategy (2/2)

With either approach you may have immediate success or you may run into difficulty. The trick is to be ready to switch to the other approach if the one you are working on does not look promising.

For more difficult questions, you may alternate several times between the two approaches before arriving at the correct answer.

The Number of Subsets of a Set

The Number of Subsets of a Set (1/4)

The following theorem states the important fact that if a set has *n* elements then its power set has 2^n elements. The proof uses mathematical induction and is based on the following observations. Suppose *X* is a set and *z* is an element of *X*.

1. The subsets of *X* can be split into two groups: those that do not contain *z* and those that do contain *z*.

2. The subsets of *X* that do not contain *z* are the same as the subsets of $X - \{z\}$.

The Number of Subsets of a Set (2/4)

3. The subsets of X that do not contain z can be matched up one for one with the subsets of X that do contain z by matching each subset A that does not contain z to the subset $A \cup \{z\}$ that contains z. Thus there are as many subsets of X that contain z as there are subsets of X that do not contain z.

The Number of Subsets of a Set (3/4)

For instance, if $X = \{x, y, z\}$, the following table shows the correspondence between subsets of X that do not contain z and subsets of X that contain z.

Subsets of X That Do Not Contain z		Subsets of X That Contain z
$\varnothing$	$\leftrightarrow$	$\varnothing \cup \{z\} = \{z\}$
$\{x\}$	$\leftrightarrow$	$\{x\} \cup \{z\} = \{x, z\}$
$\{y\}$	$\leftrightarrow$	$\{y\} \cup \{z\} = \{y, z\}$
$\{x, y\}$	$\leftrightarrow$	$\{x, y\} \cup \{z\} = \{x, y, z\}$

The Number of Subsets of a Set (4/4)

Theorem 6.3.1

For every integer $n \geq 0$, if a set X has n elements, then $\mathcal{P}(X)$ has 2^n elements.

“Algebraic” Proofs of Set Identities

“Algebraic” Proofs of Set Identities (1/5)

Let U be a universal set and consider the power set of U, $\wp(U)$. The set identities given in Theorem 6.2.2 hold for all elements of $\wp(U)$.

Once a certain number of identities and other properties have been established, new properties can be derived from them algebraically without having to use element method arguments.

“Algebraic” Proofs of Set Identities (2/5)

It turns out that only identities (1–5) of Theorem 6.2.2 are needed to prove any other identity involving only unions, intersections, and complements.

With the addition of identity (12), the set difference law, any set identity involving unions, intersections, complements, and set differences can be established.

“Algebraic” Proofs of Set Identities (3/5)

To use known properties to derive new ones, you need to use the fact that such properties are universal statements. Like the laws of algebra for real numbers, they apply to a wide variety of different situations. Assume that all sets are subsets of $\wp(U)$, then, for instance, one of the distributive laws states that

for all sets A, B, and C, $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

This law can be viewed as a general template into which *any* three particular sets can be placed.

“Algebraic” Proofs of Set Identities (4/5)

Thus, for example, if A_1, A_2, and A_3 represent particular sets, then

$$\underbrace{A_1}_{A} \cap (\underbrace{A_2}_{B} \cup \underbrace{A_3}_{C}) = (\underbrace{A_1}_{A} \cap \underbrace{A_2}_{B}) \cup (\underbrace{A_1}_{A} \cap \underbrace{A_3}_{C}),$$

where A_1 takes the place of A, A_2 takes the place of B, and A_3 takes the place of C.

“Algebraic” Proofs of Set Identities (5/5)

Similarly, if W, X, Y, and Z are any particular sets, then, by the distributive law,

$$\underbrace{(W \cap X)}_{\substack{\updownarrow \\ A}} \cap \underset{\substack{\updownarrow \\ B}}{(Y} \cup \underset{\substack{\updownarrow \\ C}}{Z)} = (\underbrace{(W \cap X)}_{\substack{\updownarrow \\ (A}} \cap \underset{\substack{\updownarrow \\ B)}}{Y}) \cup (\underbrace{(W \cap X)}_{\substack{\updownarrow \\ (A}} \cap \underset{\substack{\updownarrow \\ C)}}{Z}),$$

where $W \cap X$ takes the place of A, Y takes the place of B, and Z takes the place of C.

Example 6.3.2 – *Deriving a Set Difference Property*

Construct an algebraic proof that for all sets A, B, and C, $(A \cup B) - C = (A - C) \cup (B - C)$.

Cite a property from Theorem 6.2.2 for each step of the proof.

Example 6.3.2 – *Solution*

Let A, B, and C be any sets. Then

$$(A \cup B) - C = (A \cup B) \cap C^c$$ by the set difference law

$$= C^c \cap (A \cup B)$$ by the commutative law for $\cap$

$$= (C^c \cap A) \cup (C^c \cap B)$$ by the distributive law

$$= (A \cap C^c) \cup (B \cap C^c)$$ by the commutative law for $\cap$

$$= (A - C) \cup (B - C)$$ by the set difference law.

Example 6.3.3 – *Deriving a Set Identity Using Properties of* $\varnothing$

Construct an algebraic proof that for all sets A and B,
$A - (A \cap B) = A - B$.

Cite a property from Theorem 6.2.2 for every step of the proof.

Example 6.3.3 – *Solution*

Suppose A, B, and C be any sets. Then

$A - (A \cap B) = A \cap (A \cap B)^c$	by the set difference law
$= A \cap (A^c \cup B^c)$	by De Morgan's law
$= (A \cap A^c) \cup (A \cap B^c)$	by the distributive law
$= \varnothing \cup (A \cap B^c)$	by the complement law
$= (A \cap B^c) \cup \varnothing$	by the commutative law for $\cup$
$= A \cap B^c$	by the identity law for $\cup$
$= A - B$	by the set difference law.

Example 6.3.4 – *Deriving a Generalized Associative Law*

Prove that for any sets A_1, A_2, A_3, and A_4,

$((A_1 \cup A_2) \cup A_3) \cup A_4 = A_1 \cup ((A_2 \cup A_3) \cup A_4)$.

Cite a property from Theorem 6.2.2 for every step of the proof.

Example 6.3.4 – *Solution*

Let A_1, A_2, A_3, and A_4 be any sets. Then

$$((A_1 \cup A_2) \cup A_3) \cup A_4$$
$$= (A_1 \cup (A_2 \cup A_3)) \cup A_4$$

by the associative law for $\cup$ with A_1 taking the place of A, A_2 taking the place of B, and A_3 taking the place of C

$$= A_1 \cup ((A_2 \cup A_3) \cup A_4)$$

by the associative law for $\cup$ with A_1 taking the place of A, $A_2 \cup A_3$ taking the place of B, and A_4 taking the place of C.

CHAPTER 6

SET THEORY

Copyright © Cengage Learning. All rights reserved.

6.4 Boolean Algebras, Russell's Paradox, and the Halting Problem

Copyright © Cengage Learning. All rights reserved.

Boolean Algebras, Russell's Paradox, and the Halting Problem (1/9)

Table summarizes the main features of the logical equivalences and the set properties. Notice how similar the entries in the two columns are.

Boolean Algebras, Russell's Paradox, and the Halting Problem (2/9)

Logical Equivalences	**Set Properties**
For all statement variables p, q, and r:	For all sets A, B, and C:
a. $p \vee q \equiv q \vee p$ b. $p \wedge q \equiv q \wedge p$	a. $A \cup B = B \cup A$ b. $A \cap B = B \cap A$
a. $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$ b. $p \vee (q \vee r) \equiv (p \vee q) \vee r$	a. $A \cap (B \cap C) = (A \cap B) \cap C$ b. $A \cup (B \cup C) = (A \cup B) \cup C$
a. $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ b. $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	a. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ b. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
a. $p \vee \mathbf{c} \equiv p$ b. $p \wedge \mathbf{t} \equiv p$	a. $A \cup \varnothing = A$ b. $A \cap U = A$
a. $p \vee \sim p \equiv \mathbf{t}$ b. $p \wedge \sim p \equiv \mathbf{c}$	a. $A \cup A^c = U$ b. $A \cap A^c = \varnothing$

Boolean Algebras, Russell's Paradox, and the Halting Problem (3/9)

Logical Equivalences	**Set Properties**
$\sim(\sim p) \equiv p$	$(A^c)^c = A$
a. $p \vee p \equiv p$ b. $p \wedge p \equiv p$	a. $A \cup A = A$ b. $A \cap A = A$
a. $p \vee \mathbf{t} \equiv \mathbf{t}$ b. $p \wedge \mathbf{c} \equiv \mathbf{c}$	a. $A \cup U = U$ b. $A \cap \varnothing = \varnothing$
a. $\sim(p \vee q) \equiv \sim p \wedge \sim q$ b. $\sim(p \wedge q) \equiv \sim p \vee \sim q$	a. $(A \cup B)^c = A^c \cap B^c$ b. $(A \cap B)^c = A^c \cup B^c$
a. $p \vee (p \wedge q) \equiv p$ b. $p \wedge (p \vee q) \equiv p$	a. $A \cup (A \cap B) = A$ b. $A \cap (A \cup B) = A$
a. $\sim\mathbf{t} \equiv \mathbf{c}$ b. $\sim\mathbf{c} \equiv \mathbf{t}$	a. $U^c = \varnothing$ b. $\varnothing^c = U$

Table 6.4.1

Boolean Algebras, Russell's Paradox, and the Halting Problem (4/9)

Definition and Axioms for a Boolean Algebra

A **Boolean algebra** is a set B together with two operations, generally denoted $+$ and $\cdot$, such that for all a and b in B both $a+b$ and $a \cdot b$ are in B and the following axioms are assumed to hold:

1. *Commutative Laws:* For all a and b in B,

$$\text{(a) } a+b=b+a \quad \text{and} \quad \text{(b) } a \cdot b=b \cdot a.$$

2. *Associative Laws:* For all a, b, and c in B,

$$\text{(a) } (a+b)+c=a+(b+c) \quad \text{and} \quad \text{(b) } (a \cdot b) \cdot c=a \cdot(b \cdot c).$$

3. *Distributive Laws:* For all a, b, and c in B,

$$\text{(a) } a+(b \cdot c)=(a+b) \cdot(a+c) \quad \text{and} \quad \text{(b) } a \cdot(b+c)=(a \cdot b)+(a \cdot c).$$

4. *Identity Laws:* There exist distinct elements 0 and 1 in B such that for each a in B,

$$\text{(a) } a+0=a \quad \text{and} \quad \text{(b) } a \cdot 1=a.$$

5. *Complement Laws:* For each a in B, there exists an element in B, denoted $\bar{a}$ and called the **complement** or **negation** of a, such that

$$\text{(a) } a+\bar{a}=1 \quad \text{and} \quad \text{(b) } a \cdot \bar{a}=0.$$

Boolean Algebras, Russell's Paradox, and the Halting Problem (5/9)

In any Boolean algebra, the complement of each element is unique, the quantities 0 and 1 are unique.

Boolean Algebras, Russell's Paradox, and the Halting Problem (6/9)

Theorem 6.4.1 Properties of a Boolean Algebra

Let B be any Boolean algebra.

1. *Uniqueness of the Complement Laws:* For all a and x in B, if $a + x = 1$ and $a \cdot x = 0$ then $x = \overline{a}$.
2. *Uniqueness of* 0 *and* 1: If there exists x in B such that $a + x = a$ for every a in B, then $x = 0$, and if there exists y in B such that $a \cdot y = a$ for every a in B, then $y = 1$.
3. *Double Complement Law:* For every $a \in B$, $\overline{(\overline{a})} = a$.
4. *Idempotent Laws:* For every $a \in B$,

$$\text{(a) } a + a = a \quad \text{and} \quad \text{(b) } a \cdot a = a.$$

5. *Universal Bound Laws:* For every $a \in B$,

$$\text{(a) } a + 1 = 1 \quad \text{and} \quad \text{(b) } a \cdot 0 = 0.$$

6. *De Morgan's Laws:* For all a and $b \in B$,

$$\text{(a) } \overline{a + b} = \overline{a} \cdot \overline{b} \quad \text{and} \quad \text{(b) } \overline{a \cdot b} = \overline{a} + \overline{b}.$$

Boolean Algebras, Russell's Paradox, and the Halting Problem (7/9)

Theorem 6.4.1 Properties of a Boolean Algebra

7. *Absorption Laws:* For all a and $b \in B$,

$$\text{(a) } (a+b)\cdot a = a \quad \text{and} \quad \text{(b) } (a\cdot b)+a = a.$$

8. *Complements of* 0 *and* 1:

$$\text{(a) } \overline{0} = 1 \quad \text{and} \quad \text{(b) } \overline{1} = 0.$$

Boolean Algebras, Russell's Paradox, and the Halting Problem (8/9)

Each of the paired statements can be obtained from the other by interchanging all the + and · signs and interchanging 1 and 0. Such interchanges transform any Boolean identity into its **dual** identity.

It can be proved that the dual of any Boolean identity is also an identity. This fact is often called the **duality principle** for a Boolean algebra.

Example 6.4.1 – *Proof of the Double Complement Law*

Prove that for all elements *a* in a Boolean algebra *B*, $\overline{(\overline{a})} = a.$

Example 6.4.1 – *Solution*

Start by supposing that *B* is a Boolean algebra and *a* is any element of *B*. The basis for the proof is the uniqueness of the complement law: that each element in *B* has a unique complement, which satisfies certain equations with respect to it.

So if *a* can be shown to satisfy those equations with respect to $\bar{a}$, then *a* must be the complement of $\bar{a}$.

Boolean Algebras, Russell's Paradox, and the Halting Problem (9/9)

Theorem 6.4.1(3) Double Complement Law

For every element a in a Boolean algebra B, $\overline{(\overline{a})} = a$.

Example 6.4.2 – *Proof of an Idempotent Law*

Fill in the blanks in the following proof that for all elements a in a Boolean algebra B, $a + a = a$.

Example 6.4.2 – *Proof*

Suppose *B* is a Boolean algebra and *a* is any element of *B*. Then

$$
\begin{aligned}
a &= a + 0 && \text{(a)} \\
&= a + (a \cdot \overline{a}) && \text{(b)} \\
&= (a + a) \cdot (a + \overline{a}) && \text{(c)} \\
&= (a + a) \cdot 1 && \text{(d)} \\
&= a + a && \text{(e)}.
\end{aligned}
$$

Example 6.4.2 – *Solution*

(a) because 0 is an identity for +

(b) by the complement law for ·

(c) by the distributive law for + over ·

(d) by the complement law for +

(e) because 1 is an identity for ·

Russell’s Paradox

Russell's Paradox (1/3)

Russell's Paradox: Most sets are not elements of themselves. For instance, the set of all integers is not an integer and the set of all horses is not a horse. However, we can imagine the possibility of a set's being an element of itself. For instance, the set of all abstract ideas might be considered an abstract idea. If we are allowed to use any description of a property as the defining property of a set, we can let *S* be the set of all sets that are not elements of themselves:

$$S = \{A \mid A \text{ is a set and } A \notin A\}.$$

Is *S* an element of itself?

Russell's Paradox (2/3)

The answer is both yes and no. For suppose $S \in S$. Then S satisfies the defining property for S, and hence $S \notin S$. This contradicts the supposition that $S \in S$ and shows that $S \notin S$. Next suppose $S \notin S$.

Then S is a set such that $S \notin S$ and so S satisfies the defining property for S, which implies that $S \in S$. This contradicts the supposition that $S \notin S$ and shows that $S \in S$.

Russell's Paradox (3/3)

Thus both $S \in S$ and $S \notin S$, which is impossible because a statement is either true or false but not both. To help explain his discovery to laypeople, Russell devised a puzzle, the barber puzzle, whose solution exhibits the same logic as his paradox.

Example 6.4.3 – *The Barber Puzzle*

In a certain town there is a male barber who shaves all those men, and only those men, who do not shave themselves. *Question:* Does the barber shave himself?

Example 6.4.3 – *Solution*

The answer is both yes and no. If the barber shaves himself, he is a member of the class of men who shave themselves. But no member of this class is shaved by the barber, and so the barber does *not* shave himself.

On the other hand, if the barber does not shave himself, he belongs to the class of men who do not shave themselves. But the barber shaves every man in this class, so the barber *does* shave himself.

The Halting Problem

The Halting Problem (1/2)

Well before the actual construction of an electronic computer, Alan M. Turing (1912–1954) deduced a profound theorem about how such computers would have to work. The argument he used is similar to that in Russell's paradox.

It is also related to those used by Gödel to prove his theorem and by Cantor to prove that it is impossible to write all the real numbers in an infinitely long list, even given an infinitely long period of time.

The Halting Problem (2/2)

Theorem 6.4.2

There is no computer algorithm that will accept any algorithm X and data set D as input and then will output "halts" or "loops forever" to indicate whether or not X terminates in a finite number of steps when X is run with data set D.

CHAPTER 7

PROPERTIES OF FUNCTIONS

Copyright © Cengage Learning. All rights reserved.

7.1 Functions Defined on General Sets

Copyright © Cengage Learning. All rights reserved.

Functions Defined on General Sets (1/4)

As used in ordinary language, the word *function* indicates dependence of one varying quantity on another. If your teacher tells you that your grade in a course will be a function of your performance on the exams, you interpret this to mean that the teacher has some rule for translating exam scores into grades.

To each collection of exam scores there corresponds a certain grade.

Functions Defined on General Sets (2/4)

In this chapter we focus on the more dynamic way functions are used in mathematics. The following is a restatement of the definition of function that includes additional terminology associated with the concept.

Functions Defined on General Sets (3/4)

Definition

A **function f from a set X to a set Y**, denoted $f: X \longrightarrow Y$, is a relation from X, the **domain** of f, to Y, the **co-domain** of f, that satisfies two properties: (1) every element in X is related to some element in Y, and (2) no element in X is related to more than one element in Y.

Thus, given any element x in X, there is a unique element in Y that is related to x by f. If we call this element y, then we say that "f sends x to y" or "f maps x to y" and write $x \xrightarrow{f} y$ or $f: x \longrightarrow y$. The unique element to which f sends x is denoted

$f(x)$ and is called **f of x**, or
the output of f for the input x, or
the value of f at x, or
the image of x under f.

The set of all values of f taken together is called the *range of f* or the *image of X under f*. Symbolically:

$$\textbf{range of } f = \textbf{image of } X \textbf{ under } f = \{y \in Y \mid y = f(x), \text{ for some } x \text{ in } X\}.$$

Given an element y in Y, there may exist elements in X with y as their image. When x is an element such that $f(x) = y$, then x is called **a preimage of y** or **an inverse image of y**. The set of all inverse images of y is called *the inverse image of y*. Symbolically:

$$\textbf{the inverse image of } y = \{x \in X \mid f(x) = y\}.$$

Functions Defined on General Sets (4/4)

The concept of function was developed over a period of centuries. A definition similar to that given above was first formulated for sets of numbers by the German mathematician Lejeune Dirichlet (DEER-ish-lay) in 1837.

Arrow Diagrams

Arrow Diagrams (1/3)

If X and Y are finite sets, you can define a function f from X to Y by drawing an arrow diagram. You make a list of elements in X and a list of elements in Y, and draw an arrow from each element in X to the corresponding element in Y, as shown in Figure 7.1.1.

Figure 7.1.1

Arrow Diagrams (2/3)

This arrow diagram does define a function because:

1. Every element of X has an arrow that points to an element in Y.

2. No element of X has two arrows that point to two different elements of Y.

Example 7.1.1 – *Functions and Nonfunctions*

Which of the arrow diagrams in Figure 7.1.2 define functions from $X = \{a, b, c\}$ to $Y = \{1, 2, 3, 4\}$?

Figure 7.1.2

Example 7.1.1 – *Solution*

Only (c) defines a function.

In (a) the element b in X is not related to any element of Y because there is no arrow that points from b to an element in Y.

And in (b) the element c is not related to a *unique* element of Y because from c there are two arrows that point to two different elements of Y—one toward 2 and the other toward 3.

Example 7.1.2 – *A Function Defined by an Arrow Diagram (1/2)*

Let $X = \{a, b, c\}$ and $Y = \{1, 2, 3, 4\}$. Define a function f from X to Y by the arrow diagram in Figure 7.1.3.

Figure 7.1.3

Example 7.1.2 – *A Function Defined by an Arrow Diagram (2/2)*

a. Write the domain and co-domain of f.

b. Find $f(a)$, $f(b)$, and $f(c)$.

c. What is the range of f?

d. Is c an inverse image of 2? Is b an inverse image of 3?

e. Find the inverse images of 2, 4, and 1.

f. Represent f as a set of ordered pairs.

Example 7.1.2 – *Solution (1/2)*

a. domain of $f = \{a, b, c\}$, co-domain of $f = \{1, 2, 3, 4\}$

b. $f(a) = 2$, $f(b) = 4$, $f(c) = 2$

c. range of $f = \{2, 4\}$

d. yes, no

Example 7.1.2 – *Solution (2/2)* continued

e. inverse image of 2 = {*a*, *c*}

inverse image of 4 = {*b*}

inverse image of 1 = $\varnothing$ *(since no arrows point to* 1)

f. {(*a*, 2), (*b*, 4), (*c*, 2)}

Arrow Diagrams (3/3)

Theorem 7.1.1 A Test for Function Equality

If $F: X \to Y$ and $G: X \to Y$ are functions, then $F = G$ if, and only if, $F(x) = G(x)$ for every $x \in X$.

Example 7.1.3 – *Equality of Functions (1/2)*

a. Let $J_3 = \{0, 1, 2\}$, and define functions f and g from J_3 to J_3 as follows:

For every x in J_3,

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

Does $f = g$?

Example 7.1.3 – *Equality of Functions (2/2)*

b. Let $F : \mathbf{R} \to \mathbf{R}$ and $G : \mathbf{R} \to \mathbf{R}$ be functions. Define new functions $F + G : \mathbf{R} \to \mathbf{R}$ and $G + F : \mathbf{R} \to \mathbf{R}$ as follows: For every $x \in R$,

$$(F + G)(x) = F(x) + G(x) \quad \text{and} \quad (G + F)(x) = G(x) + F(x).$$

Does $F + G = G + F$?

Example 7.1.3 – *Solution (1/2)*

a. Yes, the table of values shows that $f(x) = g(x)$ for every x in J_3.

x	x^2+x+1	$f(x) = (x^2+x+1) \bmod 3$	$(x+2)^2$	$g(x) = (x+2)^2 \bmod 3$
0	1	$1 \bmod 3 = 1$	4	$4 \bmod 3 = 1$
1	3	$3 \bmod 3 = 0$	9	$9 \bmod 3 = 0$
2	7	$7 \bmod 3 = 1$	16	$16 \bmod 3 = 1$

Example 7.1.3 – *Solution (2/2)* continued

b. Again the answer is yes. For every real number x,

$$(F + G)(x) = F(x) + G(x) \quad \text{by definition of } F + G$$

$$= G(x) + F(x) \quad \text{by the commutative law for addition of real numbers}$$

$$= (G + F)(x) \quad \text{by definition of } G + F.$$

Hence $F + G = G + F$.

Examples of Functions

Examples of Functions (1/3)

The next examples illustrate some of the wide variety of different types of functions.

Example 7.1.4 – *The Identity Function on a Set*

Given a set X, define a function I_X from X to X by

$$I_X(x) = x \quad \text{for each } x \text{ in } X.$$

The function I_X is called the **identity function on *X*** because it sends each element of X to the element that is identical to it. Thus the identity function can be pictured as a machine that sends each piece of input directly to the output chute without changing it in any way.

Let X be any set, and suppose that a_{ij}^k and $\phi(z)$ are elements of X. Find $I_X\left(a_{ij}^k\right)$ and $I_X(\phi(z))$.

Example 7.1.4 – *Solution*

Whatever is input to the identity function comes out unchanged, so $I_X = \left(a_{ij}^k\right) = a_{ij}^k$ and $I_X(\phi(z)) = \phi(z)$.

Example 7.1.5 – *Sequences (1/2)*

The formal definition of sequences specifies that an infinite sequence is a function defined on the set of integers that are greater than or equal to a particular integer. For example, the sequence denoted

$$1, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{4}, \frac{1}{5}, \ldots, \frac{(-1)^n}{n+1}, \ldots$$

can be thought of as the function *f* from the nonnegative integers to the real numbers that associates $0 \to 1$, $1 \to -\frac{1}{2}$, $2 \to \frac{1}{3}$, $3 \to -\frac{1}{4}$, $4 \to \frac{1}{5}$, and, in general, $n \to \frac{(-1)^n}{n+1}$.

Example 7.1.5 – *Sequences (2/2)*

In other words, $f: \mathbf{Z}^{nonneg} \to \mathbf{R}$ is the function defined as follows:

Send each integer $n \geq 0$ to $f(n) = \dfrac{(-1)^n}{n+1}$.

In fact, there are many functions that can be used to define a given sequence. For instance, express the sequence above as a function from the set of *positive* integers to the set of real numbers.

Example 7.1.5 – *Solution*

Define $g : \mathbf{Z}^+ \to \mathbf{R}$ by $g(n) = \frac{(-1)^{n+1}}{n}$, for each $n \in \mathbf{Z}^+$.

Then $g(1) = 1, g(2) = -\frac{1}{2}$, $g(3) = \frac{1}{3}$, and, in general,

$$g(n+1) = \frac{(-1)^{n+2}}{n+1} = \frac{(-1)^n}{n+1} = f(n).$$

Example 7.1.6 – *A Function Defined on a Power Set*

We know that $\mathcal{P}(A)$ denotes the set of all subsets of the set *A*. Define a functi $F : \mathcal{P}(\{a, b, c\}) \to \mathbf{Z}^{nonneg}$ as follows: For each $X \in \mathcal{P}(\{a, b, c\})$,

$$F(X) = \text{the number of elements in } X.$$

Draw an arrow diagram for *F*.

Example 7.1.6 – *Solution*

Example 7.1.7 – *Functions Defined on a Cartesian Product (1/2)*

Define functions $M: \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ and $R: \mathbf{R} \times \mathbf{R} \to \mathbf{R} \times \mathbf{R}$ as follows: For each ordered pair (a, b) of integers,

$$M(a, b) = ab \quad \text{and} \quad R(a, b) = (-a, b).$$

Then M is the multiplication function that sends each pair of real numbers to the product of the two, and R is the reflection function that sends each point in the plane that corresponds to a pair of real numbers to the mirror image of the point across the vertical axis.

Example 7.1.7 – *Functions Defined on a Cartesian Product (2/2)*

Note It is customary to omit one set of parentheses for functions defined on Cartesian products. For example, we write $M(a, b)$ rather than $M((a, b))$.

Find the following:

a. $M(-1, -1)$

b. $M\left(\frac{1}{2}, \frac{1}{2}\right)$

c. $M(\sqrt{2}, \sqrt{2})$

d. $R(2, 5)$

e. $R(-2, 5)$

f. $R(3, -4)$

Example 7.1.7 – *Solution*

a. $(-1)(-1) = 1$

b. $(1/2)(1/2) = 1/4$

c. $\sqrt{2} \cdot \sqrt{2} = 2$

d. $(-2, 5)$

e. $(-(-2), 5) = (2, 5)$

f. $(-3, -4)$

Examples of Functions (2/3)

Definition Logarithms and Logarithmic Functions

Let b be a positive real number with $b \neq 1$. For each positive real number x, the **logarithm with base b of x**, written $\log_b x$, is the exponent to which b must be raised to obtain x. Symbolically:

$$\log_b x = y \quad \Leftrightarrow \quad b^y = x.$$

The **logarithmic function with base b** is the function from $\mathbf{R}^+$ to $\mathbf{R}$ that takes each positive real number x to $\log_b x$.

Example 7.1.8 – *The Logarithmic Function with Base b*

Find the following:

a. $\log_3 9$

b. $\log_2\left(\frac{1}{2}\right)$

c. $\log_{10}(1)$

d. $\log_2(2^m)$ (*m* is any real number)

e. $2^{\log_2(m)} (m > 0)$

Example 7.1.8 – *Solution*

a. $\log_3 9 = 2$ because $3^2 = 9$.

b. $\log_2\left(\frac{1}{2}\right) = -1$ because $2^{-1} = \frac{1}{2}$.

c. $\log_{10}(1) = 0$ because $10^0 = 1$.

d. $\log_2(2^m) = m$ because the exponent to which 2 must be raised to obtain 2^m is m.

e. $2^{\log_2(m)} = m$ because $\log_2(m)$ is the exponent to which 2 must be raised to obtain m.

Examples of Functions (3/3)

If S is a nonempty, finite set of characters, then a **string over *S*** can be regarded as a finite sequence of elements of S. The number of characters in a string is called the **length** of the string.

The **null string over *S*** is the “string” with no characters. It is usually denoted λ and is said to have length 0.

Example 7.1.9 – *Encoding and Decoding Functions (1/6)*

Digital messages consist of finite sequences of 0's and 1's. When they are communicated across a transmission channel, they are frequently coded in special ways to reduce the possibility that they will be garbled by interfering noise in the transmission lines.

For example, suppose a message consists of a sequence of 0's and 1's.

Example 7.1.9 – *Encoding and Decoding Functions (2/6)*

A simple way to encode the message is to write each bit three times. Thus the message

$$00101111$$

would be encoded as

$$000000111000111111111111.$$

Example 7.1.9 – *Encoding and Decoding Functions (3/6)*

The receiver of the message decodes it by replacing each section of three identical bits by the one bit to which all three are equal.

Let A be the set of all strings of 0's and 1's, and let T be the set of all strings of 0's and 1's that consist of consecutive triples of identical bits. The encoding and decoding processes described above are actually functions from A to T and from T to A.

Example 7.1.9 – *Encoding and Decoding Functions (4/6)*

The encoding function E is the function from A to T defined as follows: For each string $s \in A$,

$E(s)$ = the string obtained from s by replacing each bit of s by the same bit written three times.

The decoding function D is defined as follows: For each string $t \in T$,

$D(t)$ = the string obtained from t by replacing each consecutive triple of three identical bits of t by a single copy of that bit.

Example 7.1.9 – *Encoding and Decoding Functions (5/6)*

The advantage of this particular coding scheme is that it makes it possible to do a certain amount of error correction when interference in the transmission channels has introduced errors into the stream of bits.

If the receiver of the coded message observes that one of the sections of three consecutive bits that should be identical does not consist of identical bits, then one bit differs from the other two.

Example 7.1.9 – *Encoding and Decoding Functions (6/6)*

In this case, if errors are rare, it is likely that the single bit that is different is the one in error, and this bit is changed to agree with the other two before applying the decoding function.

Example 7.1.10 – *The Hamming Distance Function (1/3)*

The Hamming distance function, named after the computer scientist Richard W. Hamming, is very important in coding theory. It gives a measure of the "difference" between two strings of 0's and 1's that have the same length. Let S_n be the set of all strings of 0's and 1's of length n.

Define a function $H : S_n \times S_n \to \mathbf{Z}^{nonneg}$ as follows: For each pair of strings $(s, t) \in S_n \times S_n$,

$H(s, t)$ = the number of positions in which s and t have different values.

Example 7.1.10 – *The Hamming Distance Function (2/3)*

Thus, letting $n = 5$,

$$H(11111, 00000) = 5$$

because 11111 and 00000 differ in all five positions, whereas

$$H(11000, 00000) = 2$$

because 11000 and 00000 differ only in the first two positions.

Example 7.1.10 – *The Hamming Distance Function (3/3)*

a. Find $H(00101, 01110)$.

b. Find $H(10001, 01111)$.

Example 7.1.10 – *Solution*

a. 3

b. 4

Boolean Functions

Boolean Functions (1/3)

Any input/output table defines a function in the following way: The elements in the input column can be regarded as ordered tuples of 0's and 1's; the set of all such ordered tuples is the domain of the function.

The elements in the output column are all either 0 or 1; thus {0, 1} is taken to be the co-domain of the function. The relation sends each input element to the output element in the same row.

Boolean Functions (2/3)

Thus, for instance, the input/output table of Figure 7.1.4(a) defines the function with the arrow diagram shown in Figure 7.1.4(b).

Input			Output
P	*Q*	*R*	*S*
1	1	1	1
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

(a)

(b)

Two Representations of a Boolean Function

Figure 7.1.4

Boolean Functions (3/3)

More generally, the input/output table corresponding to a circuit with *n* input wires has *n* input columns. Such a table defines a function from the set of all *n*-tuples of 0's and 1's to the set {0, 1}.

Definition

An (***n*-place**) **Boolean function** f is a function whose domain is the set of all ordered n-tuples of 0's and 1's and whose co-domain is the set $\{0, 1\}$. More formally, the domain of a Boolean function can be described as the Cartesian product of n copies of the set $\{0, 1\}$, which is denoted $\{0, 1\}^n$. Thus $f: \{0, 1\}^n \rightarrow \{0, 1\}$.

Example 7.1.11 – *A Boolean Function*

Consider the three-place Boolean function defined from the set of all 3-tuples of 0's and 1's to {0, 1} as follows: For each triple (x_1, x_2, x_3) of 0's and 1's,

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3) \; mod \; 2.$$

Describe f using an input/output table.

Example 7.1.11 – *Solution*

$f(1, 1, 1) = (1 + 1 + 1) \; mod \; 2 = 3 \; mod \; 2 = 1$

$f(1, 1, 0) = (1 + 1 + 0) \; mod \; 2 = 2 \; mod \; 2 = 0$

The rest of the values of *f* can be calculated similarly to obtain the following table.

Input			Output
x_1	x_2	x_3	$(x_1 + x_2 + x_3) \; mod \; 2$
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

Checking Whether a Function Is Well Defined

Checking Whether a Function Is Well Defined (1/2)

It can sometimes happen that what appears to be a function defined by a rule is not really a function at all. To give an example, suppose we wrote, "Define a function $f : \mathbf{R} \to \mathbf{R}$ by specifying that for each real number x,

$$f(x) \text{ is the real number } y \text{ such that } x^2 + y^2 = 1."$$

There are two distinct reasons why this description does not define a function. For almost all values of x, either (1) there is no y that satisfies the given equation or (2) there are two different values of y that satisfy the equation.

Checking Whether a Function Is Well Defined (2/2)

For instance, when $x = 2$, there is no real number y such that $2^2 + y^2 = 1,$ and when $x = 0$, both $y = -1$ and $y = 1$ satisfy the equation $0^2 + y^2 = 1.$

In general, we say that a “function” is **not well defined** if it fails to satisfy at least one of the requirements for being a function.

Example 7.1.12 – *A Function That Is Not Well Defined*

We know that **Q** represents the set of all rational numbers. Suppose you read that a function $f : \mathbf{Q} \to \mathbf{Z}$ is to be defined by the formula

$$f\left(\frac{m}{n}\right) = m \quad \text{for all integers } m \text{ and } n \text{ with } n \neq 0.$$

That is, the integer associated by f to the number $\frac{m}{n}$ is m. Is f well defined? Why?

Example 7.1.12 – *Solution (1/2)*

The function *f* is not well defined. The reason is that fractions have more than one representation as quotients of integers.

For instance, $\frac{1}{2} = \frac{3}{6}$.

Now if *f* were a function, then the definition of a function would imply that $f\left(\frac{1}{2}\right) = \left(\frac{3}{6}\right)$ since $\frac{1}{2} = \frac{3}{6}$.

Example 7.1.12 – *Solution (2/2)* continued

But applying the formula for *f*, you find that

$$f\left(\frac{1}{2}\right) = 1 \quad \text{and} \quad f\left(\frac{3}{6}\right) = 3,$$

and so

$$f\left(\frac{1}{2}\right) \neq \left(\frac{3}{6}\right).$$

This contradiction shows that *f* is not well defined and, therefore, is not a function.

Functions Acting on Sets

Functions Acting on Sets (1/1)

Given a function from a set X to a set Y, you can consider the set of images in Y of all the elements in a subset of X and the set of inverse images in X of all the elements in a subset of Y.

Definition

If $f: X \to Y$ is a function and $A \subseteq X$ and $C \subseteq Y$, then

$$f(A) = \{y \in Y \mid y = f(x) \text{ for some } x \text{ in } A\}$$

and

$$f^{-1}(C) = \{x \in X \mid f(x) \in C\}.$$

$f(A)$ is called the **image of A**, and $f^{-1}(C)$ is called the **inverse image of C**.

Example 7.1.13 – *The Action of a Function on Subsets of a Set*

Let $X = \{1, 2, 3, 4\}$ and $Y = \{a, b, c, d, e\}$, and define $F : X \to Y$ by the following arrow diagram:

Let $A = \{1, 4\}$, $C = \{a, b\}$, and $D = \{c, e\}$. Find $F(A)$, $F(X)$, $F^{-1}(C)$, and $F^{-1}(D)$.

Example 7.1.13 – *Solution*

$F(A) = \{b\}$

$F(X) = \{a, b, d\}$

$F^{-1}(C) = \{1, 2, 4\}$

$F^{-1}(D) = \varnothing$

Example 7.1.14 – *Interaction of a Function with Union*

Let X and Y be sets, let F be a function from X to Y, and let A and B be any subsets of X.

Prove that $F(A \cup B) \subseteq F(A) \cup F(B)$.

Example 7.1.14 – *Solution (1/3)*

The fact that X, Y, F, A, and B were formally introduced prior to the word "Prove" allows you to regard their existence and relationships as part of your background knowledge.

Thus to prove that $F(A \cup B) \subseteq F(A) \cup F(B)$, you only need show that if y is any element in $F(A \cup B)$, then y is an element of $F(A) \cup F(B)$.

Example 7.1.14 – *Solution (2/3)*

Proof:

Suppose $y \in F(A \cup B)$. *[We must show that* $y \in F(A) \cup F(B)$*.]* By definition of function, $y = F(x)$ for some $x \in A \cup B$. By definition of union, $x \in A$ or $x \in B$.

***Case 1*, *x* $\in$ *A*:** In this case, $y = F(x)$ for some x in A. Hence $y \in F(A)$, and so by definition of union, $y \in F(A) \cup F(B)$.

Example 7.1.14 – *Solution (3/3)*

Case 2, $x \in$ ***B*:** In this case, $y = F(x)$ for some x in B. Hence $y \in F(B)$, and so by definition of union, $y \in F(A) \cup F(B)$.

Thus in either case $y \in F(A) \cup F(B)$ *[as was to be shown]*.

CHAPTER 7

PROPERTIES OF FUNCTIONS

Copyright © Cengage Learning. All rights reserved.

7.2 One-to-One, Onto, and Inverse Functions

Copyright © Cengage Learning. All rights reserved.

One-to-One, Onto, and Inverse Functions

In this section we discuss two important properties that functions may satisfy: the property of being *one-to-one* and the property of being *onto*.

Functions that satisfy both properties are called *one-to-one correspondences* or *one-to-one onto functions*.

When a function is a one-to-one correspondence, the elements of its domain and co-domain match up perfectly, and we can define an *inverse function* from the co-domain to the domain that “undoes” the action of the function.

One-to-One Functions

One-to-One Functions (1/4)

In terms of arrow diagrams, this means that two or more arrows that start in the domain can point to the same element in the co-domain.

On the other hand, if no two arrows that start in the domain point to the same element of the co-domain then the function is called *one-to-one* or *injective*.

For a one-to-one function, each element of the co-domain is the image of at most one element of the domain.

One-to-One Functions (2/4)

Definition

Let F be a function from a set X to a set Y. F is **one-to-one** (or **injective**) if, and only if, for all elements x_1 and x_2 in X,

$$\text{if } F(x_1) = F(x_2), \text{ then } x_1 = x_2,$$

or, equivalently,

$$\text{if } x_1 \neq x_2, \text{ then } F(x_1) \neq F(x_2).$$

Symbolically:

$$F: X \longrightarrow Y \text{ is one-to-one} \Leftrightarrow \forall x_1, x_2 \in X, \text{ if } F(x_1) = F(x_2) \text{ then } x_1 = x_2.$$

To obtain a precise statement of what it means for a function *not* to be one-to-one, take the negation of one of the equivalent versions of the definition above.

One-to-One Functions (3/4)

Thus:

A function $F: X \longrightarrow Y$ is *not* one-to-one $\Leftrightarrow$ $\exists$ elements x_1 and x_2 in X with $F(x_1) = F(x_2)$ and $x_1 \neq x_2$.

In other words, if elements x_1 and x_2 can be found that have the same function value but are not equal, then F is not one-to-one.

In terms of arrow diagrams, a one-to-one function can be thought of as a function that separates points. That is, it takes distinct points of the domain to distinct points of the codomain.

One-to-One Functions (4/4)

A function that is not one-to-one fails to separate points. In other words, at least two points of the domain are taken to the same point of the co-domain. This distinction is illustrated in Figures 7.2.1(a) and 7.2.1(b).

A One-to-One Function Separates Points

Figures 7.2.1(a)

A Function That Is Not One-to-One Collapses Points Together

Figures 7.2.1(b)

Example 7.2.1 – *Identifying One-to-One Functions Defined on Finite Sets (1/2)*

a. Do either of the arrow diagrams in Figure 7.2.2 define one-to-one functions?

Figures 7.2.2

Example 7.2.1 – *Identifying One-to-One Functions Defined on Finite Sets (2/2)*

b. Let $X = \{1, 2, 3\}$ and $Y = \{a, b, c, d\}$. Define $H: X \to Y$ as follows: $H(1) = c$, $H(2) = a$, and $H(3) = d$.

Define $K: X \to Y$ as follows: $K(1) = d$, $K(2) = b$, and $K(3) = d$. Is either H or K one-to-one?

Example 7.2.1 – *Solution (1/2)*

a. F is one-to-one but G is not. F is one-to-one because no two different elements of X are sent by F to the same element of Y.

G is not one-to-one because the elements a and c are both sent by G to the same element of Y: $G(a) = G(c) = w$ but $a \neq c$.

Example 7.2.1 – *Solution (2/2)* continued

b. H is one-to-one but K is not. H is one-to-one because each of the three elements of the domain of H is sent by H to a different element of the co-domain: $H(1) \neq H(2)$, $H(1) \neq H(3)$, and $H(2) \neq H(3)$.

K, however, is not one-to-one because $K(1) = K(3) = d$ but $1 \neq 3$.

One-to-One Functions on Infinite Sets

One-to-One Functions on Infinite Sets (1/2)

Now suppose f is a function defined on an infinite set X. By definition, f is one-to-one if, and only if, the following universal statement is true:

$$\forall x_1, x_2 \in X, \text{ if } f(x_1) = f(x_2) \text{ then } x_1 = x_2.$$

Thus, to prove f is one-to-one, you will generally use the method of direct proof:

suppose x_1 and x_2 are elements of X such that $f(x_1) = f(x_2)$

and **show** that $x_1 = x_2$.

One-to-One Functions on Infinite Sets (2/2)

To show that f is *not* one-to-one, you will ordinarily

find elements x_1 and x_2 in X so that $f(x_1) = f(x_2)$ but $x_1 \neq x_2$.

Example 7.2.2 – *Proving or Disproving That Functions Are One-to-One*

Define $f : \mathbf{R} \to \mathbf{R}$ and $g : \mathbf{Z} \to \mathbf{Z}$ by the rules

$$f(x) = 4x - 1 \text{ for all } x \in \mathbf{R}$$

and

$$g(n) = n^2 \quad \text{for all } n \in \mathbf{Z}.$$

a. Is f one-to-one? Prove or give a counterexample.

b. Is g one-to-one? Prove or give a counterexample.

Example 7.2.2 – *Solution (1/8)*

It is usually best to start by taking a positive approach to answering questions like these. Try to prove the given functions are one-to-one and see whether you run into difficulty.

If you finish without running into any problems, then you have a proof. If you do encounter a problem, then analyzing the problem may lead you to discover a counterexample.

Example 7.2.2 – *Solution (2/8)*

continued

a. The function $f : \mathbf{R} \to \mathbf{R}$ is defined by the rule

$$f(x) = 4x - 1 \quad \text{for each real number } x.$$

To prove that f is one-to-one, you need to prove that

$\forall$ real numbers x_1 and x_2, if $f(x_1) = f(x_2)$ then $x_1 = x_2$.

Example 7.2.2 – *Solution (3/8)* continued

Substituting the definition of f into the outline of a direct proof, you

suppose x_1 and x_2 are any real numbers such that $4x_1 - 1 = 4x_2 - 1$,

and **show** that $x_1 = x_2$.

Answer to (a):

If the function $f: \mathbf{R} \to \mathbf{R}$ is defined by the rule $f(x) = 4x - 1$, for each real number x, then f is one-to-one.

Example 7.2.2 – *Solution (4/8)* continued

Proof: Suppose x_1 and x_2 are real numbers such that $f(x_1) = f(x_2)$. *[We must show that $x_1 = x_2$.]* By definition of f,

$$4x_1 - 1 = 4x_2 - 1.$$

Adding 1 to both sides gives

$$4x_1 = 4x_2,$$

and dividing both sides by 4 gives

$$x_1 = x_2,$$

[as was to be shown].

Example 7.2.2 – *Solution (5/8)*

continued

b. The function g: **Z** → **Z** is defined by the rule

$$\boxed{g(n) = n^2 \quad \text{for each integer } n.}$$

As above, you start as though you were going to prove that g is one-to-one. Substituting the definition of g into the outline of a direct proof, you

suppose n_1 and n_2 are integers such that $n_1^2 = n_2^2$,

and **try to show** that $n_1 = n_2$.

Example 7.2.2 – *Solution (6/8)* continued

Can you reach what is to be shown from the supposition? No! It is quite possible for two numbers to have the same squares and yet be different. For example, $2^2 = (-2)^2$ but $2 \neq -2$.

Thus, in trying to prove that *g* is one-to-one, you run into difficulty. But analyzing this difficulty leads to the discovery of a counterexample, which shows that *g* is not one-to-one.

Example 7.2.2 – *Solution (7/8)* continued

This discussion is summarized as follows:

Answer to (b):

If the function $g: \mathbf{Z} \to \mathbf{Z}$ is defined by the rule $g(n) = n^2$, for all $n \in \mathbf{Z}$, then g is not one-to-one.

Counterexample:

Let $n_1 = 2$ and $n_2 = -2$. Then by definition of g,

$$g(n_1) = g(2) = 2^2 = 4 \text{ and also}$$
$$g(n_2) = g(-2) = (-2)^2 = 4.$$

Example 7.2.2 – *Solution (8/8)* continued

Hence $g(n_1) = g(n_2)$ but $n_1 \neq n_2$,

and so g is not one-to-one.

Application: Hash Functions

Application: Hash Functions (1/9)

Definition: Hash Function

A **hash function** is a function defined from a larger, possibly infinite, set of data to a smaller fixed-size set of integers.

To make it efficient for the university to store the records, a hash function is needed that (1) is one-to-one and (2) has a co-domain that is very much smaller than one billion. Most hash functions are modifications of *mod* functions and are defined using prime numbers to increase the chance that their values will be scattered rather than clustered together.

Application: Hash Functions (2/9)

In addition, making their co-domains 50% to 100% larger than their domains makes it more likely that they will be one-to-one.

Nonetheless, two input values may **collide**, that is, have the same output value, and various methods are used to avoid such a **collision**.

Example 7.2.3 – *Computing Values of a Hash Function (1/5)*

Instead of 10,000 students, suppose there are only 6. Define a function H, from the set of student ID numbers to the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10} as follows:

$$H(n) = n \bmod 11 \text{ for each ID number } n.$$

To compute values of H either use a calculator or a computer with a built-in *mod* function or use the formula $n \bmod 11 = n - (n \operatorname{div} 11)$.

Example 7.2.3 – *Computing Values of a Hash Function (2/5)*

In other words, divide n by 11, multiply the integer part of the result by 11, and subtract that number from n.

As an example, since $328343419/11 = 29849401.73$.

$$H(328343419) = 328343419 - (11 \cdot 29849401) = 8$$

Example 7.2.3 – *Computing Values of a Hash Function (3/5)*

To store the link to the record for the student with ID number *n*, start by computing *H*(*n*).

For instance, if the ID numbers are 328343419, 356633102, 223799061, and 513408716, the corresponding *H*-values are as shown in Table 7.2.1.

	0
356633102	1
223799061	2
	3
	4
	5
	6
513408716	7
328343419	8
	9
	10

Table 7.2.1

Example 7.2.3 – *Computing Values of a Hash Function (4/5)*

As noted, *H* may not be one-to-one: two different ID numbers could have the same *H*-value.

But because it is important for each value in the table to link to a single student record, a **collision resolution** method is needed.

Example 7.2.3 – *Computing Values of a Hash Function (5/5)*

One of the simplest, called a **linear probe**, works as follows: If $H(n)$ is already occupied when a new student ID number is input, start from $H(n)$ in the table and search downward to put a link for the student's record in the first empty position that occurs; if no empty positions remain going down, go up to the beginning of the table and search from there.

Because 11 is greater than 6, empty positions are guaranteed. Suppose the ID number for another student is 607275830. Find the position in the table for this number.

Example 7.2.3 – *Solution*

When you compute $H(607275830)$ you find that it equals 7, which is already occupied by the link to the record for ID number 513408716. Searching downward from position 7, you find that position 8 is also occupied but position 9 is free.

$$607275830 \rightarrow \underset{\text{occupied}}{\underset{\uparrow}{7}} \rightarrow \underset{\text{occupied}}{\underset{\uparrow}{8}} \rightarrow \underset{\text{free}}{\underset{\uparrow}{9}}$$

Therefore, you put the link for the record with ID number 607275830 in position 9.

Application: Hash Functions (3/9)

A special class of hash functions, known as *cryptographic hash functions*, is used to secure digital data. A **cryptographic hash function** is designed to satisfy the following conditions:

1. It is a function from bit strings to bit strings of a fixed length.

2. It is close to being one-to-one: the probability of collisions is very small.

Application: Hash Functions (4/9)

3. It is close to being a *one-way function*: given any bit string in its range, finding the inverse image of the string is computationally very difficult.

4. Its values can be quickly computed.

5. A very slight change in an input string results in an extensive change in the output string.

One use of cryptographic hash functions is to provide password security.

Application: Hash Functions (5/9)

Passwords in a company's user account file are almost never stored as "clear" text. A basic protection is to apply a cryptographic hash function to the passwords, or to a combination consisting of the passwords plus extra content provided by the company, and to store only the values of the hash function, called the *hashes*.

To log in, a user keys in a password, which is immediately *hashed* (meaning that its value is input to the hash function), and the result is compared to the hash stored in the account file.

Application: Hash Functions (6/9)

In order to complete the login process, the two hashes have to agree.

A somewhat similar kind of hashing is used for checking the integrity of files. When a file is intended to be copied, a cryptographic hash function is applied to it.

The accuracy of a copy is checked by applying the same hash function. If the two hashes agree, the copy is accepted.

Application: Hash Functions (7/9)

Similarly, when *A* needs to send a file through a possibly insecure network to *B*, *A* can first apply a cryptographic hash function to the file.

Then *A* sends the hash separately to *B* through a secure network, and when *B* receives the file, *B* applies the same hash function to it that *A* used.

Application: Hash Functions (8/9)

B compares the result with the hash received from *A*, and if the two agree, *B* can have confidence that the file was unchanged during transmission.

Cryptographic hash functions are also used in *blockchain* technology. A blockchain is a public register on the Internet made up of linked blocks consisting of records or programs.

To make it impossible to change the data in any part of a block, each includes a time stamp plus a hash computed from all the previous parts of the blockchain.

Application: Hash Functions (9/9)

To keep the system operating at a reasonable pace and to validate additions, a time-consuming *proof of work* is required to add a block to the blockchain.

A commonly used proof of work requires a programmer to use repeated random trials, to discover the input that needs to be added to the data in the blockchain so that when the two are hashed together the output will contain a specified number of initial zeroes.

Onto Functions

Onto Functions (1/4)

There may be an element of the co-domain of a function that is not the image of any element in the domain. On the other hand, it is possible for *every* element in a function's co-domain to be the image of some element in its domain.

Such a function is called *onto* or *surjective*. When a function is onto, its range is equal to its co-domain.

Definition

Let F be a function from a set X to a set Y. F is **onto** (or **surjective**) if, and only if, given any element y in Y, it is possible to find an element x in X with the property that $y = F(x)$.

Symbolically:

$$F: X \longrightarrow Y \text{ is onto} \quad \Leftrightarrow \quad \forall y \in Y, \exists x \in X \text{ such that } F(x) = y.$$

Onto Functions (2/4)

To obtain a precise statement of what it means for a function *not* to be onto, take the negation of the definition of onto:

$$F: X \longrightarrow Y \text{ is } \textit{not} \text{ onto} \quad \Leftrightarrow \quad \exists y \text{ in } Y \text{ such that } \forall x \in X,\ F(x) \neq y.$$

That is, there is some element in *Y* that is *not* the image of *any* element in *X*.

Onto Functions (3/4)

In terms of arrow diagrams, a function is onto if each element of the co-domain has an arrow pointing to it from some element of the domain.

A function is not onto if at least one element in its co-domain does not have an arrow pointing to it.

Onto Functions (4/4)

This is illustrated in Figures 7.2.3(a) and 7.2.3(b).

A Function That Is Onto

Figures 7.2.3(a)

A Function That Is Not Onto

Figures 7.2.3(b)

Example 7.2.4 – *Identifying Onto Functions Defined on Finite Sets (1/2)*

a. Do either of the arrow diagrams in Figure 7.2.4 define onto functions?

Figures 7.2.4

Example 7.2.4 – *Identifying Onto Functions Defined on Finite Sets (2/2)*

b. Let $X = \{1, 2, 3, 4\}$ and $Y = \{a, b, c\}$. Define $H : X \to Y$ as follows: $H(1) = c$, $H(2) = a$, $H(3) = c$, $H(4) = b$.

Define $K : X \to Y$ as follows: $K(1) = c$, $K(2) = b$, $K(3) = b$, and $K(4) = c$. Is either H or K onto?

Example 7.2.4 – *Solution*

a. F is not onto because $b \neq F(x)$ for any x in X. G is onto because each element of Y equals $G(x)$ for some x in X : $a = G(3)$, $b = G(1)$, $c = G(2) = G(4)$, and $d = G(5)$.

b. H is onto but K is not. H is onto because each of the three elements of the co-domain of H is the image of some element of the domain of H : $a = H(2)$, $b = H(4)$, and $c = H(1) = H(3)$. K, however, is not onto because $a \neq K(x)$ for any x in $\{1, 2, 3, 4\}$.

Onto Functions on Infinite Sets

Onto Functions on Infinite Sets (1/2)

Now suppose F is a function from a set X to a set Y, and suppose Y is infinite. By definition, F is onto if, and only if, the following universal statement is true:

$$\forall y \in Y, \exists x \in X \text{ such that } F(x) = y.$$

Thus to prove F is onto, you will ordinarily use the method of generalizing from the generic particular:

suppose that y is any element of Y

and **show** that there is an element x in X with $F(x) = y$.

Onto Functions on Infinite Sets (2/2)

To prove F is *not* onto, you will usually

find an element y of Y such that $y \neq F(x)$ for *any* x in X.

Example 7.2.5 – *Proving or Disproving That Functions Are Onto*

Define $f : \mathbf{R} \to \mathbf{R}$ and $h : \mathbf{Z} \to \mathbf{Z}$ by the rules

$$f(x) = 4x - 1 \text{ for each } x \in \mathbf{R}$$

and

$$h(n) = 4n - 1 \text{ for each } n \in \mathbf{Z}.$$

a. Is f onto? Prove or give a counterexample.

b. Is h onto? Prove or give a counterexample.

Example 7.2.5 – *Solution (1/10)*

a. The best approach is to start trying to prove that *f* is onto and be alert for difficulties that might indicate that it is not.

Now $f : \mathbf{R} \to \mathbf{R}$ is the function defined by the rule

$$f(x) = 4x - 1 \quad \text{for each real number } x.$$

Example 7.2.5 – *Solution (2/10)* continued

To prove that f is onto, you must prove

$$\forall y \in Y, \exists x \in X \text{ such that } f(x) = y.$$

Substituting the definition of f into the outline of a proof by the method of generalizing from the generic particular, you

suppose y is a real number

and **show** that there exists a real number x such that $y = 4x - 1$.

Example 7.2.5 – *Solution (3/10)* continued

Scratch Work: **If** such a real number *x* exists, then

$$
\begin{aligned}
4x - 1 &= y \\
4x &= y + 1 && \text{by adding 1 to both sides} \\
x &= \frac{y+1}{4} && \text{by dividing both sides by 4.}
\end{aligned}
$$

Thus *if* such a number *x* exists, it must equal $(y + 1)/4$.
Does such a number exist? Yes.

Example 7.2.5 – *Solution (4/10)* continued

To show this, let $x = (y + 1)/4$, and then make sure that (1) x is a real number and that (2) f really does send x to y.

The following formal answer summarizes this process.

Answer to (a):

If $f: \mathbf{R} \to \mathbf{R}$ is the function defined by the rule $f(x) = 4x - 1$ for each real number x, then f is onto.

Example 7.2.5 – *Solution (5/10)* continued

Proof: Let $y \in \mathbf{R}$. *[We must show that* $\exists x$ *in* $\mathbf{R}$ *such that* $f(x) = y$*.]* Let $x = (y + 1)/4$.

Then x is a real number since sums and quotients (other than by 0) of real numbers are real numbers. It follows that

$$f(x) = f\left(\frac{y+1}{4}\right) \qquad \text{by substitution}$$

$$= 4 \cdot \left(\frac{y+1}{4}\right) - 1 \qquad \text{by definition of } f$$

Example 7.2.5 – *Solution (6/10)* continued

$= (y + 1) - 1 = y$ by basic algebra,

[as was to be shown].

b. The function $h : \mathbf{Z} \to \mathbf{Z}$ is defined by the rule

$$\boxed{h(n) = 4n - 1 \quad \text{for each integer } n.}$$

To prove that h is onto, you must prove that

$\forall$ integer m, $\exists$ an integer n such that $h(n) = m$.

Example 7.2.5 – *Solution (7/10)* continued

Substituting the definition of h into the outline of a proof by the method of generalizing from the generic particular shows that you need to

suppose m is any integer

and **show** that there is an integer n with $4n - 1 = m$.

Can you reach what is to be shown from the supposition? No! If $4n - 1 = m$,

$$n = \frac{m+1}{4} \quad \text{by adding 1 and dividing by 4.}$$

Example 7.2.5 – *Solution (8/10)* continued

But n must be an integer. And when, for example, $m = 0$,

$$n = \frac{0+1}{4} = \frac{1}{4},$$

which is *not* an integer.

Thus, in trying to prove that h is onto, you run into difficulty, and this difficulty reveals a counterexample that shows h is not onto.

Example 7.2.5 – *Solution (9/10)* continued

This discussion is summarized in the following formal answer.

Answer to (b):

If the function $h: \mathbf{Z} \to \mathbf{Z}$ is defined by the rule $h(n) = 4n - 1$ for each integer n, then h is not onto.

Counterexample: The co-domain of h is $\mathbf{Z}$ and $0 \in \mathbf{Z}$.

But $h(n) \neq 0$ for any integer n.

Example 7.2.5 – *Solution (10/10)*continued

For if $h(n) = 0$, then

$$4n - 1 = 0 \qquad \text{by definition of } h$$

which implies that

$$4n = 1 \qquad \text{by adding 1 to both sides}$$

and so

$$n = \frac{1}{4} \qquad \text{by dividing both sides by 4.}$$

But $\frac{1}{4}$ is not an integer. Hence there is no integer n for which $f(n) = 0$, and thus f is not onto.

Relations between Exponential and Logarithmic Functions

Relations between Exponential and Logarithmic Functions (1/4)

For positive numbers $b \neq 1$, the **exponential function with base *b***, denoted $\exp_b$, is the function from $\mathbf{R}$ to $\mathbf{R}^+$ defined as follows. For each real number x,

$$\exp_b(x) = b^x,$$

where $b^0 = 1$ and $b^{-x} = 1/b^x$.

When working with the exponential function, it is useful to recall the laws of exponents from elementary algebra.

Relations between Exponential and Logarithmic Functions (2/4)

Laws of Exponents

If b and c are any positive real numbers and u and v are any real numbers, the following laws of exponents hold true:

$$b^u b^v = b^{u+v} \tag{7.2.1}$$

$$(b^u)^v = b^{uv} \tag{7.2.2}$$

$$\frac{b^u}{b^u} = b^{u-v} \tag{7.2.3}$$

$$(bc)^u = b^u c^u \tag{7.2.4}$$

It can be shown using calculus that both the exponential and logarithmic functions are one-to-one and onto.

Relations between Exponential and Logarithmic Functions (3/4)

Therefore, by definition of one-to-one, the following properties hold true:

For any positive real number b with $b \neq 1$,

$$\text{if } b^u = b^v \text{ then } u = v \text{ for all real numbers } u \text{ and } v, \qquad 7.2.5$$

and

$$\text{if } \log_b u = \log_b v \text{ then } u = v \text{ for all positive real numbers } u \text{ and } v. \qquad 7.2.6$$

These properties are used to derive many additional facts about exponents and logarithms.

Relations between Exponential and Logarithmic Functions (4/4)

In particular we have the following properties of logarithms.

Theorem 7.2.1 Properties of Logarithms

For any positive real numbers b, c, x, and y with $b \neq 1$ and $c \neq 1$ and for every real number a:

a. $\log_b(xy) = \log_b x + \log_b y$

b. $\log_b\left(\frac{x}{y}\right) = \log_b x - \log_b y$

c. $\log_b(x^a) = a \log_b x$

d. $\log_c x = \frac{\log_b x}{\log_b c}$

Example 7.2.7 – *Computing Logarithms with Base 2 on a Calculator*

In computer science it is often necessary to compute logarithms with base 2. Most calculators do not have keys to compute logarithms with base 2 but do have keys to compute logarithms with base 10 (called **common logarithms** and often denoted simply log) and logarithms with base *e* (called **natural logarithms** and usually denoted ln).

Suppose your calculator shows that $\ln 5 \cong 1.609437912$ and $\ln 2 \cong 0.6931471806$. Use Theorem 7.2.1(d) to find an approximate value for $\log_2 5$.

Example 7.2.7 – *Solution*

By Theorem 7.2.1(d),

$$\log_2 5 = \frac{\ln 5}{\ln 2} \cong \frac{1.609437912}{0.6931471806}$$

$$\cong 2.321928095.$$

One-to-One Correspondences

One-to-One Correspondences (1/3)

Consider a function $F : X \rightarrow Y$ that is both one-to-one and onto. Given any element x in X, there is a unique corresponding element $y = F(x)$ in Y (since F is a function).

Also given any element y in Y, there is an element x in X such that $F(x) = y$ (since F is onto) and there is only one such x (since F is one-to-one).

Thus, a function that is one-to-one and onto sets up a pairing between the elements of X and the elements of Y that matches each element of X with exactly one element of Y and each element of Y with exactly one element of X.

One-to-One Correspondences (2/3)

Such a pairing is called a *one-to-one correspondence* or *bijection* and is illustrated by the arrow diagram in Figure 7.2.5.

An Arrow Diagram for a One-to-One Correspondence

Figures 7.2.5

One-to-One Correspondences (3/3)

One-to-one correspondences are often used as aids to counting. The pairing of Figure 7.2.5, for example, shows that there are five elements in the set X.

Definition

A **one-to-one correspondence** (or **bijection**) from a set X to a set Y is a function $F: X \longrightarrow Y$ that is both one-to-one and onto.

Example 7.2.8 – *A Function from a Power Set to a Set of Strings (1/2)*

Let $\mathcal{P}(\{a, b\})$ be the set of all subsets of {*a*, *b*} and let *S* be the set of all strings of length 2 made up of 0's and 1's. Then $\mathcal{P}(\{a, b\}) = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$ and *S* = {00, 01, 10, 11}.

Define a function *h* from $\mathcal{P}(\{a, b\})$ to *S* as follows: Given any subset *A* of {*a*, *b*}, *a* is either in *A* or not in *A*, and *b* is either in *A* or not in *A*.

If *a* is in *A*, write a 1 in the first position of the string *h*(*A*); otherwise write a 0 there.

Example 7.2.8 – *A Function from a Power Set to a Set of Strings (2/2)*

Similarly, if b is in A, write a 1 in the second position of the string $h(A)$; otherwise write a 0 there. This definition is summarized in the following table.

h

Subset A of $\{a, b\}$	Status of a in A	Status of b in A	String $h(A)$ in S
$\varnothing$	not in	not in	00
$\{a\}$	in	not in	10
$\{b\}$	not in	in	01
$\{a, b\}$	in	in	11

Is h a one-to-one correspondence?

Example 7.2.8 – *Solution (1/2)*

The arrow diagram shown in Figure 7.2.6 shows clearly that *h* is a one-to-one correspondence.

Figures 7.2.6

Example 7.2.8 – *Solution (2/2)* continued

It is onto because each element of *S* has an arrow pointing to it.

It is one-to-one because each element of *S* has no more than one arrow pointing to it.

Inverse Functions

Inverse Functions (1/3)

If F is a one-to-one correspondence from a set X to a set Y, then there is a function from Y to X that "undoes" the action of F; that is, it sends each element of Y back to the element of X that it came from. This function is called the *inverse function* for F.

Theorem 7.2.2

Suppose $F: X \longrightarrow Y$ is a one-to-one correspondence; in other words, suppose F is one-to-one and onto. Then there is a function $\boldsymbol{F^{-1}: Y \longrightarrow X}$ that is defined as follows: Given any element y in Y,

$$\boldsymbol{F^{-1}(y)} = \text{that unique element } x \text{ in } X \text{ such that } F(x) \text{ equals } y.$$

Or, equivalently,

$$F^{-1}(y) = x \quad \Leftrightarrow \quad y = F(x).$$

Inverse Functions (2/3)

Definition

The function F^{-1} is called the **inverse function** for F.

Note that according to this definition, the logarithmic function with base $b > 0$ and $b \neq 1$ is the inverse of the exponential function with base b. The diagram that follows illustrates the fact that an inverse function sends each element back to where it came from.

Example 7.2.11 – *Finding an Inverse Function for a Function Given by an Arrow Diagram*

Define the inverse function for the one-to-one correspondence *h* given in Example 7.2.8.

Example 7.2.11 – *Solution*

The arrow diagram for h^{-1} is obtained by tracing the *h*-arrows back from *S* to $\mathscr{P}(\{a, b\})$ as shown below.

$h^{-1}(00) = \emptyset \quad h^{-1}(10) = \{a\}$

$h^{-1}(01) = \{b\} \quad h^{-1}(11) = \{a, b\}$

Example 7.2.12 – *Finding an Inverse Function for a Function Given in Words*

Define the inverse function for the one-to-one correspondence *g* given below:

$$g : T \to T$$

Example 7.2.12 – *Solution (1/2)*

The function $g : T \to T$ is defined by the following rule:
For all strings t in T,

$g(t)$ = the string obtained by writing the characters of t in reverse order.

Now if the characters of t are written in reverse order and then written in reverse order again, the original string is recovered. Thus given any string t in T,

$$g^{-1}(t) = \text{the unique string that, when written in reverse order, equals } t$$

Example 7.2.12 – *Solution (2/2)* continued

= the string obtained by writing the characters of *t* in reverse order

= *g*(*t*).

Hence $g^{-1}: T \to T$ is the same as *g*, or, in other words, $g^{-1} = g$.

Example 7.2.13 – *Finding an Inverse Function for a Function Given by a Formula*

The function $f : \mathbf{R} \to \mathbf{R}$ defined by the formula

$$f(x) = 4x - 1 \text{ for each real number } x$$

was shown to be one-to-one in Example 7.2.2 and onto in Example 7.2.5. Find its inverse function.

Example 7.2.13 – *Solution*

For any *[particular but arbitrarily chosen] y* in **R**, by definition of f^{-1},

$f^{-1}(y)$ = that unique real number x such that $f(x) = y$.

But $f(x) = y$

$\Leftrightarrow$ $4x - 1 = y$ by definition of f

$\Leftrightarrow$ $x = \dfrac{y+1}{4}$ by algebra.

Hence $f^{-1}(y) = \dfrac{y+1}{4}$.

Inverse Functions (3/3)

Theorem 7.2.3

If X and Y are sets and $F: X \longrightarrow Y$ is one-to-one and onto, then $F^{-1}: Y \longrightarrow X$ is also one-to-one and onto.

Example 7.2.14 – *Finding an Inverse Function for a Function of Two Variables*

Define the inverse function $F^{-1}\colon \mathbf{R} \times \mathbf{R} \to \mathbf{R} \times \mathbf{R}$ for the one-to-one correspondence given below:

$$F\colon \mathbf{R} \times \mathbf{R} \to \mathbf{R} \times \mathbf{R}$$

Example 7.2.14 – *Solution*

We know that

$$F\left(\frac{u+v}{2}, \frac{u-v}{2}\right) = (u, v).$$

Because F is one-to-one, this means that

$\left(\frac{u+v}{2}, \frac{u-v}{2}\right)$ is the unique ordered pair in the domain of F that is sent to (u, v) by F.

Thus, F^{-1} is defined as follows: For each ordered pair $(u, v) \in \mathbf{R} \times \mathbf{R}$,

$$F^{-1}(u, v) = \left(\frac{u+v}{2}, \frac{u-v}{2}\right).$$

CHAPTER 7

PROPERTIES OF FUNCTIONS

Copyright © Cengage Learning. All rights reserved.

7.3 Composition of Functions

Copyright © Cengage Learning. All rights reserved.

Composition of Functions (1/7)

Consider two functions, the successor function and the squaring function, both defined from **Z** to **Z**, and imagine that each is represented by a machine.

If the two machines are hooked up so that the output from the successor function is used as input to the squaring function, then they work together to operate as one larger machine.

In this larger machine, an integer *n* is first increased by 1 to obtain *n* + 1; then the quantity *n* + 1 is squared to obtain $(n+1)^2$.

Composition of Functions (2/7)

This is illustrated in the following drawing.

Composition of Functions (3/7)

Combining functions in this way is called *composing* them; the resulting function is called the *composition* of the two functions.

Note that the composition can be formed only if the output of the first function is acceptable input to the second function. That is, the range of the first function must be contained in the domain of the second function.

Composition of Functions (4/7)

Definition

Let $f: X \longrightarrow Y$ and $g: Y' \longrightarrow Z$ be functions with the property that the range of f is a subset of the domain of g. Define a new function $g \circ f: X \longrightarrow Z$ as follows:

$$(g \circ f)(x) = g(f(x)) \quad \text{for each } x \in X,$$

where $g \circ f$ is read "g circle f" and $g(f(x))$ is read "g of f of x." The function $g \circ f$ is called the **composition of f and g**.

Composition of Functions (5/7)

This definition is shown schematically below.

Example 7.3.1 – *Composition of Functions Defined by Formulas*

Let $f : \mathbf{Z} \to \mathbf{Z}$ be the successor function and let $g : \mathbf{Z} \to \mathbf{Z}$ be the squaring function. Then $f(n) = n + 1$ for each $n \in \mathbf{Z}$ and $g(n) = n^2$ for each $n \in \mathbf{Z}$.

a. Find the compositions $g \circ f$ and $f \circ g$.

b. Is $g \circ f = f \circ g$? Explain.

Example 7.3.1 – *Solution (1/3)*

a. Because *f* sends each integer to that integer plus 1 and *g* sends each integer to the square of that integer, it can be helpful to think of the action of *f* and *g* as follows:

f (any integer) = that integer + 1 *[even if the integer is g(n)].*

$g(\text{any integer}) = (\text{that integer})^2$ *[even if the integer is f(n)].*

Example 7.3.1 – *Solution (2/3)* continued

Thus the functions $g \circ f$ and $f \circ g$ are defined as follows:

$$(g \circ f)(n) = g(f(n)) = g(n+1) = (n+1)^2 \quad \text{for each } n \in \mathbf{Z},$$

and

$$(f \circ g)(n) = f(g(n)) = f(n^2) = n^2 + 1 \quad \text{for each } n \in \mathbf{Z}.$$

b. Two functions from one set to another are equal if, and only if, they always take the same values. In this case,

$$(g \circ f)(1) = (1+1)^2 = 4, \text{ whereas } (f \circ g)(1) = 1^2 + 1 = 2.$$

Example 7.3.1 – *Solution (3/3)*

continued

Thus the two functions $g \circ f$ and $f \circ g$ are not equal:

$$g \circ f \neq f \circ g.$$

Example 7.3.2 – *Composition of Functions Defined on Finite Sets*

Let $X = \{1, 2, 3\}$, $Y' = \{a, b, c, d\}$, $Y = \{a, b, c, d, e\}$, and $Z = \{x, y, z\}$. Define functions $f : X \to Y'$ and $g : Y \to Z$ by the arrow diagrams below.

Draw the arrow diagram for $g \circ f$. What is the range of $g \circ f$?

Example 7.3.2 – *Solution*

To find the arrow diagram for $g \circ f$, just trace the arrows all the way across from *X* to *Z* through *Y*. The result is shown below.

$$(g \circ f)(1) = g(f(1)) = g(c) = z$$
$$(g \circ f)(2) = g(f(2)) = g(b) = y$$
$$(g \circ f)(3) = g(f(3)) = g(a) = y$$

The range of $g \circ f$ is $\{y, z\}$.

Example 7.3.3 – *Composition with the Identity Function*

Let $X = \{a, b, c, d\}$ and $Y = \{u, v, w\}$, and suppose $f : X \to Y$ is given by the arrow diagram shown below.

Find $f \circ I_X$ and $I_Y \circ f$.

Example 7.3.3 – *Solution (1/2)*

The values of $f \circ I_X$ are obtained by tracing through the arrow diagram shown below.

$$(f \circ I_X)(a) = f(I_X(a)) = f(a) = u$$
$$(f \circ I_X)(b) = f(I_X(b)) = f(b) = v$$
$$(f \circ I_X)(c) = f(I_X(c)) = f(c) = v$$
$$(f \circ I_X)(d) = f(I_X(d)) = f(d) = u$$

Thus, for every element *x* in *X*,

$$(f \circ I_X)(x) = f(x).$$

Example 7.3.3 – *Solution (2/2)*

continued

By definition of equality of functions, this means that $f \circ I_X = f$.

Similarly, the equality $I_Y \circ f = f$ can be verified by tracing through the arrow diagram below for each *x* in *X* and noting that in each case, $(I_Y \circ f)(x) = f(x)$.

Composition of Functions (6/7)

Theorem 7.3.1 Composition with an Identity Function

If f is a function from a set X to a set Y, and I_X is the identity function on X, and I_Y is the identity function on Y, then

$$\text{(a) } f \circ I_X = f \quad \text{and} \quad \text{(b) } I_Y \circ f = f.$$

Example 7.3.4 – *Composing a Function with Its Inverse (1/3)*

Let $X = \{a, b, c\}$ and $Y = \{x, y, z\}$. Define $f : X \to Y$ by the following arrow diagram.

You can see from the diagram that f is one-to-one and onto.

Example 7.3.4 – *Composing a Function with Its Inverse (2/3)*

Thus f^{-1} exists and is found by tracing the arrows backwards, as shown below.

Now $f^{-1} \circ f$ is found by following the arrows from *X* to *Y* by *f* and back to *X* by f^{-1}. If you do this, you will see that

$$(f^{-1} \circ f)(a) = f^{-1}(f(a)) = f^{-1}(z) = a$$

Example 7.3.4 – *Composing a Function with Its Inverse (3/3)*

$$(f^{-1} \circ f)(b) = f^{-1}(f(b)) = f^{-1}(x) = b$$

and

$$(f^{-1} \circ f)(c) = f^{-1}(f(c)) = f^{-1}(y) = c.$$

Thus the composition of *f* and f^{-1} sends each element to itself. So by definition of the identity function,

$$f^{-1} \circ f = I_X.$$

In a similar way, you can see that

$$f \circ f^{-1} = I_Y.$$

Composition of Functions (7/7)

Theorem 7.3.2 Composition of a Function with Its Inverse

If $f: X \longrightarrow Y$ is a one-to-one and onto function with inverse function $f^{-1}: Y \longrightarrow X$, then

$$\text{(a) } f^{-1} \circ f = I_X \quad \text{and} \quad \text{(b) } f \circ f^{-1} = I_Y.$$

Composition of One-to-One Functions

Composition of One-to-One Functions (1/4)

The composition of functions interacts in interesting ways with the properties of being one-to-one and onto.

What happens, for instance, when two one-to-one functions are composed?

Must their composition be one-to-one?

Composition of One-to-One Functions (2/4)

For example, let $X = \{a, b, c\}$, $Y = \{w, x, y, z\}$, and $Z = \{1, 2, 3, 4, 5\}$, and define one-to-one functions $f : X \to Y$ and $g : Y \to Z$ as shown in the arrow diagrams of Figure 7.3.1.

Figure 7.3.1

Composition of One-to-One Functions (3/4)

Then $g \circ f$ is the function with the arrow diagram shown in Figure 7.3.2.

Figure 7.3.2

From the diagram it is clear that for these particular functions, the composition is one-to-one.

Composition of One-to-One Functions (4/4)

This result is no accident. It turns out that the composition of two one-to-one functions is always one-to-one.

Theorem 7.3.3

If $f: X \longrightarrow Y$ and $g: Y \longrightarrow Z$ are both one-to-one functions, then $g \circ f$ is one-to-one.

Composition of Onto Functions

Composition of Onto Functions (1/3)

Now consider what happens when two onto functions are composed. For example, let $X = \{a, b, c, d, e\}$, $Y = \{w, x, y, z\}$, and $Z = \{1, 2, 3\}$. Define onto functions f and g by the following arrow diagrams.

Composition of Onto Functions (2/3)

Then $g \circ f$ is the function with the arrow diagram shown below.

It is clear from the diagram that $g \circ f$ is onto.

Composition of Onto Functions (3/3)

It turns out that the composition of any two onto functions (that can be composed) is onto.

Theorem 7.3.4

If $f: X \longrightarrow Y$ and $g: Y \longrightarrow Z$ are both onto functions, then $g \circ f$ is onto.

CHAPTER 7

PROPERTIES OF FUNCTIONS

Copyright © Cengage Learning. All rights reserved.

7.4 Cardinality with Applications to Computability

Copyright © Cengage Learning. All rights reserved.

Cardinality with Applications to Computability (1/4)

The term *cardinal number* refers to the size of a set (“This set has *eight* elements”), whereas the term *ordinal number* refers to the order of an element in a sequence (“This is the *eighth* element in the row”). The mathematical definition of cardinal number is based on the use of fingers or tally marks to represent numbers.

For instance, small children often indicate their age by holding up the same number of fingers as the years of their life, and adults frequently use one tally mark to correspond to each vote received by a candidate in an election.

Cardinality with Applications to Computability (2/4)

A pairing of the elements of two sets is called a one-to-one correspondence. We say that two finite sets whose elements can be paired by a one-to-one correspondence have the *same size*.

This is illustrated by the following diagram.

The elements of set A can be put into one-to-one correspondence with the elements of B.

Cardinality with Applications to Computability (3/4)

Definition

Let A and B be any sets. $\boldsymbol{A}$ **has the same cardinality as** $\boldsymbol{B}$ if, and only if, there is a one-to-one correspondence from A to B. In other words, A has the same cardinality as B if, and only if, there is a function f from A to B that is one-to-one and onto.

The following theorem gives some basic properties of cardinality,

Theorem 7.4.1 Properties of Cardinality

For all sets A, B, and C:

a. **Reflexive property of cardinality:** A has the same cardinality as A.

b. **Symmetric property of cardinality:** If A has the same cardinality as B, then B has the same cardinality as A.

c. **Transitive property of cardinality:** If A has the same cardinality as B and B has the same cardinality as C, then A has the same cardinality as C.

Cardinality with Applications to Computability (4/4)

Definition

A and B **have the same cardinality** if, and only if, A has the same cardinality as B or B has the same cardinality as A.

Example 7.4.1 – *An Infinite Set and a Proper Subset Can Have the Same Cardinality*

Let $2\mathbf{Z}$ be the set of all even integers. Prove that $2\mathbf{Z}$ and $\mathbf{Z}$ have the same cardinality.

Example 7.4.1 – *Solution (1/3)*

Consider the function H from $\mathbf{Z}$ to $2\mathbf{Z}$ defined as follows:

$$H(n) = 2n \text{ for each } n \in \mathbf{Z}.$$

A (partial) arrow diagram for H is shown below.

Example 7.4.1 – *Solution (2/3)* continued

To show that H is one-to-one, suppose $H(n_1) = H(n_2)$ for some integers n_1 and n_2.

Then $2n_1 = 2n_2$ by definition of H, and dividing both sides by 2 gives $n_1 = n_2$. Hence h is one-to-one.

To show that H is onto, suppose m is any element of $2\mathbf{Z}$. Then m is an even integer, and so $m = 2k$ for some integer k. It follows that $H(k) = 2k = m$.

Example 7.4.1 – *Solution (3/3)* continued

Thus there exists k in $\mathbf{Z}$ with $H(k) = m$, and hence H is onto.

Therefore, by definition of cardinality, $\mathbf{Z}$ and $2\mathbf{Z}$ have the same cardinality.

Countable Sets

Countable Sets (1/3)

The most basic of all infinite sets is $\mathbf{Z}^+$ the set of counting numbers {1, 2, 3, 4, …}. A set *A* having the same cardinality as this set is called *countably infinite*.

The reason is that the one-to-one correspondence between the two sets can be used to "count" the elements of *A*: If *F* is a one-to-one and onto function from $\mathbf{Z}^+$ to *A*, then *F*(1) can be designated as the first element of *A*, *F*(2) as the second element of *A*, *F*(3) as the third element of *A*, and so forth.

Countable Sets (2/3)

This is illustrated graphically in Figure 7.4.1 below.

"Counting" a Countably Infinite Set

Figure 7.4.1

Countable Sets (3/3)

Because *F* is one-to-one, no element is ever counted twice, and because it is onto, every element of *A* is counted eventually.

Definition

A set is **finite** if, and only if, it is the empty set or can be put into one-to-one correspondence with a set of the form $\{1, 2, \ldots, n\}$ for some positive integer n. A set is **countably infinite** if, and only if, it has the same cardinality as the set of positive integers $\mathbf{Z}^+$. A set is **countable** if, and only if, it is finite or countably infinite. A set that is not countable is called **uncountable**.

Example 7.4.2 – *Countability of Z, the Set of All Integers*

Show that **Z**, the set of all integers, is countable.

Example 7.4.2 – *Solution (1/5)*

The set **Z** is certainly not finite, so if it is countable, it must be because it is countably infinite. To show that **Z** is countably infinite, find a function from the positive integers $\mathbf{Z}^+$ to **Z** that is one-to-one and onto.

This seems to contradict common sense because judging from the diagram below, there appear to be more than twice as many integers than there are positive integers.

$$\underbrace{\cdots -5 \quad -4 \quad -3 \quad -2 \quad -1 \quad 0 \quad \overbrace{1 \quad 2 \quad 3 \quad 4 \quad 5 \cdots}^{\text{positive integers}}}_{\text{all integers}}$$

Example 7.4.2 – *Solution (2/5)* continued

But you were alerted that results in this section might be surprising.

Try to think how you might “count” the set of all integers.

One way is to start in the middle and work outward systematically.

Example 7.4.2 – *Solution (3/5)* continued

Let the first integer be 0, the second 1, the third −1, the fourth 2, the fifth −2, and so forth, as shown in Figure 7.4.2, starting at 0 and swinging outward in back-and-forth arcs from positive to negative integers and back again, picking up one additional integer at each swing.

"Counting" the Set of All Integers

Figure 7.4.2

Example 7.4.2 – *Solution (4/5)* continued

It is clear from the diagram that no integer is counted twice (so the function is one-to-one) and every integer is counted eventually (so the function is onto).

Consequently, this diagram defines a function from $\mathbf{Z}^+$ to **Z** that is one-to-one and onto.

Even though in one sense there seem to be more integers than positive integers, the elements of the two sets can be paired up one for one.

Example 7.4.2 – *Solution (5/5)* continued

It follows by definition of cardinality that $\mathbf{Z}^+$ has the same cardinality as **Z**. Thus **Z** is countably infinite and hence countable.

The diagrammatic description of the previous function is acceptable as given. You can check, however, that the function can also be described by the explicit formula

$$F(n) = \begin{cases} \dfrac{n}{2} & \text{if } n \text{ is an even positive integer} \\ -\dfrac{n-1}{2} & \text{if } n \text{ is an odd positive integer.} \end{cases}$$

The Search for Larger Infinities: The Cantor Diagonalization Process

Example 7.4.4 – *The Set of All Positive Rational Numbers Is Countable*

Show that $\mathbf{Q}^+$, the set of all positive rational numbers, is countable.

Example 7.4.4 – *Solution (1/4)*

Display the elements of the set $\mathbf{Q}^+$ in a grid as shown in Figure 7.4.3 below.

$$
\begin{array}{cccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \cdots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \frac{2}{4} & \frac{2}{5} & \frac{2}{6} \cdots \\
\frac{3}{1} & \frac{3}{2} & \frac{3}{3} & \frac{3}{4} & \frac{3}{5} & \frac{3}{6} \cdots \\
\frac{4}{1} & \frac{4}{2} & \frac{4}{3} & \frac{4}{4} & \frac{4}{5} & \frac{4}{6} \cdots \\
\frac{5}{1} & \frac{5}{2} & \frac{5}{3} & \frac{5}{4} & \frac{5}{5} & \frac{5}{6} \cdots \\
\frac{6}{1} & \frac{6}{2} & \frac{6}{3} & \frac{6}{4} & \frac{6}{5} & \frac{6}{6} \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
$$

Figure 7.4.3

Example 7.4.4 – *Solution (2/4)*

continued

Define a function *F* from $\mathbf{Z}^+$ to $\mathbf{Q}^+$ by starting to count at $\frac{1}{1}$ and following the arrows as indicated, skipping over any number that has already been counted.

To be specific: Set $F(1) = 1$, $F(2) = \frac{1}{2}$, $F(3) = \frac{2}{1}$, and $F(4) = \frac{3}{1}$. Then skip $\frac{2}{2}$ since $\frac{2}{2} = \frac{1}{1}$, which was counted already. After that, set $F(5) = \frac{1}{3}$, $F(6) = \frac{1}{4}$, $F(7) = \frac{2}{3}$, $F(8) = \frac{3}{2}$, $F(9) = \frac{4}{1}$, and $F(10) = \frac{5}{1}$. Then skip $\frac{4}{2}$, $\frac{3}{3}$, and $\frac{2}{4}$ (since $\frac{4}{2} = \frac{2}{1}$, $\frac{3}{3} = \frac{1}{1}$, and $\frac{2}{4} = \frac{1}{2}$), and set $F(11) = \frac{1}{5}$.

Example 7.4.4 – *Solution (3/4)* continued

Continue in this way, defining $F(n)$ for each positive integer n.

Note that every positive rational number appears somewhere in the grid, and the counting procedure is set up so that every point in the grid is reached eventually. Thus the function F is onto.

Also, skipping numbers that have already been counted ensures that no number is counted twice.

Example 7.4.4 – *Solution (4/4)*

continued

Thus *F* is one-to-one. Consequently, *F* is a function from $\mathbf{Z}^+$ to $\mathbf{Q}^+$ that is one-to-one and onto, and so $\mathbf{Q}^+$ is countably infinite and hence countable.

The Search for Larger Infinities: The Cantor Diagonalization Process (1/4)

In 1874 the German mathematician Georg Cantor achieved success in the search for a larger infinity by showing that the set of all real numbers is uncountable. His method of proof was somewhat complicated, however.

We give a proof of the uncountability of the set of all real numbers between 0 and 1 using a simpler technique introduced by Cantor in 1891 and now called the **Cantor diagonalization process**.

The Search for Larger Infinities: The Cantor Diagonalization Process (2/4)

Over the intervening years, this technique and variations on it have been used to establish a number of important results in logic and the theory of computation.

Before stating and proving Cantor's theorem, we note that every real number, which is a measure of location on a number line, can be represented by a decimal expansion of the form

$$a_0.a_1a_2a_3\ldots,$$

where a_0 is an integer (positive, negative, or zero) and for each $i \geq 1$, a_i is an integer from 0 through 9.

The Search for Larger Infinities: The Cantor Diagonalization Process (3/4)

Theorem 7.4.2 (Cantor)

The set of all real numbers between 0 and 1 is uncountable.

Along with demonstrating the existence of an uncountable set, Cantor developed a whole arithmetic theory of infinite sets of various sizes.

One of the most basic theorems of the theory states that any subset of a countable set is countable.

The Search for Larger Infinities: The Cantor Diagonalization Process (4/4)

Theorem 7.4.3

Any subset of any countable set is countable.

Corollary 7.4.4

Any set with an uncountable subset is uncountable.

Example 7.4.5 – *The Cardinality of the Set of All Real Numbers*

Show that the set of all real numbers has the same cardinality as the set of real numbers between 0 and 1.

Example 7.4.5 – *Solution (1/4)*

Let S be the open interval of real numbers between 0 and 1:

$$S = \{x \in \mathbf{R} \mid 0 < x < 1\}.$$

Imagine picking up S and bending it into a circle as shown below.

Example 7.4.5 – *Solution (2/4)* continued

Since S does not include either endpoint 0 or 1, the top-most point of the circle is omitted from the drawing.

Define a function $F: S \to \mathbf{R}$ as follows:

Draw a number line and place the interval, S, somewhat enlarged and bent into a circle, tangent to the number line at the point 0. This is shown below.

Example 7.4.5 – *Solution (3/4)* continued

For each point x on the circle representing S, draw a straight line L through the top-most point of the circle and x, and let $F(x)$ be the point of intersection of L and the number line. ($F(x)$ is called the *projection* of x onto the number line.)

It is clear from the geometry of the situation that distinct points on the circle go to distinct points on the number line, so F is one-to-one.

In addition, given any point y on the number line, a line can be drawn through y and the top-most point of the circle.

Example 7.4.5 – *Solution (4/4)* continued

This line must intersect the circle at some point x, and, by definition, $y = F(x)$.

Thus F is onto. Hence F is a one-to-one correspondence from S to $\mathbf{R}$, and so S and $\mathbf{R}$ have the same cardinality.

Application: Cardinality and Computability

Example 7.4.6 – *Countability of the Set of Computer Programs in a Computer Language*

Show that the set of all computer programs in a given computer language is countable.

Example 7.4.6 – *Solution (1/5)*

This result is a consequence of the fact that any computer program in any language can be regarded as a finite string of symbols in the (finite) alphabet of the language.

Given any computer language, let P be the set of all computer programs in the language. Either P is finite or P is infinite. If P is finite, then P is countable and we are done.

If P is infinite, set up a binary code to translate the symbols of the alphabet of the language into strings of 0's and 1's.

Example 7.4.6 – *Solution (2/5)* continued

(For instance, either the seven-bit American Standard Code for Information Interchange, known as ASCII, or the eight-bit Extended Binary-Coded Decimal Interchange Code, known as EBCDIC, might be used.)

For each program in *P*, use the code to translate all the symbols in the program into 0's and 1's.

Order these strings by length, putting shorter before longer, and order all strings of a given length by regarding each string as a binary number and writing the numbers in ascending order.

Example 7.4.6 – *Solution (3/5)* continued

Define a function $F: \mathbf{Z}^+ \to P$ by specifying that

$F(n)$ = the nth program in the list for each $n \in \mathbf{Z}^+$.

By construction, F is one-to-one and onto, and so P is countably infinite and hence countable.

As a simple example, suppose the following are all the programs in P that translate into bit strings of length less than or equal to 5:

10111, 11, 0010, 1011, 01, 00100, 1010, 00010.

Example 7.4.6 – *Solution (4/5)* continued

Ordering these by length gives

length 2: 11, 01

length 4: 0010, 1011, 1010

length 5: 10111, 00100, 00010

And ordering those of each given length by the size of the binary number they represent gives

$01 = F(1)$

$11 = F(2)$

$0010 = F(3)$

$1010 = F(4)$

Example 7.4.6 – *Solution (5/5)* continued

$$1011 = F(5)$$
$$00010 = F(6)$$
$$00100 = F(7)$$
$$10111 = F(8)$$

Note that when viewed purely as numbers and ignoring leading zeros, 0010 = 00010.

This illustrates why it is important to order the strings by length before arranging them in ascending numeric order because otherwise the values of *F* would not be uniquely determined.

Example 7.4.7 – *The Cardinality of a Set of Functions and Computability*

a. Let T be the set of all functions from the positive integers to the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Show that T is uncountable.

b. Derive the consequence that there are noncomputable functions. Specifically, show that for any computer language there must be a function F from $\mathbf{Z}^+$ to {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} with the property that no computer program can be written in the language to take arbitrary values as input and output the corresponding function values.

Example 7.4.7 – *Solution (1/4)*

a. Let S be the set of all real numbers between 0 and 1. As noted before, any number in S can be represented in the form

$$0.a_1a_2a_3\ldots a_n\ldots,$$

where each a_i is an integer from 0 to 9. This representation is unique if decimals that end in all 9's are omitted.

Define a function F from S to a subset of T as follows:

$F(0.a_1a_2a_3\ldots a_n\ldots)$ = the function that sends each positive integer n to a_n.

Example 7.4.7 – *Solution (2/4)* continued

Choose the co-domain of F to be exactly that subset of T that makes F onto, recalling that T is the set of all functions from $\mathbb{Z}^+$ to {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

In other words, define the co-domain of F to equal the image of F.

Now F is one-to-one because in order for the functions $F(x_1)$ and $F(x_2)$ to be equal, they must have the same value for each positive integer, and so each decimal digit of x_1 must equal the corresponding decimal digit of x_2, which implies that $x_1 = x_2$.

Example 7.4.7 – *Solution (3/4)* continued

Thus F is a one-to-one correspondence from S to a subset of T. But S is uncountable by Theorem 7.4.2.

Theorem 7.4.2 (Cantor)

The set of all real numbers between 0 and 1 is uncountable.

Hence T has an uncountable subset, and so, by Corollary 7.4.4, T is uncountable.

Corollary 7.4.4

Any set with an uncountable subset is uncountable.

Example 7.4.7 – *Solution (4/4)*

continued

b. Part (a) shows that the set *T* of all functions from $\mathbf{Z}^+$ to {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} is uncountable.

But, by Example 7.4.6, given any computer language, the set of all programs in that language is countable.

Consequently, in any computer language there are not enough programs to compute values of every function in *T*. There must exist functions that are not computable!

CHAPTER 8

PROPERTIES OF RELATIONS

Copyright © Cengage Learning. All rights reserved.

8.1 Relations on Sets

Copyright © Cengage Learning. All rights reserved.

Example 8.1.1 – *The Less-than Relation for Real Numbers*

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

$$x \, L \, y \Leftrightarrow x < y.$$

a. Is $57 \, L \, 53$? b. Is $(-17) \, L \, (-14)$?

c. Is $143 \, L \, 143$? d. Is $(-35) \, L \, 1$?

e. Draw the graph of L as a subset of the Cartesian plane $\mathbf{R} \times \mathbf{R}$.

Example 8.1.1 – *Solution*

a. No, $57 > 53$. b. Yes, $-17 < -14$.

c. No, $143 = 143$. d. Yes, $-35 < 1$.

e. For each value of x, all the points (x, y) with $y > x$ are on the graph. So the graph consists of all the points above the line $x = y$.

Example 8.1.2 – *The Congruence Modulo 2 Relation*

Define a relation E from $\mathbf{Z}$ to $\mathbf{Z}$ as follows: For every $(m, n) \in \mathbf{Z} \times \mathbf{Z}$,

$$m \; E \; n \Leftrightarrow m - n \text{ is even.}$$

a. Is $4 \; E \; 0$? Is $2 \; E \; 6$? Is $3 \; E \; (-3)$? Is $5 \; E \; 2$?

b. List five integers that are related by E to 1.

c. Prove that if n is any odd integer, then $n \; E \; 1$.

Example 8.1.2 – *Solution (1/4)*

a. Yes, 4 E 0 because 4 − 0 = 4 and 4 is even.

Yes, 2 E 6 because 2 − 6 = −4 and −4 is even.

Yes, 3 E (−3) because 3 − (−3) = 6 and 6 is even.

No, $5 \not E\, 2$ because 5 − 2 = 3 and 3 is not even.

Example 8.1.2 – *Solution (2/4)* continued

b. There are many such lists. One is

1 because $1 - 1 = 0$ is even.

3 because $3 - 1 = 2$ is even.

5 because $5 - 1 = 4$ is even.

-1 because $-1 - 1 = -2$ is even.

-3 because $-3 - 1 = -4$ is even.

Example 8.1.2 – *Solution (3/4)* continued

c. **Proof:** Suppose n is any odd integer. Then $n = 2k + 1$ for some integer k. Now by definition of E, $n\ E\ 1$ if, and only if, $n - 1$ is even.

But by substitution,

$$n - 1 = (2k + 1) - 1 = 2k,$$

and since k is an integer, $2k$ is even. Hence $n\ E\ 1$ *[as was to be shown]*.

Example 8.1.2 – *Solution (4/4)* continued

It can be shown that integers m and n are related by E if, and only if, $m \bmod 2 = n \bmod 2$ (that is, both are even or both are odd).

When this occurs m and n are said to be **congruent modulo 2**.

Example 8.1.3 – *A Relation on a Power Set*

Let $X = \{a, b, c\}$.

Then $\mathscr{P}(X) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$.

Define a relation **S** from $\mathscr{P}(X)$ to $\mathscr{P}(X)$ as follows: For all sets A and B $\mathscr{P}(X)$ (that is, for all subsets A and B of X),

A **S** $B \Longleftrightarrow A$ has at least as many elements as B.

a. Is $\{a, b\}$ **S** $\{b, c\}$?

b. Is $\{a\}$ **S** $\varnothing$?

c. Is $\{b, c\}$ **S** $\{a, b, c\}$?

d. Is $\{c\}$ **S** $\{a\}$?

Example 8.1.3 – *Solution*

a. Yes, both sets have two elements.

b. Yes, $\{a\}$ has one element and $\varnothing$ has zero elements, and $1 \geq 0$.

c. No, $\{b, c\}$ has two elements and $\{a, b, c\}$ has three elements and $2 < 3$.

d. Yes, both sets have one element.

The Inverse of a Relation

The Inverse of a Relation (1/1)

If R is a relation from A to B, then a relation R^{-1} from B to A can be defined by interchanging the elements of all the ordered pairs of R.

Definition

Let R be a relation from A to B. Define the inverse relation R^{-1} from B to A as follows:

$$R^{-1} = \{(y, x) \in B \times A \mid (x, y) \in R\}.$$

This definition can be written operationally as follows:

$$\text{For all } x \in A \text{ and } y \in B, \quad (y, x) \in R^{-1} \quad \Leftrightarrow \quad (x, y) \in R.$$

Example 8.1.4 – *The Inverse of a Finite Relation*

Let $A = \{2, 3, 4\}$ and $B = \{2, 6, 8\}$, and let R be the "divides" relation from A to B: For every ordered pair $(x, y) \in A \times B$,

$$x\,R\,y \quad \Leftrightarrow \quad x \mid y \quad x \text{ divides } y.$$

a. State explicitly which ordered pairs are in R and R^{-1}, and draw arrow diagrams for R and R^{-1}.

b. Describe R^{-1} in words.

Example 8.1.4 – *Solution (1/3)*

a. $R = \{(2, 2), (2, 6), (2, 8), (3, 6), (4, 8)\}$

$$R^{-1} = \{(2, 2), (6, 2), (8, 2), (6, 3), (8, 4)\}$$

Example 8.1.4 – *Solution (2/3)* continued

To draw the arrow diagram for R^{-1}, you can copy the arrow diagram for *R* but reverse the directions of the arrows.

Or you can redraw the diagram so that *B* is on the left.

Example 8.1.4 – *Solution (3/3)*

continued

b. R^{-1} can be described in words as follows: For every ordered pair $(y, x) \in B \times A$,

$y \quad R^{-1} x \Leftrightarrow y$ is a multiple of x.

Example 8.1.5 – *The Inverse of an Infinite Relation*

Define a relation R from $\mathbf{R}$ to $\mathbf{R}$ as follows: For every ordered pair $(x, y) \in \mathbf{R} \times \mathbf{R}$,

$$x\,R\,y \quad \Leftrightarrow \quad y = 2|x|.$$

Draw the graphs of R and R^{-1} in the Cartesian plane. Is R^{-1} a function?

Example 8.1.5 – *Solution (1/2)*

A point (v, u) is on the graph of R^{-1} if, and only if, (u, v) is on the graph of R.

Note that if $x \geq 0$, then the graph of $y = 2|x| = 2x$ is a straight line with slope 2.

And if $x < 0$, then the graph of $y = 2|x| = 2(-x) = -2x$ is a straight line with slope −2.

Example 8.1.5 – *Solution (2/2)* continued

Some sample values are tabulated and the graphs are shown below.

$R = \{(x, y) | y = 2|x|\}$

x	y
0	0
1	2
−1	2
2	4
−2	4

1st coordinate 2nd coordinate

$R^{-1} = \{(y, x) | y = 2|x|\}$

y	x
0	0
2	1
2	−1
4	2
4	−2

1st coordinate 2nd coordinate

R^{-1} is not a function because, for instance, both (2, 1) and (2, −1) are in R^{-1}.

Directed Graph of a Relation

Directed Graph of a Relation (1/2)

Definition

A **relation on a set** A is a relation from A to A.

When a relation R is defined *on* a set A, the arrow diagram of the relation can be modified so that it becomes a **directed graph**.

Instead of representing A as two separate sets of points, represent A only once, and draw an arrow from each point of A to each related point.

Directed Graph of a Relation (2/2)

As with an ordinary arrow diagram,

For all points x and y in A,

$$\text{there is an arrow from } x \text{ to } y \quad \Leftrightarrow \quad x\,R\,y \quad \Leftrightarrow \quad (x, y) \in R.$$

Example 8.1.6 – *Directed Graph of a Relation*

Let $A = \{3, 4, 5, 6, 7, 8\}$ and define a relation R on A as follows: For every $x, y \in A$,

$$x\,R\,y \quad \Leftrightarrow \quad 2 \mid (x - y).$$

Draw the directed graph of R.

Example 8.1.6 – *Solution (1/2)*

Note that 3 *R* 3 because 3 − 3 = 0 and $2|0$ since 0 = 2 ⋅ 0. Thus there is a loop from 3 to itself. Similarly, there is a loop from 4 to itself, from 5 to itself, and so forth, since the difference of each integer with itself is 0, and $2|0$.

Note also that 3 *R* 5 because 3 − 5 = −2 = 2 ⋅ (−1). And 5 *R* 3 because 5 − 3 = 2 = 2 ⋅ 1. Hence there is an arrow from 3 to 5 and also an arrow from 5 to 3.

Example 8.1.6 – *Solution (2/2)*

continued

The other arrows in the directed graph, as shown below, are obtained by similar reasoning.

N-ary Relations and Relational Databases

N-*ary Relations and Relational Databases (1/1)*

A special group of relations, called *n*-ary relations, form the mathematical foundation for relational database theory. Just as a binary relation is a subset of a Cartesian product of two sets, an *n*-ary relation is a subset of a Cartesian product of *n* sets.

Definition

Given sets $A_1, A_2, \ldots, A_n$, an ***n*-ary relation** R on $A_1 \times A_2 \times \cdots \times A_n$ is a subset of $A_1 \times A_2 \times \cdots \times A_n$. The special cases of 2-ary, 3-ary, and 4-ary relations are called **binary, ternary**, and **quaternary relations**, respectively.

Example 8.1.7 – *A Simple Database (1/6)*

The following is a radically simplified version of a database that might be used in a hospital. Let A_1 be a set of positive integers, A_2 a set of alphabetic character strings, A_3 a set of numeric character strings, and A_4 a set of alphabetic character strings. Define a quaternary relation R on $A_1 \times A_2 \times A_3 \times A_4$ as follows:

$(a_1, a_2, a_3, a_4) \in R \Longleftrightarrow$ a patient with patient ID number a_1, named a_2, was admitted on date a_3, with primary diagnosis a_4.

Example 8.1.7 – *A Simple Database (2/6)*

At a particular hospital, this relation might contain the following 4-tuples:

(011985, John Schmidt, 020719, asthma)

(574329, Tak Kurosawa, 011419, pneumonia)

(466581, Mary Lazars, 010319, appendicitis)

(008352, Joan Kaplan, 112419, gastritis)

(011985, John Schmidt, 021719, pneumonia)

(244388, Sarah Wu, 010319, broken leg)

(778400, Jamal Baskers, 122719, appendicitis)

Example 8.1.7 – *A Simple Database (3/6)*

In discussions of relational databases, the *n*-tuples are normally thought of as being written in tables.

Each row of the table corresponds to one *n*-tuple, and the header for each column gives the descriptive attribute for the elements in the column.

Operations within a database allow the data to be manipulated in many different ways.

Example 8.1.7 – *A Simple Database (4/6)*

For example, in the database language SQL, if the above database is denoted *S*, the result of the query

SELECT Patient_ID#, Name FROM *S* WHERE

Admission_Date = 010319

would be a list of the ID numbers and names of all patients admitted on 01-03-19:

466581	Mary Lazars
244388	Sarah Wu

Example 8.1.7 – *A Simple Database (5/6)*

This is obtained by taking the intersection of the set $A_1 \times A_2 \times \{010319\} \times A_4$ with the database and then projecting onto the first two coordinates.

Similarly, SELECT can be used to obtain a list of all admission dates of a given patient. For John Schmidt this list is

02–07–19

02–17–19

Example 8.1.7 – *A Simple Database (6/6)*

Individual entries in a database can be added, deleted, or updated, and most databases can sort data entries in various ways.

In addition, entire databases can be merged, and the entries common to two databases can be moved to a new database.

CHAPTER 8

PROPERTIES OF RELATIONS

Copyright © Cengage Learning. All rights reserved.

8.2 Reflexivity, Symmetry, and Transitivity

Copyright © Cengage Learning. All rights reserved.

Reflexivity, Symmetry, and Transitivity (1/8)

Let $A = \{2, 3, 4, 6, 7, 9\}$ and define a relation R on A as follows: For every $x, y \in A$,

$$x\,R\,y \quad \Leftrightarrow \quad 3 \mid (x - y).$$

Then $2\,R\,2$ because $2 - 2 = 0$, and $3 \mid 0$. Similarly, $3\,R\,3$, $4\,R\,4$, $6\,R\,6$, $7\,R\,7$, and $9\,R\,9$.

Also $6\,R\,3$ because $6 - 3 = 3$, and $3 \mid 3$. And $3\,R\,6$ because $3 - 6 = -(6 - 3) = -3$, and $3 \mid (-3)$. Similarly, $3\,R\,9$, $9\,R\,3$, $6\,R\,9$, $9\,R\,6$, $4\,R\,7$, and $7\,R\,4$.

Reflexivity, Symmetry, and Transitivity (2/8)

Thus the directed graph for *R* has the appearance shown below.

Reflexivity, Symmetry, and Transitivity (3/8)

This graph has three important properties:

1. Each point of the graph has an arrow looping around from it and going back to it.
2. In each case where there is an arrow going from one point to a second, there is an arrow going from the second point back to the first.
3. In each case where there is an arrow going from one point to a second and from the second point to a third, there is an arrow going from the first point to the third. That is, there are no “incomplete directed triangles” in the graph.

Reflexivity, Symmetry, and Transitivity (4/8)

Properties (1), (2), and (3) correspond to properties of general relations called *reflexivity*, *symmetry*, and *transitivity*.

Definition

Let R be a relation on a set A.

1. R is **reflexive** if, and only if, for every $x \in A$, $x\,R\,x$.
2. R is **symmetric** if, and only if, for every $x, y \in A$, if $x\,R\,y$ then $y\,R\,x$.
3. R is **transitive** if, and only if, for every $x, y, z \in A$, if $x\,R\,y$ and $y\,R\,z$ then $x\,R\,z$.

Reflexivity, Symmetry, and Transitivity (5/8)

The equivalence of the expressions $x\ R\ y$ and $(x, y) \in R$ for every x and y in A, the reflexive, symmetric, and transitive properties can also be written as follows:

1. R is reflexive $\Leftrightarrow$ for every x in A, $(x, x) \in R$.
2. R is symmetric $\Leftrightarrow$ for every x and y in A, ***if*** $(x, y) \in R$ then $(y, x) \in R$.
3. R is transitive $\Leftrightarrow$ for every x, y, and z in A, ***if*** $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$.

Reflexivity, Symmetry, and Transitivity (6/8)

In informal terms, properties (1)–(3) say the following:

1. **Reflexive:** Each element is related to itself.
2. **Symmetric:** If any one element is related to any other element, then the second element is related to the first.
3. **Transitive:** If any one element is related to a second and that second element is related to a third, then the first element is related to the third.

Note that the definitions of reflexivity, symmetry, and transitivity are universal statements.

Reflexivity, Symmetry, and Transitivity (7/8)

This means that to prove a relation has one of the properties, you use either the method of exhaustion or the method of generalizing from the generic particular.

Recall that the negation of a universal statement is existential. Hence if R is a relation on a set A, then

1. R is **not reflexive** $\Leftrightarrow$ there is an element x in A such that $x \not R\, x$ *[that is, such that* $(x, x) \notin R$*]*.

Reflexivity, Symmetry, and Transitivity (8/8)

2. R is **not symmetric** $\Leftrightarrow$ there are elements x and y in A such that $x\ R\ y$ but *[that is,* $y \not R\ x$ *such that* $(x, y) \in R$ *but* $(y, x) \notin R$*]*.

3. R is **not transitive** $\Leftrightarrow$ there are elements x, y, and z in A such that $x\ R\ y$ and $y\ R\ z$ but *[that is,* $x \not R\ z$ *such that* $(x, y) \in R$ *and* $(y, z) \in R$ *but* $(x, z) \notin R$*]*.

It follows that you can show that a relation does *not* have one of the properties by finding a counterexample.

Example 8.2.1 – *Properties of Relations on Finite Sets*

Let $A = \{0, 1, 2, 3\}$ and define relations R, S, and T on A as follows:

$R = \{(0, 0), (0, 1), (0, 3), (1, 0), (1, 1), (2, 2), (3, 0), (3, 3)\}$,

$S = \{(0, 0), (0, 2), (0, 3), (2, 3)\}$,
$T = \{(0, 1), (2, 3)\}$.

a. Is R reflexive? symmetric? transitive?

b. Is S reflexive? symmetric? transitive?

c. Is T reflexive? symmetric? transitive?

Example 8.2.1 – *Solution (1/9)*

a. The directed graph of *R* has the appearance shown below.

***R is reflexive*:** There is a loop at each point of the directed graph. This means that each element of *A* is related to itself, so *R* is reflexive.

Example 8.2.1 – *Solution (2/9)* continued

***R is symmetric*:** In each case where there is an arrow going from one point of the graph to a second, there is an arrow going from the second point back to the first.

This means that whenever one element of *A* is related by *R* to a second, then the second is related to the first. Hence *R* is symmetric.

***R is not transitive*:** There is an arrow going from 1 to 0 and an arrow going from 0 to 3, but there is no arrow going from 1 to 3.

Example 8.2.1 – *Solution (3/9)* continued

This means that there are elements of A—0, 1, and 3—such that $1 \, R \, 0$ and $0 \, R \, 3$ but $1 \not R \, 3$. Hence R is not transitive.

b. The directed graph of S has the appearance shown below.

Example 8.2.1 – *Solution (4/9)* continued

***S is not reflexive*:** There is no loop at 1, for example. Thus $(1, 1) \notin S$, and so S is not reflexive.

***S is not symmetric*:** There is an arrow from 0 to 2 but not from 2 to 0. Hence $(0, 2) \in S$ but $(2, 0) \notin S$, and so S is not symmetric.

***S is transitive*:** There are three cases for which there is an arrow going from one point of the graph to a second and from the second point to a third.

Example 8.2.1 – *Solution (5/9)* continued

In particular, there are arrows going from 0 to 2 and from 2 to 3; there are arrows going from 0 to 0 and from 0 to 2; and there are arrows going from 0 to 0 and from 0 to 3.

In each case there is an arrow going from the first point to the third. (Note again that the “first,” “second,” and “third” points need not be distinct.)

This means that whenever $(x, y) \in S$ and $(y, z) \in S$, then $(x, z) \in S$, for every $x, y, z \in \{0, 1, 2, 3\}$, and so S is transitive.

Example 8.2.1 – *Solution (6/9)* continued

c. The directed graph of T has the appearance shown below.

***T is not reflexive*:** There is no loop at 0, for example. Thus $(0, 0) \notin T$, so T is not reflexive.

***T is not symmetric*:** There is an arrow from 0 to 1 but not from 1 to 0. Thus $(0, 1) \in T$ but $(1, 0) \notin T$, and so T is not symmetric.

Example 8.2.1 – *Solution (7/9)* continued

***T is transitive*:** The transitivity condition is vacuously true for T. To see this, observe that the transitivity condition says that

For every $x, y, z \in A$, if $(x, y) \in T$ and $(y, z) \in T$ then $(x, z) \in T$.

The only way for this to be false would be for there to exist elements of A that make the hypothesis true and the conclusion false.

Example 8.2.1 – *Solution (8/9)* continued

That is, there would have to be elements x, y, and z in A such that

$$(x, y) \in T \text{ and } (y, z) \in T \text{ and } (x, z) \notin T.$$

In other words, there would have to be two ordered pairs in T that have the potential to "link up" by having the *second* element of one pair be the *first* element of the other pair.

But the only elements in T are (0, 1) and (2, 3), and these do not have the potential to link up.

Example 8.2.1 – *Solution (9/9)* continued

Hence the hypothesis is never true. It follows that it is impossible for *T not* to be transitive, and thus *T* is transitive.

Properties of Relations on Infinite Sets

Example 8.2.2 – *Properties of Equality*

Define a relation R on $\mathbf{R}$ as follows: For all real numbers x and y,

$$\boxed{x\,R\,y \quad \Leftrightarrow \quad x = y.}$$

a. Is R reflexive? b. Is R symmetric? c. Is R transitive?

Example 8.2.2 – *Solution (1/4)*

a. ***R is reflexive*:** R is reflexive if, and only if, the following statement is true:

For every $x \in \mathbf{R}$, $x \, R \, x$.

Since $x \, R \, x$ just means that $x = x$, this is the same as saying

For every $x \in \mathbf{R}$, $x = x$.

But this statement is certainly true; every real number is equal to itself.

Example 8.2.2 – *Solution (2/4)* continued

b. ***R is symmetric*:** R is symmetric if, and only if, the following statement is true:

For every $x, y \in \mathbf{R}$, **if** $x\,R\,y$ then $y\,R\,x$.

By definition of R, $x\,R\,y$ means that $x = y$ and $y\,R\,x$ means that $y = x$. Hence R is symmetric if, and only if,

For every $x, y \in \mathbf{R}$, **if** $x = y$ then $y = x$.

But this statement is certainly true; if one number is equal to a second, then the second is equal to the first.

Example 8.2.2 – *Solution (3/4)* continued

c. ***R is transitive*:** R is transitive if, and only if, the following statement is true:

For every $x, y, z \in \mathbf{R}$, **if** $x\,R\,y$ and $y\,R\,z$ then $x\,R\,z$.

By definition of R, $x\,R\,y$ means that $x = y$, $y\,R\,z$ means that $y = z$, and $x\,R\,z$ means that $x = z$.

Hence R is transitive if, and only if, the following statement is true:

For every $x, y, z \in \mathbf{R}$, **if** $x = y$ and $y = z$ then $x = z$.

Example 8.2.2 – *Solution (4/4)* continued

But this statement is certainly true: If one real number equals a second and the second equals a third, then the first equals the third.

Example 8.2.3 – *Properties of "Less Than"*

Define a relation R on $\mathbf{R}$ as follows: For all real numbers x and y,

$$\boxed{x\,R\,y \quad \Leftrightarrow \quad x < y.}$$

a. Is R reflexive? b. Is R symmetric? c. Is R transitive?

Example 8.2.3 – *Solution (1/2)*

a. ***R is not reflexive*:** R is reflexive if, and only if, $\forall x \in \mathbf{R}$, $x\ R\ x$. By definition of R, this means that $\forall x \in \mathbf{R}$, $x < x$. But this is false: $\exists x \in \mathbf{R}$ such that $x \not< x$.

As a counterexample, let $x = 0$ and note that $0 \not< 0$. Hence R is not reflexive.

b. ***R is not symmetric*:** R is symmetric if, and only if, $\forall x$, $y \in \mathbf{R}$, if $x\ R\ y$ then $y\ R\ x$. By definition of R, this means that $\forall x, y \in \mathbf{R}$, if $x < y$ then $y < x$.

Example 8.2.3 – *Solution (2/2)* continued

But this is false: $\exists x, y \in \mathbf{R}$ such that $x < y$ and $y \not< x$. As a counterexample, let $x = 0$ and $y = 1$ and note that $0 < 1$ but $1 \not< 0$. Hence R is not symmetric.

c. ***R is transitive*:** R is transitive if, and only if, $\forall x, y, z \in \mathbf{R}$, if $x\,R\,y$ and $y\,R\,z$ then $x\,R\,z$. By definition of R, this means that $\forall x, y, z \in \mathbf{R}$, if $x < y$ and $y < z$, then $x < z$. But this statement is true by the transitive law of order for real numbers. Hence R is transitive.

Example 8.2.4 – *Properties of Congruence Modulo 3*

Define a relation T on $\mathbf{Z}$ (the set of all integers) as follows: For all integers m and n,

$$\boxed{m\,T\,n \quad \Leftrightarrow \quad 3 \mid (m-n).}$$

This relation is called **congruence modulo 3**.

a. Is T reflexive? b. Is T symmetric? c. Is T transitive?

Example 8.2.4 – *Solution (1/6)*

a. ***T is reflexive*:** To show that T is reflexive, it is necessary to show that For every $m \in \mathbf{Z}$, $m\ T\ m$.

By definition of T, this means that

$$\text{For every } m \in \mathbf{Z}, \quad 3 \mid (m - m),$$

which is true because $m - m = 0$ and $3 \mid 0$ (since $0 = 3 \cdot 0$). Hence T is reflexive.

b. ***T is symmetric*:** To show that T is symmetric, it is necessary to show that

For every $m, n \in \mathbf{Z}$, **if** $m\ T\ n$ then $n\ T\ m$.

Example 8.2.4 – *Solution (2/6)* continued

By definition of *T* this means that

$$\text{For every } m, n \in \mathbf{Z}, \quad \textbf{if } 3 \mid (m-n) \text{ then } 3 \mid (n-m).$$

Is this true? Suppose *m* and *n* are particular but arbitrarily chosen integers such that $3 \mid (m-n)$.

Must it follow that $3 \mid (n-m)$? *[In other words, can we find an integer so that n − m = 3 ⋅ (that integer)?]* By definition of "divides," since

$$3 \mid (m-n),$$

then $m - n = 3k$ for some integer k.

Example 8.2.4 – *Solution (3/6)* continued

The crucial observation is that $n - m = -(m - n)$. Hence, you can multiply both sides of this equation by -1 to obtain

$$-(m - n) = -3k,$$

which is equivalent to

$$n - m = 3(-k).$$

[Thus we have found an integer, $-k$, so that $n - m = 3 \cdot$ (that integer).]

Example 8.2.4 – *Solution (4/6)* continued

Since $-k$ is an integer, this equation shows that

$$3 \mid (n - m).$$

It follows that T is symmetric.

c. ***T is transitive*:** To show that T is transitive, it is necessary to show that

For every $m, n, p \in \mathbf{Z}$, **if** $m\ T\ n$ and $n\ T\ p$ then $m\ T\ p$.

By For every $m, n \in \mathbf{Z}$, **if** $3 \mid (m - n)$ and $3 \mid (n - p)$ then $3 \mid (m - p)$.

Example 8.2.4 – *Solution (5/6)*

continued

Is this true? Suppose *m*, *n*, and *p* are particular but arbitrarily chosen integers such that $3 \mid (m-n)$ and $3 \mid (n-p)$.

Must it follow that $3 \mid (m-p)$? *[In other words, can we find an integer so that m − p = 3 · (that integer)?]* By definition of "divides," since

$$3 \mid (m-n) \quad \text{and} \quad 3 \mid (n-p),$$

then

$$m - n = 3r \qquad \text{for some integer } r,$$

and

$$n - p = 3s \qquad \text{for some integer } s.$$

Example 8.2.4 – *Solution (6/6)* continued

The crucial observation is that $(m - n) + (n - p) = m - p$. Add these two equations together to obtain

$$(m - n) + (n - p) = 3r + 3s,$$

which is equivalent to

$$m - p = 3(r + s).$$

[Thus we have found an integer so that $m - p = 3 \cdot$ (that integer).] Since r and s are integers, $r + s$ is an integer. So this equation shows that

$$3 \mid (m - p).$$

It follows that T is transitive.

The Transitive Closure of a Relation

The Transitive Closure of a Relation (1/1)

The relation obtained by adding the least number of ordered pairs to ensure transitivity is called the *transitive closure* of the relation. More precisely, the transitive closure of a relation is the smallest transitive relation that contains the relation.

Definition

Let A be a set and R a relation on A. The **transitive closure** of R is the relation R^t on A that satisfies the following three properties:

1. R^t is transitive.
2. $R \subseteq R^t$.
3. If S is any other transitive relation that contains R, then $R^t \subseteq S$.

Example 8.2.5 – *Transitive Closure of a Relation*

Let $A = \{0, 1, 2, 3\}$ and consider the relation R defined on A as follows:

$$R = \{(0, 1), (1, 2), (2, 3)\}.$$

Find the transitive closure of R.

Example 8.2.5 – *Solution (1/2)*

Every ordered pair in *R* is in R^t, so

$$\{(0, 1), (1, 2), (2, 3)\} \subseteq R^t.$$

Thus the directed graph of *R* contains the arrows shown below.

Since there are arrows going from 0 to 1 and from 1 to 2, R^t must have an arrow going from 0 to 2.

Example 8.2.5 – *Solution (2/2)*

continued

Hence $(0, 2) \in R^t$. Then $(0, 2) \in R^t$ and $(2, 3) \in R^t$, so since R^t is transitive, $(0, 3) \in R^t$. Also, since $(1, 2) \in R^t$ and $(2, 3) \in R^t$, then $(1, 3) \in R^t$. Thus R^t contains at least the following ordered pairs:

$$\{(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)\}.$$

But this relation *is* transitive; hence it equals R^t. The directed graph of R^t is shown below.

CHAPTER 8

PROPERTIES OF RELATIONS

Copyright © Cengage Learning. All rights reserved.

8.3 Equivalence Relations

Copyright © Cengage Learning. All rights reserved.

The Relation Induced by a Partition

The Relation Induced by a Partition (1/3)

A **partition** of a set A is a finite or infinite collection of nonempty, mutually disjoint subsets whose union is A. The diagram of Figure 8.3.1 illustrates a partition of a set A by subsets $A_1, A_2, \ldots, A_6$.

$A_i \cap A_j = \varnothing$, whenever $i \neq j$
$A_1 \cup A_2 \cup \cdots \cup A_6 = A$

A Partition of a Set

Figure 8.3.1

The Relation Induced by a Partition (2/3)

Definition

Given a partition of a set A, the **relation induced by the partition**, R, is defined on A as follows: For every $x, y \in A$,

$$x \, R \, y \quad \Leftrightarrow \quad \text{there is a subset } A_i \text{ of the partition such that both } x \text{ and } y \text{ are in } A_i.$$

Example 8.3.1 – *Relation Induced by a Partition*

Let $A = \{0, 1, 2, 3, 4\}$ and consider the following partition of A:

$$\{0, 3, 4\}, \{1\}, \{2\}.$$

Find the relation R induced by this partition.

Example 8.3.1 – *Solution (1/3)*

Since {0, 3, 4} is a subset of the partition,

0 R 3 because both 0 and 3 are in {0, 3, 4}

3 R 0 because both 3 and 0 are in {0, 3, 4}

0 R 4 because both 0 and 4 are in {0, 3, 4}

4 R 0 because both 4 and 0 are in {0, 3, 4}

3 R 4 because both 3 and 4 are in {0, 3, 4}

and

4 R 3 because both 4 and 3 are in {0, 3, 4}.

Example 8.3.1 – *Solution (2/3)* continued

Also,

$0\ R\ 0$ because both 0 and 0 are in $\{0, 3, 4\}$

$3\ R\ 3$ because both 3 and 3 are in $\{0, 3, 4\}$

and

$4\ R\ 4$ because both 4 and 4 are in $\{0, 3, 4\}$.

Example 8.3.1 – *Solution (3/3)* continued

Since {1} is a subset of the partition,

1 *R* 1 because both 1 and 1 are in {1},

and since {2} is a subset of the partition,

2 *R* 2 because both 2 and 2 are in {2}.

Hence

$R = \{(0, 0), (0, 3), (0, 4), (1, 1), (2, 2), (3, 0), (3, 3), (3, 4), (4, 0), (4, 3), (4, 4)\}.$

The Relation Induced by a Partition (3/3)

Theorem 8.3.1

Let A be a set with a partition and let R be the relation induced by the partition. Then R is reflexive, symmetric, and transitive.

Definition of an Equivalence Relation

Definition of an Equivalence Relation (1/1)

A relation on a set that satisfies the three properties of reflexivity, symmetry, and transitivity is called an *equivalence relation*.

Definition

Let A be a set and R a relation on A. R is an **equivalence relation** if, and only if, R is reflexive, symmetric, and transitive.

Example 8.3.2 – *An Equivalence Relation on a Set of Subsets*

Let X be the set of all nonempty subsets of {1, 2, 3}. Then

$$X = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

Define a relation **R** on X as follows: For every A and B in X,

A **R** $B \Leftrightarrow$ the least element of A equals the least element of B.

Prove that **R** is an equivalence relation on X.

Example 8.3.2 – *Solution (1/2)*

R *is reflexive*: Suppose A is a nonempty subset of {1, 2, 3}. *[We must show that* A **R** A*.]* It is true to say that the least element of A equals the least element of A. Thus, by definition of **R**, A **R** A.

R *is symmetric*: Suppose A and B are nonempty subsets of {1, 2, 3} and A **R** B. *[We must show that* B **R** A*.]* Since A **R** B, the least element of A equals the least element of B. But this implies that the least element of B equals the least element of A, and so, by definition of **R**, B **R** A.

Example 8.3.2 – *Solution (2/2)* continued

R *is transitive*: Suppose A, B, and C are nonempty subsets of {1, 2, 3}, A **R** B, and B R C. *[We must show that* A **R** C*.]* Since A **R** B, the least element of A equals the least element of B and since B **R** C, the least element of B equals the least element of C.

Thus the least element of A equals the least element of C, and so, by definition of **R**, A **R** C.

Equivalence Classes of an Equivalence Relation

Equivalence Classes of an Equivalence Relation (1/5)

Suppose there is an equivalence relation on a certain set. If *a* is any particular element of the set, then one can ask, "What is the subset of all elements that are related to *a*?" This subset is called the *equivalence class* of *a*.

Definition

Suppose A is a set and R is an equivalence relation on A. For each element a in A, the **equivalence class of a**, denoted $[a]$ and called the **class of a** for short, is the set of all elements x in A such that x is related to a by R.

In symbols:

$$[a] = \{x \in A \mid x \, R \, a\}$$

Equivalence Classes of an Equivalence Relation (2/5)

The procedural version of this definition is

$$\text{for every } x \in A, \quad x \in [a] \quad \Leftrightarrow \quad x\,R\,a.$$

The notation $[a]_R$ may be used to denote the equivalence class of *a* for the relation *R*.

Example 8.3.5 – *Equivalence Classes of a Relation Given as a Set of Ordered Pairs (1/2)*

Let $A = \{0, 1, 2, 3, 4\}$ and define a relation R on A as follows:

$$R = \{(0, 0), (0, 4), (1, 1), (1, 3), (2, 2), (3, 1), (3, 3), (4, 0), (4, 4)\}.$$

Example 8.3.5 – *Equivalence Classes of a Relation Given as a Set of Ordered Pairs (2/2)* continued

The directed graph for R is as shown below. As can be seen by inspection, R is an equivalence relation on A. Find the distinct equivalence classes of R.

Example 8.3.5 – *Solution*

First find the equivalence class of every element of *A*.

$$[0] = \{x \in A \mid x\,R\,0\} = \{0, 4\}$$
$$[1] = \{x \in A \mid x\,R\,1\} = \{1, 3\}$$
$$[2] = \{x \in A \mid x\,R\,2\} = \{2\}$$
$$[3] = \{x \in A \mid x\,R\,3\} = \{1, 3\}$$
$$[4] = \{x \in A \mid x\,R\,4\} = \{0, 4\}$$

Note that [0] = [4] and [1] = [3]. Thus the *distinct* equivalence classes of the relation are {0, 4}, {1, 3}, and {2}.

Example 8.3.8 – *Equivalence Classes of the Identity Relation*

Let A be any set and define a relation R on A as follows: For every x and y in A,

$$\boxed{x\,R\,y \quad \Leftrightarrow \quad x = y.}$$

Then R is an equivalence relation. Describe the distinct equivalence classes of R.

Example 8.3.8 – *Solution (1/2)*

Given any *a* in *A*, the class of *a* is

$$[a] = \{x \in A \mid x\, R\, a\}.$$

Now by definition of *R*, *a R x* if, and only if, *a* = *x*. So

$$[a] = \{x \in A \mid x = a\}$$

= [*a*] since the only element of *A* that equals *a* is *a*.

Example 8.3.8 – *Solution (2/2)*

continued

Hence, given any a in A,

$$[a] = \{a\},$$

and if $x \neq a$ then $\{x\} \neq \{a\}$.

Consequently, all the classes of all the elements of A are distinct, and the distinct equivalence classes of R are all the single-element subsets of A.

Equivalence Classes of an Equivalence Relation (3/5)

The first lemma says that if two elements of A are related by an equivalence relation R, then their equivalence classes are the same.

Lemma 8.3.2

Suppose A is a set, R is an equivalence relation on A, and a and b are elements of A. If $a \; R \; b$, then $[a] = [b]$.

Equivalence Classes of an Equivalence Relation (4/5)

The second lemma says that any two equivalence classes of an equivalence relation are either mutually disjoint or identical.

Lemma 8.3.3

If A is a set, R is an equivalence relation on A, and a and b are elements of A, then

$$\text{either} \quad [a] \cap [b] = \varnothing \quad \text{or} \quad [a] = [b].$$

Equivalence Classes of an Equivalence Relation (5/5)

Theorem 8.3.4 The Partition Induced by an Equivalence Relation

If A is a set and R is an equivalence relation on A, then the distinct equivalence classes of R form a partition of A; that is, the union of the equivalence classes is all of A, and the intersection of any two distinct classes is empty.

Congruence Modulo n

Example 8.3.10 – *Equivalence Classes of Congruence Modulo 3*

Let R be the relation of congruence modulo 3 on the set $\mathbf{Z}$ of all integers. That is, for all integers m and n,

$$\boxed{m\,R\,n \quad \Leftrightarrow \quad 3 \mid (m-n).}$$

Describe the distinct equivalence classes of R.

Example 8.3.10 – *Solution (1/4)*

For each integer *a*,

$$\begin{aligned}[a] &= \{x \in \mathbf{Z} \mid x\, R\, a\} \\ &= \{x \in \mathbf{Z} \mid 3 \mid (x - a)\} \\ &= \{x \in \mathbf{Z} \mid x - a = 3k, \text{ for some integer } k\}.\end{aligned}$$

Therefore,

$$[a] = \{x \in Z \mid x = 3k + a, \text{ for some integer } k\}.$$

Example 8.3.10 – *Solution (2/4)* continued

In particular,

$$
\begin{aligned}
[0] &= \{x \in \mathbf{Z} \mid x = 3k + 0, \text{ for some integer } k\} \\
&= \{x \in \mathbf{Z} \mid x = 3k, \text{ for some integer } k\} \\
&= \{\ldots -9, -6, -3, 0, 3, 6, 9, \ldots\}, \\
[1] &= \{x \in \mathbf{Z} \mid x = 3k + 1, \text{ for some integer } k\} \\
&= \{\ldots -8, -5, -2, 1, 4, 7, 10, \ldots\}, \\
[2] &= \{x \in \mathbf{Z} \mid x = 3k + 2, \text{ for some integer } k\} \\
&= \{\ldots -7, -4, -1, 2, 5, 8, 11, \ldots\}.
\end{aligned}
$$

Example 8.3.10 – *Solution (3/4)* continued

Now since $3 \; R \; 0$, then by Lemma 8.3.2,

$$[3] = [0].$$

More generally, by the same reasoning,

$$[0] = [3] = [-3] = [6] = [-6] = \ldots \text{ , and so on.}$$

Similarly,

$$[1] = [4] = [-2] = [7] = [-5] = \ldots \text{ , and so on.}$$

And

$$[2] = [5] = [-1] = [8] = [-4] = \ldots \text{ , and so on.}$$

Example 8.3.10 – *Solution (4/4)* continued

Notice that every integer is in class [0], [1], or [2]. Hence the distinct equivalence classes are

$$\{x \in \mathbf{Z} \mid x = 3k, \text{ for some integer } k\},$$

$$\{x \in \mathbf{Z} \mid x = 3k + 1, \text{ for some integer } k\}, \quad \text{and}$$

$$\{x \in \mathbf{Z} \mid x = 3k + 2, \text{ for some integer } k\}.$$

In words, the three classes of congruence modulo 3 are (1) the set of all integers that are divisible by 3, (2) the set of all integers that leave a remainder of 1 when divided by 3, and (3) the set of all integers that leave a remainder of 2 when divided by 3.

Congruence Modulo n (1/2)

Definition

Suppose R is an equivalence relation on a set A and S is an equivalence class of R. A **representative** of the class S is any element a such that $[a] = S$.

Congruence Modulo n (2/2)

The following notation is used frequently when referring to congruence relations. It was introduced by Carl Friedrich Gauss in the first chapter of his book *Disquisitiones Arithmeticae*. This work, which was published when Gauss was only 24, laid the foundation for modern number theory.

Definition

Let m and n be integers and let d be a positive integer. We say that **m is congruent to n modulo d** and write

$$\boldsymbol{m \equiv n \pmod{d}}$$

if, and only if, $d \mid (m - n)$.

Symbolically: $m \equiv n \pmod{d} \quad \Leftrightarrow \quad d \mid (m - n)$.

Example 8.3.11 – *Evaluating Congruences*

Determine which of the following congruences are true and which are false.

a. $12 \equiv 7 \pmod{5}$ b. $6 \equiv -8 \pmod{4}$ c. $3 \equiv 3 \pmod{7}$

Example 8.3.11 – *Solution*

a. True. $12 - 7 = 5 = 5 \cdot 1$. Hence $5 \mid (12 - 7)$, and so $12 \equiv 7 \pmod 5$.

b. False. $6 - (-8) = 14$, and $4 \nmid 14$ because $14 \neq 4 \cdot k$ for any integer k. Consequently, $6 \not\equiv -8 \pmod 4$.

c. True. $3 - 3 = 0 = 7 \cdot 0$. Hence $7 \mid (3 - 3)$, and so $3 \equiv 3 \pmod 7$.

A Definition for Rational Numbers

A Definition for Rational Numbers (1/2)

For a moment, forget what you know about fractional arithmetic and look at the numbers

$$\frac{1}{3} \quad \text{and} \quad \frac{2}{6}$$

as *symbols*. Considered as symbolic expressions, these *appear* quite different. In fact, if they were written as ordered pairs (1, 3) and (2, 6) they would *be* different.

A Definition for Rational Numbers (2/2)

The fact that we regard them as “the same” is a specific instance of our general agreement to regard any two numbers

$$\frac{a}{b} \quad \text{and} \quad \frac{c}{d}$$

as equal provided the *cross products* are equal; in other words, if, and only if, $ad = bc$.

Example 8.3.12 – *Rational Numbers Are Really Equivalence Classes (1/2)*

Let A be the set of all ordered pairs of integers for which the second element of the pair is nonzero. Symbolically:

$$A = \mathbf{Z} \times (\mathbf{Z} - \{0\}).$$

Define a relation R on A as follows: For all pairs (a, b) and (c, d) in A,

$$\boxed{(a, b)\, R\, (c, d) \quad \Leftrightarrow \quad ad = bc.}$$

Example 8.3.12 – *Rational Numbers Are Really Equivalence Classes (2/2)*

continued

The fact is that R is an equivalence relation.

a. Prove that R is transitive.

b. Describe the distinct equivalence classes of R.

Example 8.3.12 – *Solution (1/4)*

a. By definition of R,

$$(1)\ ad = bc \quad \text{and} \quad (2)\ cf = de.$$

Since the second elements of all ordered pairs in A are nonzero, $b \neq 0$, $d \neq 0$, and $f \neq 0$. Multiply both sides of equation (1) by f and both sides of equation (2) by b to obtain

$$(1')\ adf = bcf \quad \text{and} \quad (2')\ bcf = bde.$$

Example 8.3.12 – *Solution (2/4)* continued

Because both equal bcf,

$$adf = bde,$$

and, since $d \neq 0$, it follows from the cancellation law for multiplication that

$$af = be.$$

Hence, by definition of R, $(a, b)\ R\ (e, f)$.

Example 8.3.12 – *Solution (3/4)* continued

b. There is one equivalence class for each distinct rational number. Each equivalence class consists of all ordered pairs (a, b) that, if written as fractions a/b, would equal each other.

The reason is that the condition for two rational numbers to be equal is the same as the condition for two ordered pairs to be related.

Example 8.3.12 – *Solution (4/4)* continued

For instance, the class of (1, 2) is

[(1, 2)] = {(1, 2), (−1, −2), (2, 4), (−2, −4), (3, 6), (−3, −6), ... }

since $\frac{1}{2} = \frac{-1}{-2} = \frac{2}{4} = \frac{-2}{-4} = \frac{3}{6} = \frac{-3}{-6}$ and so forth.

CHAPTER 8

PROPERTIES OF RELATIONS

Copyright © Cengage Learning. All rights reserved.

8.4 Modular Arithmetic with Applications to Cryptography

Copyright © Cengage Learning. All rights reserved.

Modular Arithmetic with Applications to Cryptography (1/4)

Cryptography is the study of methods for sending secret messages. It involves **encryption**, in which a message, called **plaintext**, is converted into a form, called **ciphertext**, that is sent over a channel possibly open to view by outside parties.

The receiver of the ciphertext uses **decryption** to convert the ciphertext back into plaintext.

Modular Arithmetic with Applications to Cryptography (2/4)

Many systems for sending secret messages require both the sender and the receiver to know both the encryption and the decryption procedures.

For instance, an encryption system once used by Julius Caesar, and now called the **Caesar cipher**, encrypts messages by changing each letter of the alphabet to the one three places farther along, with X wrapping around to A, Y to B, and Z to C.

Modular Arithmetic with Applications to Cryptography (3/4)

In other words, say each letter of the alphabet is coded by its position relative to the others—so that

A = 01, B = 02, … , Z = 26.

If the numerical version of the plaintext for a letter is denoted *M* and the numeric version of the ciphertext is denoted *C*, then

$$C = (M + 3) \; mod \; 26.$$

Modular Arithmetic with Applications to Cryptography (4/4)

The receiver of such a message can easily decrypt it by using the formula

$$M = (C - 3) \; mod \; 26.$$

For reference, here are the letters of the alphabet, together with their numeric equivalents:

A 01	B 02	C 03	D 04	E 05	F 06	G 07	H 08	I 09	J 10	K 11	L 12	M 13
N 14	O 15	P 16	Q 17	R 18	S 19	T 20	U 21	V 22	W 23	X 24	Y 25	Z 26

Example 8.4.1 – *Encrypting and Decrypting with the Caesar Cipher*

a. Use the Caesar cipher to encrypt the message HOW ARE YOU.

b. Use the Caesar cipher to decrypt the message L DP ILQH.

Example 8.4.1 – *Solution (1/2)*

a. First, translate the letters of HOW ARE YOU into their numeric equivalents:

08 15 23 01 18 05 25 15 21.

Next, encrypt the message by adding 3 to each number. The result is

11 18 26 04 21 08 02 18 24.

Finally, substitute the letters that correspond to these numbers. The encrypted message becomes

KRZ DUH BRX.

Example 8.4.1 – *Solution (2/2)* continued

b. First, translate the letters of L DP ILQH into their numeric equivalents:

12 04 16 09 12 17 08.

Next, decrypt the message by subtracting 3 from each number:

09 01 13 06 09 14 05.

Then, translate back into letters to obtain the original message: I AM FINE.

Properties of Congruence Modulo *n*

Properties of Congruence Modulo *n (1/4)*

Theorem 8.4.1 Modular Equivalences

Let a, b, and n be any integers and suppose $n > 1$. The following statements are all equivalent:

1. $n \mid (a - b)$
2. $a \equiv b \pmod{n}$
3. $a = b + kn$ for some integer k
4. a and b have the same (nonnegative) remainder when divided by n
5. $a \bmod n = b \bmod n$

Properties of Congruence Modulo *n (2/4)*

Another consequence of the quotient-remainder theorem is this: When an integer a is divided by a positive integer n, a unique quotient q and remainder r are obtained with the property that $a = nq + r$ and $0 \leq r < n$.

Because there are exactly n integers that satisfy the inequality $0 \leq r < n$ (the numbers from 0 through $n - 1$), there are exactly n possible remainders that can occur.

Properties of Congruence Modulo *n (3/4)*

These are called the *least nonnegative residues modulo n* or simply the *residues modulo n*.

Definition

Given integers a and n with $n > 1$, **the residue of a modulo n** is $a \bmod n$, the nonnegative remainder obtained when a is divided by n. The numbers $0, 1, 2, \ldots, n-1$ are called a **complete set of residues modulo n**. To **reduce a number modulo n** means to set it equal to its residue modulo n. If a modulus $n > 1$ is fixed throughout a discussion and an integer a is given, the words "modulo n" are often dropped and we simply speak of **the residue of a**.

Properties of Congruence Modulo *n* (4/4)

Theorem 8.4.2 Congruence Modulo *n* Is an Equivalence Relation

If n is any integer with $n > 1$, congruence modulo n is an equivalence relation on the set of all integers. The distinct equivalence classes of the relation are the sets [0], [1], [2], . . . , $[n-1]$, where for each $a = 0, 1, 2, \ldots, n-1$,

$$[a] = \{m \in Z \mid m \equiv a \pmod{n}\},$$

or, equivalently,

$$[a] = \{m \in Z \mid m = a + kn \text{ for some integer } k\}.$$

Modular Arithmetic

Modular Arithmetic (1/5)

A fundamental fact about congruence modulo *n* is that if you first perform an addition, subtraction, or multiplication on integers and then reduce the result modulo *n*, you will obtain the same answer as if you had first reduced each of the numbers modulo *n*, performed the operation, and then reduced the result modulo *n*. For instance, instead of computing

$$(5 \cdot 8) = 40 \equiv 1 \pmod 3$$

you will obtain the same answer if you compute

$$(5 \; mod \; 3)\,(8 \; mod \; 3) = 2 \cdot 2 = 4 \equiv 1 \pmod 3.$$

Modular Arithmetic (2/5)

Theorem 8.4.3 Modular Arithmetic

Let a, b, c, d, and n be integers with $n > 1$, and suppose

$$a \equiv c \pmod{n} \quad \text{and} \quad b \equiv d \pmod{n}.$$

Then

1. $(a + b) \equiv (c + d) \pmod{n}$
2. $(a - b) \equiv (c - d) \pmod{n}$
3. $ab \equiv cd \pmod{n}$
4. $a^m \equiv c^m \pmod{n}$ for every positive integer m.

Example 8.4.2 – *Getting Started with Modular Arithmetic*

The most practical use of modular arithmetic is to reduce computations involving large integers to computations involving smaller ones. For instance, note that $55 \equiv 3 \pmod 4$ because $55 - 3 = 52$, which is divisible by 4, and $26 \equiv 2 \pmod 4$ because $26 - 2 = 24$, which is also divisible by 4. Verify the following statements.

a. $55 + 26 \equiv (3 + 2) \pmod 4$ b. $55 - 26 \equiv (3 - 2) \pmod 4$

c. $55 \cdot 26 \equiv (3 \cdot 2) \pmod 4$ d. $55^2 \equiv 3^2 \pmod 4$

Example 8.4.2 – *Solution (1/2)*

a. Compute 55 + 26 = 81 and 3 + 2 = 5. By definition of congruence modulo *n*, to show that 81 ≡ 5 (mod 4), you need to show that $4 \mid (81 - 5)$. But this is true because 81 − 5 = 76, and $4 \mid 76$ since 76 = 4 · 19.

b. Compute 55 − 26 = 29 and 3 − 2 = 1. By definition of congruence modulo *n*, to show that 29 ≡ 1 (mod 4), you need to show that $4 \mid (29 - 1)$. But this is true because 29 − 1 = 28, and $4 \mid 28$ since 28 = 4 · 7.

Example 8.4.2 – *Solution (2/2)*

continued

c. Compute 55 · 26 = 1430 and 3 · 2 = 6. By definition of congruence modulo *n*, to show that 1430 ≡ 6 (mod 4), you need to show that $4 \mid (1430 - 6)$. But this is true because 1430 − 6 = 1424, and $4 \mid 1424$ since 1424 = 4 · 356.

d. Compute $55^2 = 3025$ and $3^2 = 9$. By definition of congruence modulo *n*, to show that 3025 ≡ 9 (mod 4), you need to show that $4 \mid (3025 - 9)$. But this is true because 3025 − 9 = 3016, and $4 \mid 3016$ since 3016 = 4 · 754.

Modular Arithmetic (3/5)

Corollary 8.4.4

Let a, b, and n be integers with $n > 1$. Then

$$ab \equiv [(a \bmod n)(b \bmod n)] \pmod{n},$$

or, equivalently,

$$ab \bmod n = [(a \bmod n)(b \bmod n)] \bmod n.$$

In particular, if m is a positive integer, then

$$a^m \equiv [(a \bmod n)^m] \pmod{n}.$$

Example 8.4.3 – *Computing a Product Modulo n*

Note that $55 \equiv 3 \pmod 4$ and $26 \equiv 2 \pmod 4$. Because both 3 and 2 are less than 4, each of these numbers is a least nonnegative residue modulo 4. Therefore, $55 \; mod \; 4 = 3$ and $26 \; mod \; 4 = 2$. Use the notation of Corollary 8.4.4 to find the residue of $55 \cdot 26$ modulo 4.

Example 8.4.3 – *Solution (1/2)*

We know that to use a calculator to compute remainders, you can use the formula $n \bmod d = n - d \cdot \lfloor n/d \rfloor$. If you are using a hand calculator with an "integer part" feature and both n and d are positive, then $\lfloor n/d \rfloor$ is the integer part of the division of n by d.

When you divide a positive integer n by a positive integer d with a more basic calculator, you can see $\lfloor n/d \rfloor$ on the calculator display by simply ignoring the digits that follow the decimal point.

Example 8.4.3 – *Solution (2/2)*

continued

By Corollary 8.4.4,

$(55 \cdot 26)\ mod\ 4 = \{(55\ mod\ 4)(26\ mod\ 4)\}\ mod\ 4$

$\equiv (3 \cdot 2)\ mod\ 4$ because 55 mod 4 = 3 and 26 mod 4 = 2

$\equiv 6\ mod\ 4$

$\equiv 2$ because $4 \mid (6-2)$ and 2 < 4.

Modular Arithmetic (4/5)

When modular arithmetic is performed with very large numbers, as is the case for RSA crytography, computations are facilitated by using two properties of exponents. The first is

$$x^{2a} = (x^a)^2 \quad \text{for all real numbers } x \text{ and } a \text{ with } x \geq 0. \qquad 8.4.1$$

Thus, for instance, if *x* is any positive real number, then

$$\begin{aligned} x^4 \; mod \; n &= (x^2)^2 \; mod \; n && \text{because } (x^2)^2 = x^4 \\ &= (x^2 \; mod \; n)^2 \; mod \; n && \text{by Corollary 8.4.4.} \end{aligned}$$

Modular Arithmetic (5/5)

A second useful property of exponents is

$$x^{a+b} = x^a x^b \quad \text{for all real numbers } x, a, \text{ and } b \text{ with } x \geq 0.$$

8.4.2

For instance, because 7 = 4 + 2 + 1,

$$x^7 = x^4 x^2 x^1.$$

Thus, by Corollary 8.4.4,

$$x^7 \; mod \; n = \{(x^4 \; mod \; n)(x^2 \; mod \; n)(x^1 \; mod \; n)\} \; mod \; n.$$

Example 8.4.4 – *Computing* a^k *mod n When k Is a Power of 2*

Find $144^4 \; mod \; 713$.

Example 8.4.4 – *Solution*

Use property (8.4.1) to write $144^4 = (144^2)^2$. Then

$$
\begin{aligned}
144^4 \; mod \; 713 &= (144^2)^2 \; mod \; 713 \\
&= (144^2 \; mod \; 713)^2 \; mod \; 713 \\
&= (20736 \; mod \; 713)^2 \; mod \; 713 && \text{because } 144^2 = 20736 \\
&= 59^2 \; mod \; 713 && \text{because } 20736 \; mod \; 713 = 59 \\
&= 3481 \; mod \; 713 && \text{because } 59^2 = 3481 \\
&= 629 && \text{because } 3481 \; mod \; 713 = 629.
\end{aligned}
$$

Example 8.4.5 – *Computing a^k mod n When k Is Not a Power of 2*

Find $12^{43}\ mod\ 713.$

Example 8.4.5 – *Solution (1/2)*

First write the exponent as a sum of powers of 2:

$$43 = 2^5 + 2^3 + 2 + 1 = 32 + 8 + 2 + 1.$$

Next compute 12^{2^k} for k = 0, 1, 2, 3, 4, and 5.

$$12 \; mod \; 713 = 12$$

$$12^2 \; mod \; 713 = 144$$

$$12^4 \; mod \; 713 = 144^2 mod \; 713 = 59$$ by Example 8.4.4

$$12^8 \; mod \; 713 = 59^2 \; mod \; 713 = 629$$ by Example 8.4.4

$$12^{16} \; mod \; 713 = 629^2 \; mod \; 713 = 639$$ by the method of Example 8.4.4

$$12^{32} \; mod \; 713 = 639^2 \; mod \; 713 = 485$$ by the method of Example 8.4.4.

Example 8.4.5 – *Solution (2/2)*

continued

By property (8.4.2),

$$12^{43} = 12^{32+8+2+1} = 12^{32} \cdot 12^{8} \cdot 12^{2} \cdot 12^{1}.$$

Thus, by Corollary 8.4.4,

$$12^{43} \; mod \; 713$$

$$= \{(12^{32} \; mod \; 713) \cdot (12^{8} \; mod \; 713) \cdot (12^{2} \; mod \; 713) \cdot (12 \; mod \; 713)\} \; mod \; 713.$$

By substitution,

$$\begin{aligned} 12^{43} \; mod \; 713 &= (485 \cdot 629 \cdot 144 \cdot 12) \; mod \; 713 \\ &= 527152320 \; mod \; 713 \\ &= 48. \end{aligned}$$

Extending the Euclidean Algorithm

Extending the Euclidean Algorithm (1/1)

An extended version of the Euclidean algorithm can be used to find a concrete expression for the greatest common divisor of integers *a* and *b*.

Definition

An integer d is said to be a **linear combination of integers** a and b if, and only if, there exist integers s and t such that $as + bt = d$.

Theorem 8.4.5 Writing a Greatest Common Divisor as a Linear Combination

For all integers a and b, not both zero, if $d = \gcd(a, b)$, then there exist integers s and t such that $as + bt = d$.

Example 8.4.6 – *Expressing a Greatest Common Divisor as a Linear Combination*

We know how to use the Euclidean algorithm to find that the greatest common divisor of 330 and 156 is 6. Use the results of these calculations to express gcd(330, 156) as a linear combination of 330 and 156.

Example 8.4.6 – *Solution (1/3)*

The first four steps were obtained by successive applications of the quotient-remainder theorem. The fifth step shows how to find the coefficients of the linear combination by substituting back through the results of the previous steps.

Step 1: $330 = 156 \cdot 2 + 18$, which implies that

$$18 = 330 - 156 \cdot 2.$$

Step 2: $156 = 18 \cdot 8 + 12$, which implies that

$$12 = 156 - 18 \cdot 8.$$

Step 3: $18 = 12 \cdot 1 + 6$, which implies that $6 = 18 - 12 \cdot 1$.

Step 4: $12 = 6 \cdot 2 + 0$, which implies that $\gcd(330, 156) = 6$.

Example 8.4.6 – *Solution (2/3)* continued

Step 5: By substituting back through steps 3 to 1:

$$6 = 18 - 12 \cdot 1$$ from step 3

$$= 18 - (156 - 8 \cdot 18) \cdot 1$$ by substitution from step 2

$$= 9 \cdot 18 + (-1) \cdot 156$$ by algebra

$$= 9 \cdot (330 - 156 \cdot 2) + (-1) \cdot 156$$ by substitution from step 1

$$= 9 \cdot 330 + (-19) \cdot 156$$ by algebra.

Example 8.4.6 – *Solution (3/3)*

Thus gcd(330, 156) = 9 · 330 + (−19) · 156. (It is always a good idea to check the result of a calculation like this to be sure you did not make a mistake. In this case, you find that 9 · 330 + (−19) · 156 does indeed equal 6.)

Finding an Inverse Modulo *n*

Finding an Inverse Modulo *n* (1/2)

Definition

Given any integer a and any positive integer n, if there exists an integer s such that $as \equiv 1 \pmod{n}$, then s is called **an inverse for a modulo n**.

Definition

Integers a and b are **relatively prime** if, and only if, $\gcd(a, b) = 1$. Integers $a_1, a_2, a_3, \ldots, a_n$ are **pairwise relatively prime** if, and only if, $\gcd(a_i, a_j) = 1$ for all integers i and j with $1 \leq i, j \leq n$, and $i \neq j$.

Corollary 8.4.6

If a and b are relatively prime integers, then there exist integers s and t such that $as + bt = 1$.

Example 8.4.7– *Expressing 1 as a Linear Combination of Relatively Prime Integers*

Show that 660 and 43 are relatively prime, and find a linear combination of 660 and 43 that equals 1.

Example 8.4.7 – *Solution (1/4)*

Step 1: Divide 660 by 43 to obtain $660 = 43 \cdot 15 + 15$, which implies that $15 = 660 - 43 \cdot 15$.

Step 2: Divide 43 by 15 to obtain $43 = 15 \cdot 2 + 13$, which implies that $13 = 43 - 15 \cdot 2$.

Step 3: Divide 15 by 13 to obtain $15 = 13 \cdot 1 + 2$, which implies that $2 = 15 - 13$.

Example 8.4.7 – *Solution (2/4)*

continued

Step 4: Divide 13 by 2 to obtain $13 = 2 \cdot 6 + 1$, which implies that $1 = 13 - 2 \cdot 6$.

Step 5: Divide 2 by 1 to obtain $2 = 1 \cdot 2 + 0$, which implies that $\gcd(660, 43) = 1$ and so 660 and 43 are relatively prime.

Example 8.4.7 – *Solution (3/4)* continued

Step 6: To express 1 as a linear combination of 660 and 43, substitute back through steps 4 to 1:

$1 = 13 - 2 \cdot 6$ from step 4

$= 13 - (15 - 13) \cdot 6$ by substitution from step 3

$= 7 \cdot 13 - 6 \cdot 15$ by algebra

$= 7 \cdot (43 - 15 \cdot 2) - 6 \cdot 15$ by substitution from step 2

Example 8.4.7 – *Solution (4/4)* continued

$= 7 \cdot 43 - 20 \cdot 15$ by algebra.

$= 7 \cdot 43 - 20 \cdot (660 - 43 \cdot 15)$ by substitution from step 1

$= 307 \cdot 43 - 20 \cdot 660$ by algebra.

Thus gcd(660, 43) = 1 = $307 \cdot 43 - 20 \cdot 660$. (And a check by direct computation confirms that $307 \cdot 43 - 20 \cdot 660$ does indeed equal 1.)

Finding an Inverse Modulo *n* (2/2)

Corollary 8.4.7 Existence of Inverses Modulo *n*

For all integers a and n, if $\gcd(a, n) = 1$, then there exists an integer s such that $as \equiv 1 \pmod{n}$, and so s is an inverse for a modulo n.

Example 8.4.8 – *Finding an Inverse Modulo n*

a. Find an inverse for 43 modulo 660. That is, find an integer s such that $43s \equiv 1 \pmod{660}$.

b. Find a positive inverse for 3 modulo 40. That is, find a positive integer s such that $3s \equiv 1 \pmod{40}$.

Example 8.4.8 – *Solution (1/4)*

a. By Example 8.4.7,

$$307 \cdot 43 - 20 \cdot 660 = 1.$$

Adding $20 \cdot 660$ to both sides gives that

$$307 \cdot 43 = 1 + 20 \cdot 660.$$

Thus, by definition of congruence modulo 660,

$$307 \cdot 43 \equiv 1 \pmod{660},$$

so 307 is an inverse for 43 modulo 660.

Example 8.4.8 – *Solution (2/4)*

continued

b. Use the technique of Example 8.4.7 to find a linear combination of 3 and 40 that equals 1.

Step 1: Divide 40 by 3 to obtain $40 = 3 \cdot 13 + 1$. This implies that $1 = 40 - 3 \cdot 13$.

Step 2: Divide 3 by 1 to obtain $3 = 3 \cdot 1 + 0$. This implies that $\gcd(3, 40) = 1$.

Example 8.4.8 – *Solution (3/4)*

continued

Step 3: Use the result of step 1 to write

$$3 \cdot (-13) = 1 + (-1)40.$$

This result implies that −13 is an inverse for 3 modulo 40. In other words, $3 \cdot (-13) \equiv 1 \pmod{40}$.

To find a positive inverse, compute 40 − 13. The result is 27, and

$$27 \equiv -13 \pmod{40}$$

because $27 - (-13) = 40$.

Example 8.4.8 – *Solution (4/4)*

continued

So, by Theorem 8.4.3(3),

$$3 \cdot 27 \equiv 3 \cdot (-13) \equiv 1 \pmod{40},$$

and thus by the transitive property of congruence modulo *n*, 27 is a positive integer that is an inverse for 3 modulo 40.

Theorem 8.4.3 Modular Arithmetic

Let a, b, c, d, and n be integers with $n > 1$, and suppose

$$a \equiv c \pmod{n} \quad \text{and} \quad b \equiv d \pmod{n}.$$

Then

1. $(a + b) \equiv (c + d) \pmod{n}$
2. $(a - b) \equiv (c - d) \pmod{n}$
3. $ab \equiv cd \pmod{n}$
4. $a^m \equiv c^m \pmod{n}$ for every positive integer m.

RSA Cryptography

RSA Cryptography (1/7)

In order to encrypt a message using the RSA cipher, a person needs to know the value of pq and of another integer e, both of which are made publicly available. But only a person who knows the individual values of p and q can decrypt an encrypted message.

We first give an example to show *how* the cipher works and then discuss some of the theory to explain *why* it works.

RSA Cryptography (2/7)

Suppose Alice decides to set up an RSA cipher. She chooses two prime numbers—say, $p = 5$ and $q = 11$—and computes $pq = 55$.

She then chooses a positive integer e that is relatively prime to $(p - 1)(q - 1)$. In this case, $(p - 1)(q - 1) = 4 \cdot 10 = 40$, so she may take $e = 3$ because 3 is relatively prime to 40.

RSA Cryptography (3/7)

The number pair (pq, e) is Alice's **public key**, which she may distribute widely. Because the RSA cipher works only on numbers, Alice also informs people how she will interpret the numbers in the messages they send her.

Let us suppose that she encodes letters of the alphabet in a similar way as was done for the Caesar cipher:

$$A = 01,\ B = 02,\ C = 03,\ \ldots\ ,\ Z = 26.$$

RSA Cryptography (4/7)

Let us also assume that the messages Alice receives consist of blocks, each of which, for simplicity, is taken to be a single, numerically encoded letter of the alphabet.

Someone who wants to send Alice a message breaks the message into blocks, each consisting of a single letter, and finds the numeric equivalent for each block.

RSA Cryptography (5/7)

The plaintext, *M*, in a block is converted into ciphertext, *C*, according to the following formula:

$$C = M^e \bmod pq. \qquad 8.4.5$$

Note that because (*pq*, *e*) is the public key, anyone who has it and knows modular arithmetic can encrypt a message to send to Alice.

Example 8.4.9 – *Encrypting a Message Using RSA Cryptography*

Bob wants to send Alice the message HI. What is the ciphertext for his message?

Example 8.4.9 – *Solution (1/2)*

Bob will send his message in two blocks, one for the H and another for the I. Because H is the eighth letter in the alphabet, it is encoded as 08, or 8.

The corresponding ciphertext is computed using formula 8.4.5 as follows:

$$C = 8^3 \; mod \; 55$$

$$= 512 \; mod \; 55$$

$$= 17.$$

Example 8.4.9 – *Solution (2/2)*

continued

Because I is the ninth letter in the alphabet, it is encoded as 09, or 9. The corresponding ciphertext is

$$C = 9^3 \; mod \; 55$$

$$= 729 \; mod \; 55$$

$$= 14.$$

Accordingly, Bob sends Alice the message: 17 14.

RSA Cryptography (6/7)

To decrypt the message, the *decryption key* must be computed. It is a number d that is a positive inverse to e modulo $(p - 1)(q - 1)$.

The plaintext M is obtained from the ciphertext C by the formula

$M = C^d \bmod pq$, where the number pair (pq, d) is Alice's **private key**.

8.4.6

Example 8.4.10 – *Decrypting a Message Using RSA Cryptography*

Imagine that Alice has hired you to help her decrypt messages and has shared with you the values of *p* and *q*. Compute Alice's private key (*pq*, *d*) and use the formula $M = C^d \bmod pq$ to decrypt the following ciphertext for her: 17 14.

Example 8.4.10 – *Solution (1/4)*

Because $p = 5$ and $q = 11$, $(p - 1)(q - 1) = 40$, the decryption key d is a positive inverse for 3 modulo 40. Knowing that you would need this number, we computed it in Example 8.4.8(b) and found it to be 27. Thus to decrypt the ciphertext 17, you need to compute

$$M = 17^d \bmod pq = 17^{27} \bmod 55.$$

To do so, note that

$$27 = 16 + 8 + 2 + 1.$$

Example 8.4.10 – *Solution (2/4)* continued

Next, find the residues obtained when 17 is raised to successively higher powers of 2, up to $2^4 = 16$:

$$
\begin{aligned}
17 \; mod \; 55 &= 17 \; mod \; 55 &&= 17 \\
17^2 \; mod \; 55 &= 17^2 \; mod \; 55 &&= 14 \\
17^4 \; mod \; 55 &= (17^2)^2 \; mod \; 55 &&= 14^2 \; mod \; 55 = 31 \\
17^8 \; mod \; 55 &= (17^4)^2 \; mod \; 55 &&= 31^2 \; mod \; 55 = 26 \\
17^{16} \; mod \; 55 &= (17^8)^2 \; mod \; 55 &&= 26^2 \; mod \; 55 = 16
\end{aligned}
$$

Example 8.4.10 – *Solution (3/4)* continued

Then use the fact that

$$17^{27} = 17^{16+8+2+1} = 17^{16} \cdot 17^{8} \cdot 17^{2} \cdot 17^{1}$$

to write

$$\begin{aligned} 17^{27} \; mod \; 55 &= (17^{16} \cdot 17^{8} \cdot 17^{2} \cdot 17) \; mod \; 55 \\ &\equiv [(17^{6} \; mod \; 55)(17^{8} \; mod \; 55)(17^{2} \; mod \; 55)(17 \; mod \; 55)] \; (mod \; 55) \end{aligned}$$

by Corollary 8.4.4

$$\equiv (16 \cdot 26 \cdot 14 \cdot 17) \; (mod \; 55)$$

$$\equiv 99008 \; (mod \; 55)$$

$$\equiv 8 \; (mod \; 55).$$

Example 8.4.10 – *Solution (4/4)* continued

Hence $17^{27} \, mod \, 55 = 8,$ and thus the plaintext of the first part of Bob's message is 8, or 08. In the last step, you find the letter corresponding to 08, which is *H*.

In exercises 14 and 15 at the end of this section, you are asked to show that when you decrypt 14, the result is 9, which corresponds to the letter I, so you can tell Alice that Bob's message is HI.

RSA Cryptography (7/7)

Figure 8.4.1 illustrates the process of sending and receiving a message using RSA cryptography.

Using RSA cryptography

Figure 8.4.1

Euclid’s Lemma

Euclid's Lemma (1/2)

Euclid's lemma is the crucial fact behind the unique factorization theorem for the integers and is also of great importance in many other parts of number theory.

Theorem 8.4.8 Euclid's Lemma

For all integers a, b, and c, if $\gcd(a, c) = 1$ and $a \mid bc$, then $a \mid b$.

Euclid's Lemma (2/2)

Theorem 8.4.9 Cancellation Theorem for Modular Congruence

For all integers a, b, c, and n with $n > 1$, if $\gcd(c, n) = 1$ and $ac \equiv bc \pmod{n}$, then $a \equiv b \pmod{n}$.

Fermat’s Little Theorem

Fermat's Little Theorem (1/2)

Fermat's little theorem was given that name to distinguish it from Fermat's last theorem. It provides the theoretical underpinning for RSA cryptography.

Theorem 8.4.10 Fermat's Little Theorem

If p is any prime number and a is any integer such that $p \nmid a$, then $a^{p-1} \equiv 1 \pmod{p}$.

Fermat's Little Theorem (2/2)

The process of message authentication is illustrated in Figure 8.4.2.

Message authentication

Figure 8.4.2

CHAPTER 8

PROPERTIES OF RELATIONS

Copyright © Cengage Learning. All rights reserved.

8.5 Partial Order Relations

Copyright © Cengage Learning. All rights reserved.

Antisymmetry

Antisymmetry (1/2)

In earlier section, we have defined three properties of relations: reflexivity, symmetry, and transitivity. A fourth property of relations is called *antisymmetry*.

In terms of the arrow diagram of a relation, saying that a relation is antisymmetric is the same as saying that whenever there is an arrow going from one element to another *distinct* element, there is *not* an arrow going back from the second to the first.

Antisymmetry (2/2)

Definition

Let R be a relation on a set A. R is **antisymmetric** if, and only if,

$$\text{for every } a \text{ and } b \text{ in } A, \quad \text{if } a\,R\,b \text{ and } b\,R\,a \text{ then } a = b.$$

By taking the negation of the definition, you can see that a relation R is ***not*** **antisymmetric** if, and only if,

there are elements a and b in A such that $a\,R\,b$ and $b\,R\,a$ but $a \neq b$.

Example 8.5.1 – *Testing for Antisymmetry of Finite Relations*

Let R_1 and R_2 be the relations on {0, 1, 2} defined as follows: Draw the directed graphs for R_1 and R_2 and indicate which relations are antisymmetric.

a. $R_1 = \{(0, 2), (1, 2), (2, 0)\}$

b. $R_2 = \{(0, 0), (0, 1), (0, 2), (1, 1), (1, 2)\}$

Example 8.5.1 – *Solution*

a. R_1 is not antisymmetric.

Since $0\ R_1\ 2$ and $2\ R_1\ 0$ but $0 \neq 2$, R_1 is not antisymmetric.

b. R_2 is antisymmetric.

In order for R_2 not to be antisymmetric, there would have to exist a pair of distinct elements of A such that each is related to the other by R_2. But you can see by inspection that no such pair exists.

Example 8.5.2 – *Testing for Antisymmetry of "Divides" Relations*

Let R_1 be the "divides" relation on the set of all positive integers, and let R_2 be the "divides" relation on the set of all integers.

$$\text{For every } a, b \in Z^+, \quad a\, R_1\, b \;\Leftrightarrow\; a \mid b.$$
$$\text{For every } a, b \in Z, \quad a\, R_2\, b \;\Leftrightarrow\; a \mid b.$$

a. Is R_1 antisymmetric? Prove or give a counterexample.

b. Is R_2 antisymmetric? Prove or give a counterexample.

Example 8.5.2 – *Solution (1/2)*

a. R_1 is antisymmetric.

Proof: Suppose a and b are positive integers such that a R_1 b and b R_1 a. *[We must show that $a = b$.]* By definition of R_1, $a \mid b$ and $b \mid a$.

Thus, by definition of divides, there are integers k_1 and k_2 with $b = k_1 a$ and $a = k_2 b$. It follows that

$$b = k_1 a = k_1(k_2 b) = (k_1 k_2) b.$$

Dividing both sides by b gives

$$k_1 k_2 = 1.$$

Example 8.5.2 – *Solution (2/2)*

continued

Now since a and b are both integers, k_1 and k_2 are both positive integers also. And the only product of two positive integers that equals 1 is $1 \cdot 1$. Thus

$$k_1 = k_2 = 1$$

and so $$a = k_2 b = 1 \cdot b = b.$$

b. R_2 is not antisymmetric.

Counterexample:

Let $a = 2$ and $b = -2$. Then $a \mid b$ *[since* $-2 = (-1) \cdot 2$*]* and $b \mid a$ *[since* $2 = (-1)(-2)$*]*. Hence $a \, R_2 \, b$ and $b \, R_2 \, a$ but $a \neq b$.

Partial Order Relations

Partial Order Relations (1/2)

A relation that is reflexive, antisymmetric, and transitive is called a *partial order*.

Definition

Let R be a relation defined on a set A. R is a **partial order relation** if, and only if, R is reflexive, antisymmetric, and transitive.

Two fundamental partial order relations are the “less than or equal to” relation on a set of real numbers and the “subset” relation on a set of sets. These can be thought of as models, or paradigms, for general partial order relations.

Example 8.5.3 – *The "Subset" Relation*

Let $\mathscr{A}$ be any collection of sets and define the "subset" relation, $\subseteq$, on $\mathscr{A}$ as follows: For every $U, V \in \mathscr{A}$

$$\boxed{U \subseteq V \quad \Leftrightarrow \quad \text{for each } x, \text{ if } x \in U \text{ then } x \in V.}$$

We know that $\subseteq$ is reflexive and transitive. Finish the proof that $\subseteq$ is a partial order relation by proving that $\subseteq$ is antisymmetric.

Example 8.5.3 – *Solution*

For $\subseteq$ to be antisymmetric means that for all sets U and V in $\mathscr{A}$, if $U \subseteq V$ and $V \subseteq U$ then $U = V$. This is true by definition of equality of sets.

Example 8.5.4 – *A "Divides" Relation on a Set of Positive Integers*

Let $|$ be the "divides" relation on a set A of positive integers. That is, for all a and b in A,

$$a \mid b \quad \Leftrightarrow \quad b = ka \text{ for some integer } k.$$

Prove that $|$ is a partial order relation on A.

Example 8.5.4 – *Solution (1/2)*

$|$ ***is reflexive*****:** *[We must show that for each a* $\in$ *A* $a|a$. *]*
Suppose *a* $\in$ *A*. Then *a* = 1 ⋅ *a*, so $a|a$ by definition of divisibility.

$|$ ***is antisymmetric*****:** *[We must show that for every a, b* $\in$ *A,* $a|b$ *and* $b|a$ *then a = b.]* The proof of this is virtually identical to that of Example 8.5.2(a).

$|$ ***is transitive*****:** To show transitivity means to show that for every *a*, *b*, *c* $\in$ *A*, if $a|b$ and $b|c$ then $a|c$.

Example 8.5.4 – *Solution (2/2)*

continued

But this is the transitivity of divisibility property, which was proved as Theorem 4.4.3.

Theorem 4.4.3 Transitivity of Divisibility

For all integers a, b, and c, if a divides b and b divides c, then a divides c.

Since $|$ is reflexive, antisymmetric, and transitive, $|$ is a partial order relation on A.

Example 8.5.5 – *The "Less Than or Equal to" Relation*

Let S be a set of real numbers and define the "less than or equal to" relation, ≤, on S as follows: For all real numbers x and y in S,

$$\boxed{x \leq y \quad \Leftrightarrow \quad x < y \text{ or } x = y.}$$

Show that ≤ is a partial order relation.

Example 8.5.5 – *Solution (1/2)*

$\leq$ *is reflexive*: For $\leq$ to be reflexive means that $x \leq x$ for every real number x in S. But $x \leq x$ means that $x < x$ or $x = x$, and $x = x$ is always true.

$\leq$ *is antisymmetric*: For $\leq$ to be antisymmetric means that for all real numbers x and y in S, if $x \leq y$ and $y \leq x$ then $x = y$. This follows immediately from the definition of $\leq$ and the trichotomy property, which says that given any real numbers x and y, exactly one of the following holds: $x < y$ or $x = y$ or $x > y$.

Example 8.5.5 – *Solution (2/2)* continued

≤ *is transitive*: For ≤ to be transitive means that for all real numbers x, y, and z in S if $x \leq y$ and $y \leq z$ then $x \leq z$.

This follows from the definition of ≤ and the transitivity property of order, which says that given any real numbers x, y, and z, if $x < y$ and $y < z$ then $x < z$.

Because ≤ is reflexive, antisymmetric, and transitive, it is a partial order relation.

Partial Order Relations (2/2)

Notation

Because of the special paradigmatic role played by the $\leq$ relation in the study of partial order relations, the symbol $\preceq$ is often used to refer to a general partial order relation, and the notation $x \preceq y$ is read "x is less than or equal to y" or "y is greater than or equal to x."

Lexicographic Order

Lexicographic Order (1/1)

Theorem 8.5.1 Lexicographic Order

Let A be a set with a partial order relation R, and let S be a set of strings over A. Define a relation $\preceq$ on S as follows:

Let s and t be any strings in S of lengths m and n, respectively, where m and n are positive integers, and let s_m and t_m be the characters in the mth position for s and t, respectively.

1. If $m \leq n$ and the first m characters of s and t are the same, then $s \preceq t$.
2. If the first $m - 1$ characters in s and t are the same, $s_m \, R \, t_m$, and $s_m \neq t_m$, then $s \preceq t$.
3. If λ is the null string then $\lambda \preceq s$.

If no strings are related by $\preceq$ other than by these three conditions, then $\preceq$ is a partial order relation on S.

Definition

The partial order relation of Theorem 8.5.1 is called the **lexicographic order for S** that corresponds to the partial order R on A.

Example 8.5.6 – *Testing Strings for Lexicographic Order*

Let $A = \{x, y\}$ and let R be the following partial order relation on A: $R = \{(x, x), (x, y), (y, y)\}$.

Let S be the set of all strings over A, and denote by the lexicographic order for S that corresponds to R.

a. Is $x \preceq x$? Is $x \preceq xx$? Is $yx \preceq yxy$?

b. Is $xxxyyy \preceq xy$?

c. Is $x \preceq y$?

d. Is $\lambda \preceq xy$?

e. Is $xyy \preceq xyx$?

Example 8.5.6 – *Solution*

a. Yes in all three cases, by property (1) of the definition of $\preceq$.

b. Yes in all cases, by property (2) of the definition of $\preceq$.

c. Yes in all cases, by property (2) of the definition of $\preceq$. In this case $m - 1 = 0$, and the statement that the first zero characters of x and y are the same is true by default.

d. Yes by property (3) of the definition of $\preceq$.

e. No because y is not related to x by R.

Hasse Diagrams

Hasse Diagrams (1/4)

Let $A = \{1, 2, 3, 9, 18\}$ and consider the "divides" relation on A: For every $a, b \in A$,

$$a \mid b \quad \Leftrightarrow \quad b = ka \text{ for some integer } k.$$

The directed graph of this relation has the following appearance:

Hasse Diagrams (2/4)

Note that there is a loop at every vertex, all other arrows point in the same direction (upward), and any time there is an arrow from one point to a second and from the second point to a third, there is an arrow from the first point to the third.

Given any partial order relation defined on a finite set, it is possible to draw the directed graph in such a way that all of these properties are satisfied.

Hasse Diagrams (3/4)

This makes it possible to associate a somewhat simpler graph, called a **Hasse diagram** (after Helmut Hasse, a twentieth-century German number theorist), with a partial order relation defined on a finite set.

To obtain a Hasse diagram, proceed as follows: Start with a directed graph of the relation, placing vertices on the page so that all arrows point upward.

Hasse Diagrams (4/4)

Then eliminate

1. the loops at all the vertices,
2. all arrows whose existence is implied by the transitive property, and
3. the direction indicators on the arrows.

For the relation given previously, the Hasse diagram is as follows:

Example 8.5.7 – *Constructing a Hasse Diagram*

Consider the "subset" relation, $\subseteq$, on the set $\mathcal{P}(\{a, b, c\})$. That is, for all sets *U* and *V* in $\mathcal{P}(\{a, b, c\})$,

$$\boxed{U \subseteq V \quad \Leftrightarrow \quad \forall x, \text{if } x \in U \text{ then } x \in V.}$$

Construct the Hasse diagram for this relation.

Example 8.5.7 – *Solution (1/2)*

Draw the directed graph of the relation in such a way that all arrows except loops point upward.

Example 8.5.7 – *Solution (2/2)*

continued

Then strip away all loops, unnecessary arrows, and direction indicators to obtain the Hasse diagram.

Example 8.5.8 – *Obtaining the Directed Graph of a Partial Order Relation from the Hasse Diagram of the Relation*

A partial order relation R has the following Hasse diagram. Find the directed graph of R.

Example 8.5.8 – *Solution*

Partially and Totally Ordered Sets

Partially and Totally Ordered Sets (1/5)

Given any two real numbers x and y, either $x \leq y$ or $y \leq x$. In a situation like this, the elements x and y are said to be *comparable*. On the other hand, given two subsets A and B of $\{a, b, c\}$, it may be the case that neither $A \subseteq B$ nor $B \subseteq A$. For instance, let $A = \{a, b\}$ and $B = \{b, c\}$. Then $A \nsubseteq B$ and $B \nsubseteq A$. In such a case, A and B are said to be *noncomparable*.

Definition

Suppose $\preceq$ is a partial order relation on a set A. Elements a and b of A are said to be **comparable** if, and only if, either $a \preceq b$ or $b \preceq a$. Otherwise, a and b are called **noncomparable**.

Partially and Totally Ordered Sets (2/5)

When all the elements of a partial order relation are comparable, the relation is called a *total order*.

Definition

If R is a partial order relation on a set A, and for any two elements a and b in A either $a\ R\ b$ or $b\ R\ a$, then R is a **total order relation** on A.

Both the "less than or equal to" relation on sets of real numbers and the lexicographic order of the set of words in a dictionary are total order relations. Note that the Hasse diagram for a total order relation can be drawn as a single vertical "chain."

Partially and Totally Ordered Sets (3/5)

A set A is called a **partially ordered set** (or **poset**) with respect to a relation $\preceq$ if, and only if, $\preceq$ is a partial order relation on *A*.

For instance, the set of real numbers is a partially ordered set with respect to the "less than or equal to" relation ≤, and a set of sets is partially ordered with respect to the "subset" relation $\subseteq$.

It is entirely straightforward to show that *any subset of a partially ordered set is partially ordered.*

Partially and Totally Ordered Sets (4/5)

This, of course, assumes the "same definition" for the relation on the subset as for the set as a whole. A set *A* is called a **totally ordered set** with respect to a relation $\preceq$ if, and only if, *A* is partially ordered with respect to $\preceq$ and $\preceq$ is a total order. A set that is partially ordered but not totally ordered may have totally ordered subsets. Such subsets are called *chains*.

Definition

Let A be a set that is partially ordered with respect to a relation $\preceq$. A subset B of A is called a **chain** if, and only if, the elements in each pair of elements in B are comparable. In other words, $a \preceq b$ or $b \preceq a$ for every a and b in B. The **length of a chain** is one less than the number of elements in the chain.

Example 8.5.9 – *A Chain of Subsets*

The set $\mathcal{P}(\{a, b, c\})$ is partially ordered with respect to the subset relation. Find a chain of length 3 in $\mathcal{P}(\{a, b, c\})$.

Example 8.5.9 – *Solution*

Since $\varnothing \subseteq \{a\} \subseteq \{a, b\} \subseteq \{a, b, c\}$, the set

$$S = \{\varnothing, \{a\}, \{a, b\}, \{a, b, c\}\}$$

is a chain of length 3 in $\mathcal{P}(\{a, b, c\})$.

Partially and Totally Ordered Sets (5/5)

A *maximal element* in a partially ordered set is an element that is greater than or equal to every element *to which it is comparable*. A *greatest element* in a partially ordered set is an element that is greater than or equal to *every* element in the set. Minimal and least elements are defined similarly.

Definition

Let a set A be partially ordered with respect to a relation $\preceq$.

1. An element a in A is called a **maximal element of A** if, and only if, for each b in A, either $b \preceq a$ or b and a are not comparable.
2. An element a in A is called a **greatest element of A** if, and only if, for each b in A, $b \preceq a$.
3. An element a in A is called a **minimal element of A** if, and only if, for each b in A, either $a \preceq b$ or b and a are not comparable.
4. An element a in A is called a **least element of A** if, and only if, for each b in A, $a \preceq b$.

Example 8.5.10 – *Maximal, Minimal, Greatest, and Least Elements*

Let $A = \{a, b, c, d, e, f, g, h, i\}$ have the partial ordering $\preceq$ defined by the following Hasse diagram.

Find all maximal, minimal, greatest, and least elements of A.

Example 8.5.10 – *Solution*

There is just one maximal element, *g*, which is also the greatest element. The minimal elements are *c*, *d*, and *i*, and there is no least element.

Topological Sorting

Topological Sorting (1/2)

Definition

Given partial order relations $\preceq$ and $\preceq'$ on a set A, $\preceq'$ is **compatible** with $\preceq$ if, and only if, for every a and b in A, if $a \preceq b$ then $a \preceq' b$.

A total order that is compatible with a given order is called a *topological sorting*.

Definition

Given partial order relations $\preceq$ and $\preceq'$ on a set A, $\preceq'$ is a **topological sorting** for $\preceq$ if, and only if, $\preceq'$ is a total order that is compatible with $\preceq$.

Topological Sorting (2/2)

Here is a formal version of the algorithm:

Constructing a Topological Sorting

Let $\preceq$ be a partial order relation on a nonempty finite set A. To construct a topological sorting:

1. Pick any minimal element x in A. *[Such an element exists since A is nonempty.]*
2. Set $A' := A - \{x\}$.
3. Repeat steps *a–c* while $A' \neq \varnothing$.
   a. Pick any minimal element y in A'.
   b. Define $x \preceq' y$.
   c. Set $A' := A' - \{y\}$ and $x := y$.

   [Completion of steps 1–3 of this algorithm gives enough information to construct the Hasse diagram for the total ordering $\preceq$. We have already shown how to use the Hasse diagram to obtain a complete directed graph for a relation.]

Example 8.5.11 – *A Topological Sorting*

Consider the set $A = \{2, 3, 4, 6, 18, 24\}$ ordered by the "divides" relation $|$. The Hasse diagram of this relation is the following:

The ordinary "less than or equal to" relation ≤ on this set is a topological sorting for it since for positive integers *a* and *b*, if $a|b$ then $a \leq b$. Find another topological sorting for this set.

Example 8.5.11 – *Solution (1/3)*

The set has two minimal elements: 2 and 3. Either one may be chosen; suppose you pick 3. The beginning of the total order is

total order: 3.

Set $A' = A - \{3\}$. You can indicate this by removing 3 from the Hasse diagram as shown below.

Example 8.5.11 – *Solution (2/3)* continued

Next choose a minimal element from $A' - \{3\}$. Only 2 is minimal, so you must pick it. The total order thus far is

$$\text{total order: } 3 \preceq 2.$$

Set $A' = (A - \{3\}) - \{2\} = A - \{3, 2\}$. You can indicate this by removing 2 from the Hasse diagram, as is shown below.

Choose a minimal element from $A' - \{3, 2\}$. Again you have two choices: 4 and 6. Suppose you pick 6.

Example 8.5.11 – *Solution (3/3)* continued

The total order for the elements chosen thus far is

$$\text{total order: } 3 \preceq 2 \preceq 6.$$

You continue in this way until every element of *A* has been picked. One possible sequence of choices gives

$$\text{total order: } 3 \preceq 2 \preceq 6 \preceq 18 \preceq 4 \preceq 24.$$

You can verify that this order is compatible with the "divides" partial order by checking that for each pair of elements *a* and *b* in *A* such that $a \mid b$, then $a \preceq b$. Note that it is *not* the case that $a \preceq b$ then $a \mid b$.

An Application

An Application (1/4)

Note that the following defines a partial order relation on the set of courses required for a university degree:

For all required courses *x* and *y*,

$$x \preceq y \quad \Leftrightarrow \quad x = y \quad \text{or} \quad x \text{ is a prerequisite for } y.$$

If the Hasse diagram for the relation is drawn, then the questions raised at the beginning of this section can be answered easily.

An Application (2/4)

For instance, consider the Hasse diagram for the requirements at a particular university, which is shown in Figure 8.5.1.

Figure 8.5.1

An Application (3/4)

The minimum number of school terms needed to complete the requirements is the size of a longest chain, which is 7 (150, 155, 225, 300, 340, 360, 390, for example).

The maximum number of courses that could be taken in the same term (assuming the university allows it) is the maximum number of noncomparable courses, which is 6 (350, 360, 345, 301, 230, 200, for example).

An Application (4/4)

A part-time student could take the courses in a sequence determined by constructing a topological sorting for the set. (One such sorting is 140, 150, 141, 155, 200, 225, 230, 300, 250, 301, 340, 345, 350, 360, 390. There are many others.)

PERT and CPM

PERT and CPM (1/2)

Two important and widely used applications of partial order relations are **PERT** (Program Evaluation and Review Technique) and **CPM** (Critical Path Method).

These techniques were developed in the 1950s as planners came to grips with the complexities of scheduling the individual activities needed to complete very large projects, and although they are very similar, their developments were independent.

PERT and CPM (2/2)

PERT was developed by the U.S. Navy to help organize the construction of the Polaris submarine, and CPM was developed by the E. I. Du Pont de Nemours company for scheduling chemical plant maintenance.

Here is a somewhat simplified example of the way the techniques work.

Example 8.5.12 – *A Job Scheduling Problem (1/10)*

At an automobile assembly plant, the job of assembling an automobile can be broken down into these tasks:

1. Build frame.
2. Install engine, power train components, gas tank.
3. Install brakes, wheels, tires.
4. Install dashboard, floor, seats.
5. Install electrical lines.
6. Install gas lines.

Example 8.5.12 – *A Job Scheduling Problem (2/10)*

continued

7. Install brake lines.
8. Attach body panels to frame.
9. Paint body.

Certain of these tasks can be carried out at the same time, whereas some cannot be started until other tasks are finished.

Example 8.5.12 – *A Job Scheduling Problem (3/10)*

continued

Table 8.5.1 summarizes the order in which tasks can be performed and the time required to perform each task.

Task	Immediately Preceding Tasks	Time Needed to Perform Task
1		7 hours
2	1	6 hours
3	1	3 hours
4	2	6 hours
5	2, 3	3 hours
6	4	1 hour
7	2, 3	1 hour
8	4, 5	2 hours
9	6, 7, 8	5 hours

Table 8.5.1

Example 8.5.12 – *A Job Scheduling Problem (4/10)*

continued

Let T be the set of all tasks, and consider the partial order relation $\preceq$ defined on T as follows: For all tasks x and y in T,

$$x \preceq y \quad \Leftrightarrow \quad x = y \text{ or } x \text{ precedes } y.$$

If the Hasse diagram of this relation is turned sideways (as is customary in PERT and CPM analysis), it has the appearance shown on the right side.

Example 8.5.12 – *A Job Scheduling Problem (5/10)*

continued

What is the minimum time required to assemble a car? You can determine this by working from left to right across the diagram, noting for each task (say, just above the box representing that task) the minimum time needed to complete that task starting from the beginning of the assembly process.

For instance, you can put a 7 above the box for task 1 because task 1 requires 7 hours.

Example 8.5.12 – *A Job Scheduling Problem (6/10)*

continued

Task 2 requires completion of task 1 (7 hours) plus 6 hours for itself, so the minimum time required to complete task 2, starting at the beginning of the assembly process, is
7 + 6 = 13 hours.

So you can put a 13 above the box for task 2. Similarly, you can put a 10 above the box for task 3 because 7 + 3 = 10.

Now consider what number you should write above the box for task 5.

Example 8.5.12 – *A Job Scheduling Problem (7/10)*

continued

The minimum times to complete tasks 2 and 3, starting from the beginning of the assembly process, are 13 and 10 hours respectively.

Since *both* tasks must be completed before task 5 can be started, the minimum time to complete task 5, starting from the beginning, is the time needed for task 5 itself (3 hours) plus the *maximum* of the times to complete tasks 2 and 3 (13 hours), and this equals 3 + 13 = 16 hours.

Thus you should place the number 16 above the box for task 5.

Example 8.5.12 – *A Job Scheduling Problem (8/10)*

continued

The same reasoning leads you to place a 14 above the box for task 7. Similarly, you can place a 19 above the box for task 4, a 20 above the box for task 6, a 21 above the box for task 8, and a 26 above the box for task 9, as shown below.

Example 8.5.12 – *A Job Scheduling Problem (9/10)*

continued

This analysis shows that at least 26 hours are required to complete task 9 starting from the beginning of the assembly process.

When task 9 is finished, the assembly is complete, so 26 hours is the minimum time needed to accomplish the whole process.

Note that the minimum time required to complete tasks 1, 2, 4, 8, and 9 in sequence is exactly 26 hours.

Example 8.5.12 – *A Job Scheduling Problem (10/10)* continued

This means that a delay in performing any one of these tasks causes a delay in the total time required for assembly of the car.

For this reason, the path through tasks 1, 2, 4, 8, and 9 is called a **critical path**.

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.1 Introduction to Probability

Copyright © Cengage Learning. All rights reserved.

Introduction to Probability (1/7)

Imagine tossing two coins and observing whether 0, 1, or 2 heads are obtained. It would be natural to guess that each of these events occurs about one-third of the time, but in fact this is not the case. Table 9.1.1 shows actual data obtained from tossing two quarters 50 times.

Event	Tally	Frequency (Number of times the event occurred)	Relative Frequency (Fraction of times the event occurred)
2 heads obtained	𝍸 𝍸 |	11	22%
1 head obtained	𝍸 𝍸 𝍸 𝍸 ||	27	54%
0 heads obtained	𝍸 𝍸 ||	12	24%

Experimental Data Obtained from Tossing Two Quarters 50 Times

Table 9.1.1

Introduction to Probability (2/7)

To formalize this analysis and extend it to more complex situations, we introduce the notions of random process, sample space, event, and probability.

To say that a process is **random** means that when it takes place, one outcome from some set of outcomes is sure to occur, but it is impossible to predict with certainty which outcome that will be.

Introduction to Probability (3/7)

For instance, if an ordinary person performs the experiment of tossing an ordinary coin into the air and allowing it to fall flat on the ground, it can be predicted with certainty that the coin will land either heads up or tails up (so the set of outcomes can be denoted {heads, tails}), but it is not known for sure whether heads or tails will occur.

Introduction to Probability (4/7)

We restricted this experiment to ordinary people because a skilled magician can toss a coin in a way that appears random but is not, and a physicist equipped with first-rate measuring devices may be able to analyze all the forces on the coin and correctly predict its landing position.

Just a few of many examples of random processes or experiments are choosing winners in state lotteries, selecting respondents in public opinion polls, and choosing subjects to receive treatments or serve as controls in medical experiments.

Introduction to Probability (5/7)

The set of outcomes that can result from a random process or experiment is called a *sample space.*

Definition

A **sample space** is the set of all possible outcomes of a random process or experiment. An **event** is a subset of a sample space.

In the case where an experiment has finitely many outcomes and all outcomes are equally likely to occur, the *probability* of an event (set of outcomes) is just the ratio of the number of outcomes in the event to the total number of outcomes.

Introduction to Probability (6/7)

Equally Likely Probability Formula

If S is a finite sample space in which all outcomes are equally likely and E is an event in S, then the **probability of E**, denoted $P(E)$, is

$$P(E) = \frac{\text{the number of outcomes in } E}{\text{the total number of outcomes in } S}.$$

Introduction to Probability (7/7)

Notation

For any finite set A, $N(A)$ denotes the number of elements in A.

With this notation, the equally likely probability formula becomes

$$P(E) = \frac{N(E)}{N(S)}.$$

Example 9.1.1 – *Probabilities for a Deck of Cards (1/3)*

An ordinary deck of cards contains 52 cards divided into four *suits.* The *red suits* are diamonds (♦) and hearts (♥), and the *black suits* are clubs (♣) and spades (♠).

Each suit contains 13 cards of the following *denominations:* 2, 3, 4, 5, 6, 7, 8, 9, 10, J (jack), Q (queen), K (king), and A (ace). The cards J, Q, and K are called *face cards.*

Mathematician Persi Diaconis, working with David Aldous in 1986 and Dave Bayer in 1992, showed that seven shuffles are needed to “thoroughly mix up” the cards in an ordinary deck.

Example 9.1.1 – *Probabilities for a Deck of Cards (2/3)*

continued

In 2000 mathematician Nick Trefethen, working with his father, Lloyd Trefethen, a mechanical engineer, used a somewhat different definition of “thoroughly mix up” to show that six shuffles will nearly always suffice.

Imagine that the cards in a deck have become—by some method—so thoroughly mixed up that if you spread them out face down and pick one at random, you are as likely to get any one card as any other.

Example 9.1.1 – *Probabilities for a Deck of Cards (3/3)*

continued

a. What is the sample space of outcomes?

b. What is the event that the chosen card is a black face card?

c. What is the probability that the chosen card is a black face card?

Example 9.1.1 – *Solution (1/2)*

a. The outcomes in the sample space *S* are the 52 cards in the deck.

b. Let *E* be the event that a black face card is chosen. The outcomes in *E* are the jack, queen, and king of clubs and the jack, queen, and king of spades. Symbolically:

$$E = \{J\clubsuit, Q\clubsuit, K\clubsuit, J\spadesuit, Q\spadesuit, K\spadesuit\}.$$

Example 9.1.1 – *Solution (2/2)*

continued

c. By part (b), $N(E) = 6$, and according to the description of the situation, all 52 outcomes in the sample space are equally likely. Therefore, by the equally likely probability formula, the probability that the chosen card is a black face card is

$$P(E) = \frac{N(E)}{N(S)} = \frac{6}{52} \cong 11.5\%$$

Counting the Elements of a List

Counting the Elements of a List (1/3)

Some counting problems are as simple as counting the elements of a list. For instance, how many integers are there from 5 through 12? To answer this question, imagine going along the list of integers from 5 to 12, counting each in turn.

list:	5	6	7	8	9	10	11	12
	↕	↕	↕	↕	↕	↕	↕	↕
count:	1	2	3	4	5	6	7	8

So the answer is 8.

Counting the Elements of a List (2/3)

More generally, if m and n are integers and $m \leq n$, how many integers are there from m through n? To answer this question, note that $n = m + (n - m)$, where $n - m \geq 0$ *[since* $n \geq m$*]*.

Note also that the element $m + 0$ is the first element of the list, the element $m + 1$ is the second element, the element $m + 2$ is the third, and so forth.

Counting the Elements of a List (3/3)

In general, the element $m + i$ is the $(I + 1)$st element of the list.

$$\begin{array}{lccccc} \text{list:} & m\ (=m+0) & m+1 & m+2 & \ldots & n\ (=m+(n-m)) \\ & \updownarrow & \updownarrow & \updownarrow & & \updownarrow \\ \text{count:} & 1 & 2 & 3 & \ldots & (n-m)+1 \end{array}$$

And so the number of elements in the list is $n - m + 1$.

Theorem 9.1.1 The Number of Elements in a List

If m and n are integers and $m \leq n$, then there are $n - m + 1$ integers from m to n inclusive.

Example 9.1.4 – *Counting the Elements of a Sublist*

a. How many three-digit integers (integers from 100 to 999 inclusive) are divisible by 5?

b. What is the probability that a randomly chosen three-digit integer is divisible by 5?

Example 9.1.4 – *Solution (1/2)*

a. Imagine writing the three-digit integers in a row, noting those that are multiples of 5 and drawing arrows between each such integer and its corresponding multiple of 5.

100	101	102	103	104	105	106	107	108	109	110	. . .	994	995	996	997	998	999
$\updownarrow$					$\updownarrow$					$\updownarrow$			$\updownarrow$				
$5 \cdot 20$					$5 \cdot 21$					$5 \cdot 22$			$5 \cdot 199$				

From the sketch it is clear that there are as many three-digit integers that are multiples of 5 as there are integers from 20 to 199 inclusive.

Example 9.1.4 – *Solution (2/2)* continued

By Theorem 9.1.1, there are 19922011, or 180, such integers. Hence there are 180 three-digit integers that are divisible by 5.

b. By Theorem 9.1.1 the total number of integers from 100 through 999 is $999 - 100 + 1 = 900$. By part (a), 180 of these are divisible by 5. Hence the probability that a randomly chosen three-digit integer is divisible by 5 is $180/900 = 1/5$.

Example 9.1.5 – *Application: Counting Elements of a One-Dimensional Array (1/2)*

Analysis of many computer algorithms requires skill at counting the elements of a one-dimensional array. Let $A[1]$, $A[2]$, …, $A[n]$ be a one-dimensional array, where n is a positive integer.

a. Suppose the array is cut at a middle value $A[m]$ so that two subarrays are formed:

   (1) $A[1]$, $A[2]$, …, $A[m]$ and (2) $A[m + 1]$, $A[m + 2]$, …, $A[n]$.

   How many elements does each subarray have?

Example 9.1.5 – *Application: Counting Elements of a One-Dimensional Array (2/2)* continued

b. What is the probability that a randomly chosen element of the array has an even subscript

(i) if n is even? (ii) if n is odd?

Example 9.1.5 – *Solution (1/5)*

a. Array (1) has the same number of elements as the list of integers from 1 through m. So by Theorem 9.1.1, it has m, or $m - 1 + 1$, elements. Array (2) has the same number of elements as the list of integers from $m + 1$ through n. So by Theorem 9.1.1, it has $n - m$, or $n - (m + 1) + 1$, elements.

Theorem 9.1.1 The Number of Elements in a List

If m and n are integers and $m \leq n$, then there are $n - m + 1$ integers from m to n inclusive.

Example 9.1.5 – *Solution (2/5)* continued

b. (i) If n is even, each even subscript starting with 2 and ending with n can be matched up with an integer from 1 to $n/2$.

$$\begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \ldots & n \\ & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow \\ & 2\cdot 1 & & 2\cdot 2 & & 2\cdot 3 & & 2\cdot 4 & & 2\cdot 5 & & 2\cdot(n/2) \end{array}$$

So there are $n/2$ array elements with even subscripts. Since the entire array has n elements, the probability that a randomly chosen element has an even subscript is

$$\frac{n/2}{n} = \frac{1}{2}.$$

Example 9.1.5 – *Solution (3/5)* continued

(ii) If n is odd, then the greatest even subscript of the array is $n - 1$. So there are as many even subscripts between 1 and n as there are from 2 through $n - 1$.

Then the reasoning of (i) can be used to conclude that there are $(n - 1)/2$ array elements with even subscripts.

$$\begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & \ldots & n-1 & n \\ & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow & \\ & 2\cdot 1 & & 2\cdot 2 & & 2\cdot 3 & \ldots & 2\cdot[(n-1)/2)] & \end{array}$$

Example 9.1.5 – *Solution (4/5)* continued

Since the entire array has *n* elements, the probability that a randomly chosen element has an even subscript is

$$\frac{(n-1)/2}{n} = \frac{n-1}{2n}.$$

Observe that as *n* gets larger and larger, this probability gets closer and closer to 1/2.

Example 9.1.5 – *Solution (5/5)*

continued

Note that the answers to (i) and (ii) can be combined using the floor notation.

Theorem 4.6.2 The Floor of $n/2$

For any integer n,

$$\left\lfloor \frac{n}{2} \right\rfloor = \begin{cases} \dfrac{n}{2} & \text{if } n \text{ is even} \\ \dfrac{n-1}{2} & \text{if } n \text{ is odd.} \end{cases}$$

By Theorem 4.6.2, the number of array elements with even subscripts is $\lfloor n/2 \rfloor$, so the probability that a randomly chosen element has an even subscript is $\frac{\lfloor n/2 \rfloor}{n}$.

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.2 Possibility Trees and the Multiplication Rule

Copyright © Cengage Learning. All rights reserved.

Possibility Trees and the Multiplication Rule

A tree structure is a useful tool for keeping systematic track of all possibilities in situations in which events happen in order. The following example shows how to use such a structure to count the number of different outcomes of a tournament.

Example 9.2.1 – *Possibilities for Tournament Play*

Teams *A* and *B* are to play each other repeatedly until one wins two games in a row or a total of three games. One way in which this tournament can be played is for *A* to win the first game, *B* to win the second, and *A* to win the third and fourth games. Denote this by writing *A*–*B*–*A*–*A*.

a. How many ways can the tournament be played?

b. Assuming that all the ways of playing the tournament are equally likely, what is the probability that five games are needed to determine the tournament winner?

Example 9.2.1 – *Solution (1/3)*

a. The possible ways for the tournament to be played are represented by the distinct paths from “root” (the start) to “leaf” (a terminal point) in the tree shown sideways in Figure 9.2.1. The label on each branching point indicates the winner of the game. The notations in parentheses indicate the winner of the tournament.

The Outcomes of a Tournament

Figure 9.2.1

Example 9.2.1 – *Solution (2/3)* continued

The fact that there are ten paths from the root of the tree to its leaves shows that there are ten possible ways for the tournament to be played. They are (moving from the top down): *A–A*, *A–B–A–A, A–B–A–B–A*, *A–B–A–B–B, A–B–B, B–A–A, B–A–B–A–A*, *B–A–B–A–B, B–A–B–B*, and *B–B*.

In five cases *A* wins, and in the other five *B* wins. The least number of games that must be played to determine a winner is two, and the most that will need to be played is five.

Example 9.2.1 – *Solution (3/3)* continued

b. Since all the possible ways of playing the tournament listed in part (a) are assumed to be equally likely, and the listing shows that five games are needed in four different cases (*A*–*B*–*A*–*B*–*A*, *A*–*B*–*A*–*B*–*B*, *B*–*A*–*B*–*A*–*B*, and *B*–*A*–*B*–*A*–*A*), the probability that five games are needed is $4/10 = 2/5 = 40\%$.

The Multiplication Rule

The Multiplication Rule (1/4)

Consider the following example. Suppose a computer installation has four input/output units (*A*, *B*, *C*, and *D*) and three central processing units (*X*, *Y*, and *Z*). Any input/output unit can be paired with any central processing unit. How many ways are there to pair an input/output unit with a central processing unit?

To answer this question, imagine the pairing of the two types of units as a two-step operation:

Step 1: Choose the input/output unit.

Step 2: Choose the central processing unit.

The Multiplication Rule (2/4)

The possible outcomes of this operation are illustrated in the possibility tree of Figure 9.2.2.

Pairing Objects Using a Possibility Tree

Figure 9.2.2

The Multiplication Rule (3/4)

The topmost path from “root” to “leaf” indicates that input/output unit *A* is to be paired with central processing unit *X.* The next lower branch indicates that input/output unit *A* is to be paired with central processing unit *Y.* And so forth.

Thus the total number of ways to pair the two types of units is the same as the number of branches of the tree, which is

$$3 + 3 + 3 + 3 = 4 \cdot 3 = 12.$$

The Multiplication Rule (4/4)

The idea behind this example can be used to prove the following rule. A formal proof uses mathematical induction and is left to the exercises.

Theorem 9.2.1 The Multiplication Rule

If an operation consists of k steps and

the first step can be performed in n_1 ways,

the second step can be performed in n_2 ways *[regardless of how the first step was performed]*,

$\vdots$

the kth step can be performed in n_k ways *[regardless of how the preceding steps were performed]*,

then the entire operation can be performed in $n_1 n_2 \cdots n_k$ ways.

Example 9.2.2 – *Counting Personal Identification Numbers (PINs)*

A certain personal identification number (PIN) is required to be a sequence of any four symbols chosen from the 26 uppercase letters in the Roman alphabet and the ten digits.

a. How many different PINs are possible if repetition of symbols is allowed?

b. How many different PINs are possible if repetition of symbols is not allowed?

c. What is the probability that a PIN does not have a repeated symbol assuming that all PINs are equally likely?

Example 9.2.2 – *Solution (1/4)*

a. Some possible PINs are RCAE, 3387, B92B, and so forth. You can think of forming a PIN as a four-step operation where each step involves placing a symbol into one of four positions, as shown below.

Example 9.2.2 – *Solution (2/4)* continued

Step 1: Choose a symbol to place in position 1.

Step 2: Choose a symbol to place in position 2.

Step 3: Choose a symbol to place in position 3.

Step 4: Choose a symbol to place in position 4.

There is a fixed number of ways to perform each step, namely 36, regardless of how preceding steps were performed. And so, by the multiplication rule, there are $36 \cdot 36 \cdot 36 \cdot 36 = 36^4$ = 1,679,616 PINs in all.

Example 9.2.2 – *Solution (3/4)* continued

b. Again think of forming a PIN as a four-step operation: Choose the first symbol, then the second, then the third, and then the fourth.

There are 36 ways to choose the first symbol, 35 ways to choose the second (since the first symbol cannot be used again), 34 ways to choose the third (since the first two symbols cannot be reused), and 33 ways to choose the fourth (since the first three symbols cannot be reused).

Example 9.2.2 – *Solution (4/4)* continued

Thus, the multiplication rule can be applied to conclude that there are 36 · 35 · 34 · 33 = 1,413,720 different PINs with no repeated symbol.

c. By part (b) there are 1,413,720 PINs with no repeated symbol, and by part (a) there are 1,679,616 PINs in all. Thus the probability that a PIN chosen at random contains no repeated symbol is $\frac{1{,}413{,}720}{1{,}679{,}616} \cong 0.8417.$

In other words, approximately 84% of PINs have no repeated symbol.

Example 9.2.5 – *Counting the Number of Iterations of a Nested Loop*

Consider the following nested loop:

```
for i := 1 to 4
    for j := 1 to 3
        [Statements in body of inner loop.
        None contain branching statements
        that lead out of the inner loop.]
    next j
next i
```

How many times will the inner loop be iterated when the algorithm is implemented and run?

Example 9.2.5 – *Solution*

The outer loop is iterated four times, and during each iteration of the outer loop, there are three iterations of the inner loop. Hence by the multiplication rule, the total number of iterations of the inner loop is 4 · 3 = 12.

This is illustrated by the trace table below.

i	1 →			2 →			3 →			4 →		
j	1	2	3	1	2	3	1	2	3	1	2	3

$$\underbrace{\qquad\qquad}_{3} + \underbrace{\qquad\qquad}_{3} + \underbrace{\qquad\qquad}_{3} + \underbrace{\qquad\qquad}_{3} = 12$$

Permutations

Permutations (1/4)

A **permutation** of a set of objects is an ordering of the objects in a row. For example, the set of elements *a*, *b*, and *c* has six permutations.

abc *acb* *cba* *bac* *bca* *cab*

In general, given a set of *n* objects, how many permutations does the set have?

Permutations (2/4)

Imagine forming a permutation as an n-step operation:

Step 1: Choose an element to write first.

Step 2: Choose an element to write second.

$\vdots \quad \vdots$

Step n: Choose an element to write nth.

Any element of the set can be chosen in step 1, so there are n ways to perform step 1. Any element except that chosen in step 1 can be chosen in step 2, so there are $n - 1$ ways to perform step 2.

Permutations (3/4)

In general, the number of ways to perform each successive step is one less than the number of ways to perform the preceding step. At the point when the *n*th element is chosen, there is only one element left, so there is only one way to perform step *n.*

Hence, by the multiplication rule, there are

$$n(n-1)(n-2)\cdots 2\cdot 1 = n!$$

ways to perform the entire operation.

Permutations (4/4)

In other words, there are $n!$ permutations of a set of *n* elements. This reasoning is summarized in the following theorem. A formal proof uses mathematical induction and is left as an exercise.

Theorem 9.2.2

For any integer n with $n \geq 1$, the number of permutations of a set with n elements is $n!$.

Example 9.2.7 – *Permutations of the Letters in a Word*

a. How many ways can the letters in the word *COMPUTER* be arranged in a row?

b. How many ways can the letters in the word *COMPUTER* be arranged if the letters *CO* must remain next to each other (in order) as a unit?

c. If letters of the word *COMPUTER* are randomly arranged in a row, what is the probability that the letters *CO* remain next to each other (in order) as a unit?

Example 9.2.7 – *Solution (1/2)*

a. All eight letters in the word *COMPUTER* are distinct, so the number of ways in which you can arrange the letters equals the number of permutations of a set of eight elements. This equals $8! = 40{,}320$.

b. If the letter group *CO* is treated as a unit, then there are effectively only seven objects that are to be arranged in a row.

$\boxed{\text{CO}}\ \boxed{\text{M}}\ \boxed{\text{P}}\ \boxed{\text{U}}\ \boxed{\text{T}}\ \boxed{\text{E}}\ \boxed{\text{R}}$

Hence there are as many ways to write the letters as there are permutations of a set of seven elements, namely, $7! = 5{,}040$.

Example 9.2.7 – *Solution (2/2)*

continued

c. When the letters are arranged randomly in a row, the total number of arrangements is 40,320 by part (a), and the number of arrangements with the letters *CO* next to each other (in order) as a unit is 5,040.

Thus the probability is

$$\frac{5{,}040}{40{,}320} = \frac{1}{8} = 12.5\%.$$

Permutations of Selected Elements

Permutations of Selected Elements (1/2)

Given the set {*a*, *b*, *c*}, there are six ways to select two letters from the set and write them in order.

ab *ac* *ba* *bc* *ca* *cb*

Each such ordering of two elements of {*a*, *b*, *c*} is called a 2-*permutation* of {*a*, *b*, *c*}.

Definition

An ***r*-permutation** of a set of n elements is an ordered selection of r elements taken from the set of n elements. The number of r-permutations of a set of n elements is denoted $\boldsymbol{P(n, r)}$.

Permutations of Selected Elements (2/2)

Theorem 9.2.3

If n and r are integers and $1 \leq r \leq n$, then the number of r-permutations of a set of n elements is given by the formula

$$P(n, r) = n(n-1)(n-2)\cdots(n-r+1) \quad \text{first version}$$

or, equivalently,

$$P(n, r) = \frac{n!}{(n-r)!} \quad \text{second version.}$$

Example 9.2.9 – *Evaluating r-Permutations*

a. Evaluate $P(5, 2)$.

b. How many 4-permutations are there of a set of seven objects?

c. How many 5-permutations are there of a set of five objects?

Example 9.2.9 – *Solution (1/2)*

a. $P(5, 2) = \dfrac{5!}{(5-2)!} = \dfrac{5 \cdot 4 \cdot \cancel{3!}}{\cancel{3!}} = 20$

b. The number of 4-permutations of a set of seven objects is

$$P(7, 4) = \frac{7!}{(7-4)!} = \frac{7 \cdot 6 \cdot 5 \cdot 4 \cdot \cancel{3!}}{\cancel{3!}}$$

$$= 7 \cdot 6 \cdot 5 \cdot 4 = 840.$$

Example 9.2.9 – *Solution (2/2)*

continued

c. The number of 5-permutations of a set of five objects is

$$P(5, 5) = \frac{5!}{(5-5)!} = \frac{5!}{0!} = \frac{5!}{1} = 5! = 120.$$

Note that the definition of $0!$ as 1 makes this calculation come out as it should, for the number of 5-permutations of a set of five objects is certainly equal to the number of permutations of the set.

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.3 Counting Elements of Disjoint Sets: The Addition Rule

Copyright © Cengage Learning. All rights reserved.

Counting Elements of Disjoint Sets: The Addition Rule (1/2)

In this section we look at counting problems that can be solved by counting the number of elements in the union of two sets, the difference of two sets, or the intersection of two sets.

The basic rule underlying the calculation of the number of elements in a union or difference or intersection is the addition rule.

Counting Elements of Disjoint Sets: The Addition Rule (2/2)

This rule states that the number of elements in a union of mutually disjoint finite sets equals the sum of the number of elements in each of the component sets.

Theorem 9.3.1 The Addition Rule

Suppose a finite set A equals the union of k distinct mutually disjoint subsets A_1, $A_2, \ldots, A_k$. Then

$$N(A) = N(A_1) + N(A_2) + \cdots + N(A_k).$$

Example 9.3.1 – *Counting the Number of Integers Divisible by 5*

How many three-digit integers (integers from 100 to 999 inclusive) are divisible by 5?

Example 9.3.1 – *Solution (1/3)*

The solution to this problem uses the addition rule. Integers that are divisible by 5 end either in 5 or in 0. Thus the set of all three-digit integers that are divisible by 5 can be split into two mutually disjoint subsets A_1 and A_2 as shown in Figure 9.3.1.

Figure 9.3.1

Example 9.3.1 – *Solution (2/3)* continued

Now there are as many three-digit integers that end in 0 as there are possible choices for the left-most and middle digits (because the right-most digit must be a 0).

As illustrated below, there are nine choices for the left-most digit (the digits 1 through 9) and ten choices for the middle digit (the digits 0 through 9). Hence $N(A_1) = 9 \cdot 10 = 90$.

□	□	□
↑	↑	↑
9 choices	10 choices	number ends in 0
1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	

Example 9.3.1 – *Solution (3/3)* continued

Similar reasoning shows that there are as many three-digit integers that end in 5 as there are possible choices for the left-most and middle digits, which are the same as for the integers that end in 0.

Hence, $N(A_2) = 90$. So

$$\begin{bmatrix} \text{the number of} \\ \text{three-digit integers} \\ \text{that are divisible by 5} \end{bmatrix} = N(A_1) + N(A_2) = 90 + 90 = 180.$$

The Difference Rule

The Difference Rule (1/4)

An important consequence of the addition rule is the fact that if the number of elements in a set *A* and the number in a subset *B* of *A* are both known, then the number of elements that are in *A* and not in *B* can be computed.

Theorem 9.3.2 The Difference Rule

If A is a finite set and B is a subset of A, then

$$N(A - B) = N(A) - N(B).$$

The Difference Rule (2/4)

The difference rule is illustrated in Figure 9.3.2.

The Difference Rule

Figure 9.3.2

The difference rule holds for the following reason: If B is a subset of A, then the two sets B and $A - B$ have no elements in common and $B \cup (A - B) = A$.

The Difference Rule (3/4)

Hence, by the addition rule,

$$N(B) + N(A - B) = N(A).$$

Subtracting $N(B)$ from both sides gives the equation

$$N(A - B) = N(A) - N(B).$$

Example 9.3.2 – *Counting PINs with Repeated Symbols*

Consider the personal identification numbers (PINs). These are made from exactly four symbols chosen from the 26 uppercase letters of the Roman alphabet and the ten digits.

There are 1,679,616 PINs with repetition allowed and 265,896 PINs with no repeated symbol.

a. How many PINs contain at least one repeated symbol?

b. If all PINs are equally likely, what is the probability that a randomly chosen PIN contains at least one repeated symbol?

Example 9.3.2 – *Solution (1/4)*

a. Let S be the set of all the PINs with repetition allowed, and let A be the set of PINs with no repeated symbol. Then $S - A$ is the set of PINs with at least one repeated symbol, and, by the difference rule,

$$\begin{aligned} N(S - A) &= N(S) - N(A) \\ &= 1{,}679{,}616 - 1{,}413{,}720 \\ &= 265{,}896. \end{aligned}$$

Hence, there are 265,896 PINs that contain at least one repeated symbol.

Example 9.3.2 – *Solution (2/4)* continued

b. *Solution* 1: Because there are 1,679,616 PINs in all and 265,896 of these contain at least one repeated symbol, by the equally likely probability formula, the probability that a randomly chosen PIN contains a repeated symbol is

$$\frac{265{,}896}{1{,}679{,}616} \cong 0.158 = 15.8\%.$$

Solution 2: $P(A)$ is the probability that a randomly chosen PIN has no repeated symbol, and so $P(S - A)$ is the probability that a randomly chosen PIN has at least one repeated symbol.

Example 9.3.2 – *Solution (3/4)* continued

Then

$$P(S-A) = \frac{N(S-A)}{N(S)}$$ by definition of probability in the equally likely case

$$= \frac{N(S)-N(A)}{N(S)}$$ by the difference rule

$$= \frac{N(S)}{N(S)} - \frac{N(A)}{N(S)}$$ by the laws of fractions

$$= 1 - P(A)$$ by definition of probability in the equally likely case

Example 9.3.2 – *Solution (4/4)*

continued

$\cong 1 - 0.842$ by Example 9.2.2

$\cong 0.158$

$= 15.8\%.$

The Difference Rule (4/4)

Formula for the Probability of the Complement of an Event

If S is a finite sample space and A is an event in S, then

$$P(A^c) = 1 - P(A),$$

where $A^c = S - A$, the complement of A in S.

Example 9.3.3 – *Passwords with 3-5 Letters*

A certain computer access password consists of 3 through 5 uppercase letters chosen from the 26 letters in the Roman alphabet, with repetitions allowed.

a. How many different passwords are possible?

b. How many different passwords have no repeated letter?

c. How many different passwords have at least one repeated letter?

d. If all passwords are equally likely, what is the probability that a randomly chosen password has at least one repeated letter?

Example 9.3.3 – *Solution (1/6)*

a. The set of all passwords can be partitioned into three subsets consisting of passwords with lengths 3, 4, and 5, as shown in Figure 9.3.3.

Figure 9.3.3

Example 9.3.3 – *Solution (2/6)* continued

By the addition rule, the total number of passwords equals the number with length 3, plus the number with length 4, plus the number with length 5.

The multiplication rule can be used to compute the number of passwords of each length. Thus the

$$\text{number of passwords with length } 3 = 26^3$$

because forming such a password can be thought of as a three-step process with 26 ways to perform each step

Example 9.3.3 – *Solution (3/6)*

continued

$$\text{number of passwords with length } 4 = 26^4$$

because forming such a password can be thought of as a four-step process with 26 ways to perform each step

$$\text{number of passwords with length } 5 = 26^5$$

because forming such a password can be thought of as a five-step process with 26 ways to perform each step.

Hence the total number of passwords is

$$26^3 + 26^4 + 26^5 = 12{,}355{,}928.$$

Example 9.3.3 – *Solution (4/6)* continued

b. Constructing a password with length 3 and no repeated letter is a three-step process with 26 choices for step 1, 25 choices for step 2, and 24 choices for step 3.

Thus there are $26 \cdot 25 \cdot 24$ passwords with length three and no repeated letter.

Similarly, there are $26 \cdot 25 \cdot 24 \cdot 23$ passwords with length 4 and no repeated letter and $26 \cdot 25 \cdot 24 \cdot 23 \cdot 22$ passwords with length 5 and no repeated letter.

Example 9.3.3 – *Solution (5/6)* continued

Hence the total number of passwords with no repeated letter is

$$26 \cdot 25 \cdot 24 + 26 \cdot 25 \cdot 24 \cdot 23 + 26 \cdot 25 \cdot 24 \cdot 23 \cdot 22$$
$$= 8{,}268{,}000.$$

c. By part (a) the total number of passwords is 12,355,928, and by part (b) 8,268,000 of these passwords do not have a repeated letter. Thus, by difference rule, the number of passwords with at least one repeated letter is 4,087,928.

Example 9.3.3 – *Solution (6/6)*

continued

d. Given the assumption that all passwords are equally likely, the equally likely probability formula can be used. So the probability that a randomly chosen password has at least one repeated letter is

$$\frac{\begin{bmatrix}\text{\# of passwords with} \\ \text{no repeated letter}\end{bmatrix}}{\text{total \# of passwords}} = \frac{4{,}087{,}928}{12{,}355{,}928} \cong 33.1\%.$$

The Inclusion/Exclusion Rule

The Inclusion/Exclusion Rule (1/2)

The addition rule says how many elements are in a union of sets if the sets are mutually disjoint. Now consider the question of how to determine the number of elements in a union of sets when some of the sets overlap. For simplicity, begin by looking at a union of two sets *A* and *B*, as shown in Figure 9.3.5.

Figure 9.3.5

The Inclusion/Exclusion Rule (2/2)

Theorem 9.3.3 The Inclusion/Exclusion Rule for Two or Three Sets

If A, B, and C are any finite sets, then

$$N(A \cup B) = N(A) + N(B) - N(A \cap B)$$

and

$$N(A \cup B \cup C) = N(A) + N(B) + N(C) - N(A \cap B) - N(A \cap C) - N(B \cap C) + N(A \cap B \cap C).$$

Example 9.3.6 – *Counting Elements of a General Union*

a. How many integers from 1 through 1,000 are multiples of 3 or multiples of 5?

b. How many integers from 1 through 1,000 are neither multiples of 3 nor multiples of 5?

Example 9.3.6 – *Solution (1/6)*

a. Let A = the set of all integers from 1 through 1,000 that are multiples of 3.

Let B = the set of all integers from 1 through 1,000 that are multiples of 5.

Then

$A \cup B$ = the set of all integers from 1 through 1,000 that are multiples of 3 or multiples of 5

and

$A \cap B$ = the set of all integers from 1 through 1,000 that are multiples of both 3 and 5

Example 9.3.6 – *Solution (2/6)* continued

= the set of all integers from 1 through 1,000 that are multiples of 15.

[Now calculate $N(A)$, $N(B)$, and $N(A \cap B)$ and use the inclusion/exclusion rule to solve for $N(A \cup B)$.]

Because every third integer from 3 through 999 is a multiple of 3, each can be represented in the form $3k$, for some integer k from 1 through 333.

Example 9.3.6 – *Solution (3/6)* continued

Hence there are 333 multiples of 3 from 1 through 1,000, and so $N(A) = 333$.

$$\begin{array}{ccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & \ldots & 996 & 997 & 998 & 999 \\ & & \updownarrow & & & \updownarrow & & \updownarrow & & & \updownarrow \\ & & 3\cdot 1 & & & 3\cdot 2 & & 3\cdot 332 & & & 3\cdot 333 \end{array}$$

Similarly, each multiple of 5 from 1 through 1,000 has the form $5k$, for some integer k from 1 through 200.

$$\begin{array}{ccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \ldots & 995 & 996 & 997 & 998 & 999 & 1{,}000 \\ & & & & \updownarrow & & & & & \updownarrow & & \updownarrow & & & & & \updownarrow \\ & & & & 5\cdot 1 & & & & & 5\cdot 2 & & 5\cdot 199 & & & & & 5\cdot 200 \end{array}$$

Example 9.3.6 – *Solution (4/6)* continued

Thus there are 200 multiples of 5 from 1 through 1,000 and $N(B) = 200$. Finally, each multiple of 15 from 1 through 1,000 has the form $15k$, for some integer k from 1 through 66 (since $990 = 66 \cdot 15$).

$$\begin{array}{ccccccccccccc} 1 & 2 & \ldots 15 & \ldots & 30 & \ldots & 975 & \ldots & 990 & \ldots & 999 & 1{,}000 \\ & & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow & & & \\ & & 15 \cdot 1 & & 15 \cdot 2 & & 15 \cdot 65 & & 15 \cdot 66 & & & \end{array}$$

Hence there are 66 multiples of 15 from 1 through 1,000, and $N(A \cap B) = 66$.

Example 9.3.6 – *Solution (5/6)* continued

It follows by the inclusion/exclusion rule that

$$N(A \cup B) = N(A) + N(B) + N(A \cap B)$$

$$= 333 + 200 - 66$$

$$= 467.$$

Thus, 467 integers from 1 through 1,000 are multiples of 3 or multiples of 5.

Example 9.3.6 – *Solution (6/6)* continued

b. There are 1,000 integers from 1 through 1,000, and by part (a), 467 of these are multiples of 3 or multiples of 5.

Thus, by the set difference rule, there are 1,000 − 467 = 533 that are neither multiples of 3 nor multiples of 5.

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.4 The Pigeonhole Principle

Copyright © Cengage Learning. All rights reserved.

The Pigeonhole Principle (1/2)

The pigeonhole principle states that if n pigeons fly into m pigeonholes and $n > m$, then at least one hole must contain two or more pigeons. This principle is illustrated in Figure 9.4.1 for $n = 5$ and $m = 4$.

Figure 9.4.1

The Pigeonhole Principle (2/2)

Illustration (a) shows the pigeons perched next to their holes, and (b) shows the correspondence from pigeons to pigeonholes. The pigeonhole principle is sometimes called the *Dirichlet box principle* because it was first stated formally by J. P. G. L. Dirichlet (1805–1859).

Pigeonhole Principle

A function from one finite set to a smaller finite set cannot be one-to-one: There must be at least two elements in the domain that have the same image in the co-domain.

Example 9.4.1 – *Applying the Pigeonhole Principle*

a. In a group of six people, must there be at least two who were born in the same month? In a group of thirteen people, must there be at least two who were born in the same month? Why?

b. Among the residents of New York City, must there be at least two people with the same number of hairs on their heads? Why?

Example 9.4.1 – *Solution (1/5)*

a. A group of six people need not contain two who were born in the same month. For instance, the six people could have birthdays in each of the six months January through June.

A group of 13 people, however, must contain at least two who were born in the same month, for there are only 12 months in a year and 13 > 12.

To get at the essence of this reasoning, think of the thirteen people as the pigeons and the twelve months of the year as the pigeonholes.

Example 9.4.1 – *Solution (2/5)* continued

Denote the thirteen people by the symbols $x_1, x_2, \ldots, x_{13}$ and define a function B from the set of people to the set of twelve months as shown in the following arrow diagram.

Example 9.4.1 – *Solution (3/5)* continued

The pigeonhole principle says that no matter what the particular assignment of months to people, there must be at least two arrows pointing to the same month. Thus at least two people must have been born in the same month.

b. The answer is yes. In this example the pigeons are the people of New York City and the pigeonholes are all possible numbers of hairs on any individual's head.

Call the population of New York City P. It is known that P is at least 8,000,000.

Example 9.4.1 – *Solution (4/5)* continued

Also, the maximum number of hairs on any person's head is known to be less than 300,000. Define a function H from the set of people in New York City $\{x_1, x_2, \ldots, x_p\}$ to the set $\{0, 1, 2, 3, \ldots, 300{,}000\}$, as shown in the arrow diagram.

Example 9.4.1 – *Solution (5/5)* continued

Since the number of people in New York City is larger than the number of possible hairs on their heads, the function *H* is not one-to-one; at least two arrows point to the same number. And this means that at least two people have the same number of hairs on their heads.

Application to Decimal Expansions of Fractions

Application to Decimal Expansions of Fractions (1/1)

One important consequence of the pigeonhole principle is the fact that

> *the decimal expansion of any rational number either terminates or repeats.*

A terminating decimal is one like 3.625, and a repeating decimal is one like

$$2.38\overline{246},$$

where the bar over the digits 246 means that these digits are repeated forever.

Example 9.4.4 – *The Decimal Expansion of a Fraction (1/4)*

Let *a* and *b* be integers and consider a fraction *a/b*, where for simplicity *a* and *b* are both assumed to be positive.

The decimal expansion of *a/b* is obtained by dividing *a* by *b* as illustrated here for *a* = 3 and *b* = 14.

Example 9.4.4 – *The Decimal Expansion of a Fraction (2/4)*

continued

Let $r_0 = a$ and let $r_1, r_2, r_3, \ldots$ be the successive remainders obtained in the long division of a by b. By the quotient-remainder theorem, each remainder must be between 0 and $b - 1$.

(In this example, a is 3 and b is 14, and so the remainders are from 0 to 13.) If some remainder $r_i = 0$, then the division terminates and a/b has a terminating decimal expansion. If no $r_i = 0$, then the division process and hence the sequence of remainders continues forever.

Example 9.4.4 – *The Decimal Expansion of a Fraction (3/4)*

continued

By the pigeonhole principle, since there are more remainders than values that the remainders can take, some remainder value must repeat: $r_j = r_k$, for some indices j and k with $j < k$. This is illustrated below for $a = 3$ and $b = 14$.

Example 9.4.4 – *The Decimal Expansion of a Fraction (4/4)*

continued

If follows that the decimal digits obtained from the divisions between r_j and r_{k-1} repeat forever.

In the case of 3/14, the repetition begins with $r_7 = 2 = r_i$ and the decimal expansion repeats the quotients obtained from the divisions from r_1 through r_6 forever:

$$3/14 = 0.2\overline{142857}.$$

Generalized Pigeonhole Principle

Generalized Pigeonhole Principle (1/4)

A generalization of the pigeonhole principle states that if n pigeons fly into m pigeonholes and, for some positive integer k, $km < n$, then at least one pigeonhole contains $k + 1$ or more pigeons.

Generalized Pigeonhole Principle (2/4)

This is illustrated in Figure 9.4.2 for $m = 4$, $n = 9$, and $k = 2$.

Figure 9.4.2

Since $2 \cdot 4 < 9$, at least one pigeonhole contains three (2 + 1) or more pigeons. (In this example, pigeonhole 3 contains three pigeons.)

Generalized Pigeonhole Principle (3/4)

Generalized Pigeonhole Principle

For any function f from a finite set X with n elements to a finite set Y with m elements and for any positive integer k, if $km < n$, then there is some $y \in Y$ such that y is the image of at least $k + 1$ distinct elements of X.

Example 9.4.5 – *Applying the Generalized Pigeonhole Principle*

Show how the generalized pigeonhole principle implies that in a group of 85 people, at least 4 must have the same last initial.

Example 9.4.5 – *Solution (1/2)*

In this example the pigeons are the 85 people and the pigeonholes are the 26 possible last initials of their names.

Consider the function L from people to initials defined by the following arrow diagram.

Example 9.4.5 – *Solution (2/2)*

continued

Since $3 \cdot 26 = 78 < 85$, the generalized pigeonhole principle states that some initial must be the image of at least four (3 + 1) people. Thus at least four people have the same last initial.

Generalized Pigeonhole Principle (4/4)

Consider the following contrapositive form of the generalized pigeonhole principle.

Generalized Pigeonhole Principle (Contrapositive Form)

For any function f from a finite set X with n elements to a finite set Y with m elements and for any positive integer k, if for each $y \in Y$, $f^{-1}(y)$ has at most k elements, then X has at most km elements; in other words, $n \leq km$.

Proof of the Pigeonhole Principle

Proof of the Pigeonhole Principle (1/3)

The truth of the pigeonhole principle depends essentially on the sets involved being finite. We know that a set is called **finite** if, and only if, it is the empty set or there is a one-to-one correspondence from $\{1, 2, \ldots, n\}$ to it, where n is a positive integer.

In the first case the **number of elements** in the set is said to be 0, and in the second case it is said to be n. A set that is not finite is called **infinite**.

Proof of the Pigeonhole Principle (2/3)

Thus any finite set is either empty or can be written in the form $\{x_1, x_2, \ldots, x_n\}$ where n is a positive integer.

Theorem 9.4.1 The Pigeonhole Principle

For any function f from a finite set X with n elements to a finite set Y with m elements, if $n > m$, then f is not one-to-one.

Proof of the Pigeonhole Principle (3/3)

An important theorem that follows from the pigeonhole principle states that a function from one finite set to another finite set of the same size is one-to-one if, and only if, it is onto.

This result does not hold for infinite sets.

Theorem 9.4.2 One-to-One and Onto for Finite Sets

Let X and Y be finite sets with the same number of elements and suppose f is a function from X to Y. Then f is one-to-one if, and only if, f is onto.

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.5 Counting Subsets of a Set: Combinations

Copyright © Cengage Learning. All rights reserved.

Counting Subsets of a Set: Combinations (1/6)

The number of subsets of size *r* that can be chosen from *S* equals the number of subsets of size *r* that *S* has. Each individual subset of size *r* is called an *r-combination* of the set.

Definition *r*-combination

Let n and r be nonnegative integers with $r \leq n$. An ***r*-combination** of a set of n elements is a subset of r of the n elements.

Notation $\binom{n}{r}$

The symbol $\binom{n}{r}$, read "n choose r," denotes the number of subsets of size r (or r-combinations) that can be formed from a set of n elements.

Example 9.5.1 – *3-Combinations*

Consider a set consisting of 4 people: Ann, Bob, Cyd, and Dan.

a. Given the set {Ann, Bob, Cyd, and Dan}, each subset of size 3 is a 3-combination of its elements. List all the 3-combinations of the set.

b. Use the result of part (a) to find $\binom{4}{3}$.

c. In how many ways can the people in the set form a committee of size 3?

Example 9.5.1 – *Solution (1/2)*

a. Each 3-combination (subset of size 3) can be obtained by leaving out one of the elements of the set:

{Bob, Cyd, Dan} leave out Ann

{Ann, Cyd, Dan} leave out Bob

{Ann, Bob, Dan} leave out Cyd

{Ann, Bob, Cyd} leave out Dan.

b. There are 4 items in the list of 3-combinations in part (a).

So, $\binom{4}{3} = 4.$

Example 9.5.1 – *Solution (2/2)* continued

c. The number of ways for the people in the set to form a committee of size 3 is the number of distinct such committees, which is the same as the number of subsets of size 3 and equals the number of 3-combinations of elements in the set.

Thus there are 4 ways the people in the set can form a committee of size 3.

Counting Subsets of a Set: Combinations (2/6)

There are two distinct methods that can be used to select *r* objects from a set of *n* elements. In an **ordered selection**, it is not only what elements are chosen but also the order in which they are chosen that matters.

Two ordered selections are said to be the same if the elements chosen are the same and also if the elements are chosen in the same order. An ordered selection of *r* elements from a set of *n* elements is an *r*-permutation of the set.

Counting Subsets of a Set: Combinations (3/6)

In an **unordered selection**, on the other hand, it is only the identity of the chosen elements that matters. Two unordered selections are said to be the same if they consist of the same elements, regardless of the order in which the elements are chosen.

An unordered selection of r elements from a set of n elements is the same as a subset of size r or an r-combination of the set.

Example 9.5.2 – *Unordered Selections*

How many unordered selections of two elements can be made from the set {0, 1, 2, 3}?

Example 9.5.2 – *Solution*

An unordered selection of two elements from {0, 1, 2, 3} is the same as a 2-combination, or subset of size 2, taken from the set. These can be listed systematically:

{0, 1}, {0, 2}, {0, 3} subsets containing 0

{1, 2}, {1, 3} subsets containing 1 but not already listed

{2, 3} subsets containing 2 but not already listed.

Since this listing exhausts all possibilities, there are six subsets in all. Thus $\binom{4}{2} = 6,$ which is the number of unordered selections of two elements from a set of four.

Counting Subsets of a Set: Combinations (4/6)

When the values of *n* and *r* are small, it is reasonable to calculate values of $\binom{n}{r}$ using the method of **complete enumeration** (listing all possibilities). But when *n* and *r* are large, it is not feasible to compute these numbers by listing and counting all possibilities.

The general values of $\binom{n}{r}$ can be found by a somewhat indirect but simple method. An equation is derived that contains $\binom{n}{r}$ as a factor. Then this equation is solved to obtain a formula for $\binom{n}{r}$.

Example 9.5.3 – *Relation between Permutations and Combinations*

Write all 2-permutations of the set {0, 1, 2, 3}. Find an equation relating the number of 2-permutations, $P(4, 2)$, and the number of 2-combinations, $\binom{4}{2}$, and solve this equation for $\binom{4}{2}$.

Example 9.5.3 – *Solution (1/5)*

According to Theorem 9.2.3, the number of 2-permutations of the set {0, 1, 2, 3} is $P(4, 2)$, which equals

$$\frac{4!}{(4-2)!} = \frac{4 \cdot 3 \cdot \cancel{2!}}{\cancel{2!}} = 12.$$

Theorem 9.2.3

If n and r are integers and $1 \leq r \leq n$, then the number of r-permutations of a set of n elements is given by the formula

$$P(n, r) = n(n-1)(n-2)\cdots(n-r+1) \quad \text{first version}$$

or, equivalently,

$$P(n, r) = \frac{n!}{(n-r)!} \quad \text{second version.}$$

Example 9.5.3 – *Solution (2/5)* continued

Now the act of constructing a 2-permutation of {0, 1, 2, 3} can be thought of as a two-step process:

Step 1: Choose a subset of two elements from {0, 1, 2, 3}.

Step 2: Choose an ordering for the two-element subset.

Example 9.5.3 – *Solution (3/5)* continued

This process can be illustrated by the possibility tree shown in Figure 9.5.1.

Relation between Permutations and Combinations

Figure 9.5.1

Example 9.5.3 – *Solution (4/5)* continued

The number of ways to perform step 1 is $\binom{4}{2}$, the same as the number of subsets of size 2 that can be chosen from {0, 1, 2, 3}. The number of ways to perform step 2 is $2!$, the number of ways to order the elements in a subset of size 2. Because the number of ways of performing the whole process is the number of 2-permutations of the set {0, 1, 2, 3}, which equals $P(4, 2)$, it follows from the product rule that

$$P(4, 2) = \binom{4}{2} \cdot 2!.$$

This is an equation that relates $P(4, 2)$ and $\binom{4}{2}$.

Example 9.5.3 – *Solution (5/5)*

continued

Solving the equation for $\binom{4}{2}$ gives

$$\binom{4}{2} = \frac{P(4, 2)}{2!}$$

We know that $P(4, 2) = \frac{4!}{(4-2)!}$ So, substituting yields

$$\binom{4}{2} = \frac{\frac{4!}{(4-2)!}}{2!} = \frac{4!}{2!(4-2)!} = 6.$$

Counting Subsets of a Set: Combinations (5/6)

Theorem 9.5.1 Computational Formula for $\binom{n}{r}$

The number of subsets of size r (or r-combinations) that can be chosen from a set of n elements, $\binom{n}{r}$, is given by the formula

$$\binom{n}{r} = \frac{P(n, r)}{r!} \qquad \text{first version}$$

or, equivalently,

$$\binom{n}{r} = \frac{n!}{r!(n-r)!} \qquad \text{second version}$$

where n and r are nonnegative integers with $r \leq n$.

Example 9.5.4 – *Calculating the Number of Teams*

Consider again the problem of choosing five members from a group of twelve to work as a team on a special project. How many distinct five-person teams can be chosen?

Example 9.5.4 – *Solution*

The number of distinct five-person teams is the same as the number of subsets of size 5 (or 5-combinations) that can be chosen from the set of 12. This number is $\binom{12}{5}$.

By Theorem 9.5.1,

$$\binom{12}{5} = \frac{12!}{5!(12-5)!} = \frac{\cancel{12}\cdot 11\cdot \cancel{10}\cdot 9\cdot 8\cdot \cancel{7!}}{(\cancel{5}\cdot \cancel{4}\cdot \cancel{3}\cdot \cancel{2}\cdot 1)\cdot \cancel{7!}} = 11\cdot 9\cdot 8 = 792.$$

Thus there are 792 distinct five-person teams.

Example 9.5.8 – *An Application to Graphs*

a. Use the multiplication rule to show that $K_{m,n}$, a complete bipartite graph on (m, n) vertices, has mn edges.

b. Use 2-combinations to show that K_n, a complete graph on n vertices, has $\frac{n(n-1)}{2}$ edges.

Example 9.5.8 – *Solution (1/4)*

a. A complete bipartite graph on (m, n) vertices, K_{mn}, is a simple graph whose vertices can be divided into two distinct, nonoverlapping sets—say, V with m vertices and W with n vertices—in such a way that there is exactly one edge from each vertex of V to each vertex of W, there is no edge from any one vertex of V to any other vertex of V, and there is no edge from any one vertex of W to any other vertex of W. Label the m vertices of V as $v_1, v_2, \ldots, v_m$.

Example 9.5.8 – *Solution (2/4)* continued

Think of constructing the edges between the vertices of V and the vertices of W as an m-step process: for each integer k from 1 through m, draw exactly one edge from v_k to each of the n edges of W.

Because each step can be performed in n ways and there are m steps, by the multiplication rule there are mn ways to construct all the edges. Thus $K_{m,n}$ has mn edges.

Example 9.5.8 – *Solution (3/4)* continued

b. A complete graph on n vertices, K_n, is a simple graph with n vertices and exactly one edge between each pair of vertices. If $n = 1$, then K_n has one vertex and 0 edges, and since $\frac{n(n-1)}{2} = \frac{1(1-1)}{2} = 0$, then K_n has $\frac{n(n-1)}{2}$ edges.

If $n \geq 2$, then, since any two distinct vertices of K_n are connected by exactly one edge, there are as many edges in K_n as there are subsets of size two in the set of n vertices.

Example 9.5.8 – *Solution (4/4)*

continued

By Theorem 9.5.1, there are $\binom{n}{2}$ such sets, and

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$$

Hence K_n has $\frac{n(n-1)}{2}$ edges.

Example 9.5.9 – *Poker Hand Problems (1/3)*

The game of poker is played with an ordinary deck of 52 cards (see Example 9.1.1). Various five-card holdings are given special names, and certain holdings beat certain other holdings. The named holdings are listed from highest to lowest below.

Royal flush: 10, J, Q, K, A of the same suit

Straight flush: five adjacent denominations of the same suit but not a royal flush—aces can be high or low, so A, 2, 3, 4, 5 of the same suit is a straight flush

Example 9.5.9 – *Poker Hand Problems (2/3)* continued

Four of a kind: four cards of one denomination—the fifth card can be any other in the deck

Full house: three cards of one denomination, two cards of another denomination

Flush: five cards of the same suit but not a straight or a royal flush

Straight: five cards of adjacent denominations but not all of the same suit—aces can be high or low

Three of a kind: three cards of the same denomination and two other cards of different denominations

Example 9.5.9 – *Poker Hand Problems (3/3)*

continued

Two pairs: two cards of one denomination, two cards of a second denomination, and a fifth card of a third denomination

One pair: two cards of one denomination and three other cards all of different denominations

No pairs: all cards of different denominations but not a straight, or straight flush, or flush, or royal flush

a. How many five-card poker hands contain two pairs?

b. If a five-card hand is dealt at random from an ordinary deck of cards, what is the probability that the hand contains two pairs?

Example 9.5.9 – *Solution (1/3)*

a. Consider forming a hand with two pairs as a four-step process:

Step 1: Choose the two denominations for the pairs.
Step 2: Choose two cards from the smaller denomination.

Step 3: Choose two cards from the larger denomination.
Step 4: Choose one card from those remaining.

The number of ways to perform step 1 is $\binom{13}{2}$ because there are 13 denominations in all. The number of ways to perform each of steps 2 and 3 is $\binom{4}{2}$ because there are four cards of each denomination, one in each suit.

Example 9.5.9 – *Solution (2/3)*

continued

The number of ways to perform step 4 is $\binom{44}{1}$ because the fifth card is chosen from the eleven denominations not included in the pair and there are four cards of each denomination. Thus

$$\begin{bmatrix}\text{the total number of}\\ \text{hands with two pairs}\end{bmatrix} = \binom{13}{2}\binom{4}{2}\binom{4}{2}\binom{44}{1}$$

$$= \frac{13!}{2!(13-2)!}\cdot\frac{4!}{2!(4-2)!}\cdot\frac{4!}{2!(4-2)!}\cdot\frac{44!}{1!(44-1)!}$$

$$= \frac{13\cdot 12\cdot 11!}{(2\cdot 1)\cdot 11!}\cdot\frac{4\cdot 3\cdot 2!}{(2\cdot 1)\cdot 2!}\cdot\frac{4\cdot 3\cdot 2!}{(2\cdot 1)\cdot 2!}\cdot\frac{44\cdot 43!}{1\cdot 43!}$$

$$= 78\cdot 6\cdot 6\cdot 44 = 123{,}552.$$

Example 9.5.9 – *Solution (3/3)*

continued

b. The total number of five-card hands from an ordinary deck of cards is $\binom{52}{5}$ = 2,598,960. Thus if all hands are equally likely, the probability of obtaining a hand with two pairs is $\frac{123{,}552}{2{,}598{,}960} \cong 4.75\%$.

Example 9.5.11 – *Permutations of a Set with Repeated Elements*

Consider various ways of ordering the letters in the word *MISSISSIPPI*:

IIMSSPISSIP, *ISSSPMIIPIS*, *PIMISSSSIIP*, and so on.

How many distinguishable orderings are there?

Example 9.5.11 – *Solution (1/4)*

This example generalizes Example 9.5.10. Imagine placing the 11 letters of *MISSISSIPPI* one after another into 11 positions.

Because copies of the same letter cannot be distinguished from one another, once the positions for a certain letter are known, then all copies of the letter can go into the positions in any order.

Example 9.5.11 – *Solution (2/4)* continued

It follows that constructing an ordering for the letters can be thought of as a four-step process:

Step 1: Choose a subset of four positions for the *S*'s.

Step 2: Choose a subset of four positions for the *I*'s.

Step 3: Choose a subset of two positions for the *P*'s.

Step 4: Choose a subset of one position for the *M*.

Since there are 11 positions in all, there are $\binom{11}{4}$ subsets of four positions for the *S*'s.

Example 9.5.11 – *Solution (3/4)* continued

Once the four *S*'s are in place, there are seven positions that remain empty, so there are $\binom{7}{4}$ subsets of four positions for the *I*'s. After the *I*'s are in place, there are three positions left empty, so there are $\binom{3}{2}$ subsets of two positions for the *P*'s. That leaves just one position for the *M*. But $1 = \binom{1}{1}$.

Hence by the multiplication rule,

$$\begin{bmatrix} \text{number of ways to} \\ \text{position all the letters} \end{bmatrix} = \binom{11}{4}\binom{7}{4}\binom{3}{2}\binom{1}{1}$$

Example 9.5.11 – *Solution (4/4)* continued

$$= \frac{11!}{4! \cdot \cancel{7!}} \cdot \frac{\cancel{7!}}{4! \cdot \cancel{3!}} \cdot \frac{\cancel{3!}}{2! \cdot \cancel{1!}} \cdot \frac{\cancel{1!}}{1! \cdot \cancel{0!}}$$

$$= \frac{11!}{4! \cdot 4! \cdot 2! \cdot 1!}$$

$$= 34{,}650.$$

Counting Subsets of a Set: Combinations (6/6)

Theorem 9.5.2 Permutations with Sets of Indistinguishable Objects

Suppose a collection consists of n objects of which

n_1 are of type 1 and are indistinguishable from each other

n_2 are of type 2 and are indistinguishable from each other

$\vdots$

n_k are of type k and are indistinguishable from each other,

and suppose that $n_1 + n_2 + \cdots + n_k = n$. Then the number of distinguishable permutations of the n objects is

$$\binom{n}{n_1}\binom{n-n_1}{n_2}\binom{n-n_1-n_2}{n_3}\cdots\binom{n-n_1-n_2-\cdots-n_{k-1}}{n_k}$$
$$= \frac{n!}{n_1!n_2!n_3!\cdots n_k!}.$$

Some Advice about Counting

Example 9.5.12 – *Double Counting (1/3)*

A group consists of five men and seven women. How many teams of five contain at least one man?

Incorrect Solution

Imagine constructing the team as a two-step process:

Step 1: Choose a subset of one man from the five men.

Step 2: Choose a subset of four others from the remaining eleven people.

Hence, by the multiplication rule, there are $\binom{5}{1} \cdot \binom{11}{4} = 1{,}650$ five-person teams that contain at least one man.

Example 9.5.12 – *Double Counting (2/3)* continued

Analysis of the Incorrect Solution The problem with the solution above is that some teams are counted more than once. Suppose the men are Anwar, Ben, Carlos, Dwayne, and Ed and the women are Fumiko, Gail, Hui-Fan, Inez, Jill, Kim, and Laura.

According to the method described previously, one possible outcome of the two-step process is as follows:

Outcome of step 1: Anwar

Outcome of step 2: Ben, Gail, Inez, and Jill.

Example 9.5.12 – *Double Counting (3/3)* continued

In this case the team would be {Anwar, Ben, Gail, Inez, Jill}. But another possible outcome is

Outcome of step 1: Ben

Outcome of step 2: Anwar, Gail, Inez, and Jill,

which also gives the team {Anwar, Ben, Gail, Inez, Jill}. Thus this one team is given by two different branches of the possibility tree, and so it is counted twice.

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.6 *r*-Combinations with Repetition Allowed

Copyright © Cengage Learning. All rights reserved.

r-Combinations with Repetition Allowed (1/2)

In this section we ask: How many ways are there to choose r elements without regard to order from a set of n elements *if repetition is allowed*? A good way to imagine this is to visualize the n elements as categories of objects from which multiple selections may be made.

Definition and Notation

An *r-combination with repetition allowed*, or *multiset of size r*, chosen from a set X of n elements is an unordered selection of elements taken from X with repetition allowed. If $X = \{x_1, x_2, \ldots, x_n\}$, we write an r-combination with repetition allowed, or multiset of size r, as $[x_{i_1}, x_{i_2}, \ldots, x_{i_r}]$ where each x_{i_j} is in X and some of the x_{i_j} may equal each other.

Example 9.6.1 – *r-Combinations with Repetition Allowed*

Write a complete list to find the number of 3-combinations with repetition allowed, or multisets of size 3, that can be selected from {1, 2, 3, 4}.

Observe that because the order in which the elements are chosen does not matter, the elements of each selection may be written in increasing order, and writing the elements in increasing order will ensure that no combinations are overlooked.

Example 9.6.1 – *Solution (1/2)*

[1, 1, 1]; [1, 1, 2]; [1, 1, 3]; [1, 1, 4]

all combinations that start with 1, 1

[1, 2, 2]; [1, 2, 3]; [1, 2, 4];

all additional combinations that start with 1, 2

[1, 3, 3]; [1, 3, 4]; [1, 4, 4];

all additional combinations that start with 1, 3 or 1, 4

[2, 2, 2]; [2, 2, 3]; [2, 2, 4];

all additional combinations that start with 2, 2

Example 9.6.1 – *Solution (2/2)* continued

[2, 3, 3]; [2, 3, 4]; [2, 4, 4];

all additional combinations that start with 2, 3 or 2, 4

[3, 3, 3]; [3, 3, 4]; [3, 4, 4];

all additional combinations that start with 3, 3 or 3, 4

[4, 4, 4] the only additional combination that starts with 4, 4.

Thus there are twenty 3-combinations with repetition allowed.

r-Combinations with Repetition Allowed (2/2)

Theorem 9.6.1

The number of r-combinations with repetition allowed (or multisets of size r) that can be selected from a set of n elements is

$$\binom{r+n-1}{r}.$$

This equals the number of ways r objects can be selected from n categories of objects with repetition allowed.

Example 9.6.2 – *Selecting 15 Cans of Soft Drinks of Five Different Types*

A person giving a party wants to set out 15 assorted cans of soft drinks for his guests. He shops at a store that sells five different types of soft drinks.

a. How many different selections of cans of 15 soft drinks can he make?

b. If root beer is one of the types of soft drink, how many different selections include at least six cans of root beer?

c. If the store has only five cans of root beer but at least 15 cans of each other type of soft drink, how many different selections are there?

Example 9.6.2 – *Solution (1/6)*

a. Think of the five different types of soft drinks as the n categories and the 15 cans of soft drinks to be chosen as the r objects (so $n = 5$ and $r = 15$).

Each selection of cans of soft drinks is represented by a string of $5 - 1 = 4$ vertical bars (to separate the categories of soft drinks) and 15 crosses (to represent the cans selected).

Example 9.6.2 – *Solution (2/6)* continued

For instance, the string

$$\times\ \times\ \times\ |\ \times\ \times\ \times\ \times\ \times\ \times\ \times\ |\ |\ \times\ \times\ \times\ |\ \times\ \times$$

represents a selection of three cans of soft drinks of type 1, seven of type 2, none of type 3, three of type 4, and two of type 5.

The total number of selections of 15 cans of soft drinks of the five types is the number of strings of 19 symbols, 5 − 1 = 4 of them $|$ and 15 of them $\times$:

$$\binom{15+5-1}{15} = \frac{19!}{15!\cdot(19-15)!} = \binom{19}{15} = \frac{19\cdot\overset{6}{\cancel{18}}\cdot 17\cdot\overset{2}{\cancel{16}}\cdot\cancel{15!}}{\cancel{15!}\cdot\cancel{4}\cdot\cancel{3}\cdot\cancel{2}\cdot 1} = 3{,}876.$$

Example 9.6.2 – *Solution (3/6)* continued

b. If at least six cans of root beer are to be included in the selection, you can imagine choosing six such cans first and then choosing nine additional cans. The choice of the nine additional cans can be represented as a string of 9 $\times$'s and 4 $|$'s.

For example, if root beer is type 1, then the string $\times\ \times\ \times\ |\ |\ \times\ \times\ |\ \times\ \times\ \times\ \times\ |$ represents a selection of three cans of root beer (in addition to the six chosen initially), none of type 2, two of type 3, four of type 4, and none of type 5.

Example 9.6.2 – *Solution (4/6)* continued

Thus the total number of selections of 15 cans of soft drinks of the five types, including at least six cans of root beer, is the number of strings of 13 symbols, 4 (= 5 − 1) of them | and 9 of them ×:

$$\binom{9+4}{9} = \frac{13!}{9!\cdot(13-9)!} = \binom{13}{9} = \frac{13\cdot\cancel{12}\cdot 11\cdot \overset{5}{\cancel{10}}\cdot\cancel{9!}}{\cancel{9!}\cdot\cancel{4}\cdot\cancel{3}\cdot\cancel{2}\cdot 1} = 715.$$

c. If the store has only five cans of root beer, then the number of different selections of 15 cans of soft drinks of the five types is the same as the number of different selections that contain five or fewer cans of root beer.

Example 9.6.2 – *Solution (5/6)* continued

Let T be the set of all possible selections assuming that there are at least 15 cans of each type, let $R_{\leq 5}$ be the set of selections from T that contain five or fewer cans of root beer, and let $R_{\geq 6}$ be the set of selections from T that contain six or more cans of root beer.

Then

$$T = R_{\leq 5} \cup R_{\geq 6} \quad \text{and} \quad R_{\leq 5} \cap R_{\geq 6} = \varnothing.$$

Example 9.6.2 – *Solution (6/6)* continued

By part (a) $N(T) = 3{,}876$ and by part (b) $N(R_{\geq 6}) = 715$. Thus, by the difference rule,

$$N(R_{\leq 5}) = N(T) - N(R_{\geq 6}) = 3{,}876 - 715 = 3{,}161.$$

So, when there are only five or fewer cans of root beer, the number of different selections of soft drinks is 3,161.

Example 9.6.3 – *Counting Triples (i, j, k) with* $1 \le i \le j \le k \le n$

If n is a positive integer, how many triples of integers from 1 through n can be formed in which the elements of the triple are written in increasing order but are not necessarily distinct?

In other words, how many triples of integers (i, j, k) are there with $1 \le i \le j \le k \le n$?

Example 9.6.3 – *Solution (1/2)*

Any triple of integers (i, j, k) with $1 \le i \le j \le k \le n$ can be represented as a string of $n - 1$ vertical bars and three crosses, with the positions of the crosses indicating which three integers from 1 to n are included in the triple.

The table below illustrates this for $n = 5$.

Category 1		2		3		4		5	Result of the Selection
	|		|	× ×	|		|	×	(3, 3, 5)
×	|	×	|		|		|		(1, 2, 4)

Example 9.6.3 – *Solution (2/2)*

continued

Thus the number of such triples is the same as the number of strings of $(n-1)$ $|$'s and 3 $\times$'s, which is

$$\binom{3+(n-1)}{3} = \binom{n+2}{3} = \frac{(n+2)!}{3!(n+2-3)!}$$

$$= \frac{(n+2)(n+1)n\cancel{(n-1)!}}{3!\cancel{(n-1)!}} = \frac{n(n+1)(n+2)}{6}.$$

Example 9.6.5 – *The Number of Integral Solutions of an Equation*

How many solutions are there to the equation

$$x_1 + x_2 + x_3 + x_4 = 10$$

if x_1, x_2, x_3, and x_4 are nonnegative integers?

Example 9.6.5 – *Solution (1/3)*

Think of the number 10 as divided into ten individual units and the variables x_1, x_2, x_3, and x_4 as four categories into which these units are placed.

The number of units in each category x_i indicates the value of x_i in a solution of the equation.

Each solution can, then, be represented by a string of three vertical bars (to separate the four categories) and ten crosses (to represent the ten individual units).

Example 9.6.5 – *Solution (2/3)* continued

For example, in the following table, the two crosses under x_1, five crosses under x_2, and three crosses under x_4 represent the solution $x_1 = 2$, $x_2 = 5$, $x_3 = 0$, and $x_4 = 3$.

Categories x_1	x_2	x_3	x_4	Solution to the Equation $x_1 + x_2 + x_3 + x_4 = 10$
× × |	× × × × × |	|	× × ×	$x_1 = 2,\quad x_2 = 5,\quad x_3 = 0,\quad$ and $\quad x_4 = 3$
× × × × |	× × × × × × |	|		$x_1 = 4,\quad x_2 = 6,\quad x_3 = 0,\quad$ and $\quad x_4 = 0$

Example 9.6.5 – *Solution (3/3)*

continued

Therefore, there are as many solutions to the equation as there are strings often crosses and three vertical bars, namely,

$$\binom{10+3}{10} = \binom{13}{10} = \frac{13!}{10!(13-10)!} = \frac{13 \cdot 12 \cdot 11 \cdot \cancel{10!}}{\cancel{10!} \cdot 3 \cdot 2 \cdot 1} = 286.$$

Remark: Deciding Which Formula to Use

Remark: Deciding Which Formula to Use (1/1)

Earlier we discussed four different ways of choosing *k* elements from *n.* The order in which the choices are made may or may not matter, and repetition may or may not be allowed. The following table summarizes which formula to use in which situation.

	Order Matters	**Order Does Not Matter**
Repetition Is Allowed	n^k	$\binom{k+n-1}{k}$
Repetition Is Not Allowed	$P(n, k)$	$\binom{n}{k}$

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.7 Pascal's Formula and the Binomial Theorem

Copyright © Cengage Learning. All rights reserved.

Pascal's Formula and the Binomial Theorem

In this section we derive several formulas for values of $\binom{n}{r}$. The most important is Pascal's formula, which is the basis for Pascal's triangle and is a crucial component of one of the proofs of the binomial theorem.

We offer two distinct proofs for both Pascal's formula and the binomial theorem. One of them is called "algebraic" because it relies to a great extent on algebraic manipulation, and the other is called "combinatorial," because it is based on the kind of counting arguments we have been discussing in this chapter.

Pascal’s Formula

Pascal's Formula (1/5)

Pascal's formula, named after the seventeenth-century French mathematician and philosopher Blaise Pascal, is one of the most famous and useful in combinatorics (which is the formal term for the study of counting and listing problems).

It relates the value of $\binom{n+1}{r}$ to the values of $\binom{n}{r-1}$ and $\binom{n}{r}$. Specifically, it says that

$$\binom{n+1}{r} = \binom{n}{r-1} + \binom{n}{r}$$

whenever n and r are positive integers with $r \leq n$.

Pascal's Formula (2/5)

This formula makes it easy to compute higher combinations in terms of lower ones: If all the values of $\binom{n}{r}$ are known, then the values of $\binom{n+1}{r}$ can be computed for every integer r such that $0 < r \leq n$.

Pascal's Formula (3/5)

Pascal's triangle, shown in Table 9.7.1, is a geometric version of Pascal's formula.

$n \backslash r$	0	1	2		3	4	5	…	$r-1$		r	…
0	1								·		·	…
1	1								·		·	…
2	1	2	1						·		·	…
3	1	3	3		1				·		·	…
4	1	4	6	+	4	1			·		·	…
5	1	5	10	=	10	5	1		·		·	…
⋮	⋮	⋮	⋮		⋮	⋮	⋮		⋮		⋮	⋮⋮⋮
n	$\binom{n}{0}$	$\binom{n}{1}$	$\binom{n}{2}$		$\binom{n}{3}$	$\binom{n}{4}$	$\binom{n}{5}$	…	$\binom{n}{r-1}$	+	$\binom{n}{r}$	…
$n+1$	$\binom{n+1}{0}$	$\binom{n+1}{1}$	$\binom{n+1}{2}$		$\binom{n+1}{3}$	$\binom{n+1}{4}$	$\binom{n+1}{5}$	…		=	$\binom{n+1}{r}$	…
·		·	·		·	·	·		·		·	…
·		·	·		·	·	·		·		·	…
·		·	·		·	·	·		·		·	…

Pascal's Triangle for Values of $\binom{n}{r}$

Table 9.7.1

Pascal’s Formula (4/5)

Sometimes it is simply called the arithmetic triangle because it was used centuries before Pascal by Chinese and Persian mathematicians.

But Pascal discovered it independently, and ever since 1654, when he published a treatise that explored many of its features, it has generally been known as Pascal’s triangle.

Example 9.7.3 – *Calculating* $\binom{n}{r}$ *Using Pascal's Triangle*

Use Pascal's triangle to compute the values of

$$\binom{6}{2} \quad \text{and} \quad \binom{6}{3}.$$

Example 9.7.3 – *Solution*

By construction, the value in row *n*, column *r* of Pascal's triangle is the value of $\binom{n}{r}$, for every pair of positive integers *n* and *r* with $r \le n$. By Pascal's formula, $\binom{n+1}{r}$ can be computed by adding together $\binom{n}{r-1}$ and $\binom{n}{r}$, which are located directly above and above left of $\binom{n+1}{r}$.

Thus,

$$\binom{6}{2} = \binom{5}{1} + \binom{5}{2} = 5 + 10 = 15 \quad \text{and}$$

$$\binom{6}{3} = \binom{5}{2} + \binom{5}{3} = 10 + 10 = 20.$$

Pascal's Formula (5/5)

Theorem 9.7.1 Pascal's Formula

Let n and r be positive integers with $r \leq n$. Then

$$\binom{n+1}{r} = \binom{n}{r-1} + \binom{n}{r}.$$

Example 9.7.4 – *Deriving New Formulas from Pascal's Formula*

Use Pascal's formula to derive a formula for $\binom{n+2}{r}$ in terms of values of $\binom{n}{r}$, $\binom{n}{r-1}$, and $\binom{n}{r-2}$.

Assume n and r are nonnegative integers and $2 \le r \le n$.

Example 9.7.4 – *Solution (1/2)*

By Pascal's formula,

$$\binom{n+2}{r} = \binom{n+1}{r-1} + \binom{n+1}{r}.$$

Now apply Pascal's formula to $\binom{n+1}{r-1}$ and $\binom{n+1}{r}$ and substitute into the above to obtain

$$\binom{n+2}{r} = \left[\binom{n}{r-2} + \binom{n}{r-1}\right] + \left[\binom{n}{r-1} + \binom{n}{r}\right].$$

Example 9.7.4 – *Solution (2/2)*

continued

Combining the two middle terms gives

$$\binom{n+2}{r} = \binom{n}{r-2} + 2\binom{n}{r-1} + \binom{n}{r}$$

for all nonnegative integers n and r such that $2 \le r \le n$.

The Binomial Theorem

The Binomial Theorem (1/2)

In algebra a sum of two terms, such as *a* + *b*, is called a **binomial**. The *binomial theorem* gives an expression for the powers of a binomial $(a+b)^n$, for each nonnegative integer *n* and all real numbers *a* and *b*.

Theorem 9.7.2 Binomial Theorem

Given any real numbers a and b and any nonnegative integer n,

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$$

$$= a^n + \binom{n}{1} a^{n-1} b^1 + \binom{n}{2} a^{n-2} b^2 + \cdots + \binom{n}{n-1} a^1 b^{n-1} + b^n.$$

The Binomial Theorem (2/2)

Note that the second expression equals the first because $\binom{n}{0} = 1$ and $\binom{n}{n} = 1$, for all nonnegative integers *n*, since $b^0 = 1$ and $a^{n-n} = 1$.

Definition

For any real number a and any nonnegative integer n, the **nonnegative integer powers of *a*** are defined as follows:

$$a^n = \begin{cases} 1 & \text{if } n = 0 \\ a \cdot a^{n-1} & \text{if } n > 0. \end{cases}$$

Example 9.7.5 – *Substituting into the Binomial Theorem*

Expand the following expressions using the binomial theorem:

a. $(a+b)^5$ b. $(x-4y)^4$

Example 9.7.5 – *Solution (1/2)*

a. $(a+b)^5 = \sum_{k=0}^{5} \binom{5}{k} a^{5-k} b^k$

$$= a^5 + \binom{5}{1} a^{5-1} b^1 + \binom{5}{2} a^{5-2} b^2 + \binom{5}{3} a^{5-3} b^3 + \binom{5}{4} a^{5-4} b^4 + b^5$$

$$= a^5 + 5a^4 b + 10a^3 b^2 + 10a^2 b^3 + 5ab^4 + b^5$$

Example 9.7.5 – *Solution (2/2)*

continued

b. Observe that $(x-4y)^4 = (x+(-4y))^4$. So let $a = x$ and $b = (-4y)$, and substitute into the binomial theorem.

$$(x-4y)^4 = \sum_{k=0}^{4} \binom{4}{k} x^{4-k}(-4y)^k$$

$$= x^4 + \binom{4}{1} x^{4-1}(-4y)^1 + \binom{4}{2} x^{4-2}(-4y)^2 + \binom{4}{3} x^{4-3}(-4y)^3 + (-4y)^4$$

$$= x^4 + 4x^3(-4y) + 6x^2(16y^2) + 4x^1(-64y^3) + (256y^4)$$

$$= x^4 - 16x^3y + 96x^2y^2 - 256xy^3 + 256y^4$$

Example 9.7.8 – *Using the Binomial Theorem to Simplify a Sum*

Express the following sum in **closed form** (without using a summation symbol and without using an ellipsis …):

$$\sum_{k=0}^{n} \binom{n}{k} 9^k.$$

Example 9.7.8 – *Solution*

When the number 1 is raised to any power, the result is still 1. Thus

$$\sum_{k=0}^{n} \binom{n}{k} 9^k = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} 9^k$$

$$= (1+9)^n$$ by the binomial theorem with $a = 1$ and $b = 9$

$$= 10^n.$$

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.8 Probability Axioms and Expected Value

Copyright © Cengage Learning. All rights reserved.

Probability Axioms and Expected Value (1/4)

Up to this point, we have calculated probabilities only for situations, such as tossing a fair coin or rolling a pair of balanced dice, where the outcomes in the sample space are all equally likely.

But coins are not always fair and dice are not always balanced.

How is it possible to calculate probabilities for these more general situations?

Probability Axioms and Expected Value (2/4)

The following axioms were formulated by A. N. Kolmogorov in 1933 to provide a theoretical foundation for a far-ranging theory of probability.

Probability Axioms

Let S be a sample space. A **probability function** P from the set of all events in S to the set of real numbers satisfies the following three axioms: For all events A and B in S:

1. $0 \leq P(A) \leq 1$.
2. $P(\varnothing) = 0$ and $P(S) = 1$.
3. If A and B are disjoint (that is, if $A \cap B = \varnothing$), then the probability of the union of A and B is

$$P(A \cup B) = P(A) + P(B).$$

Probability Axioms and Expected Value (3/4)

In this section we state the axioms, derive a few consequences, and introduce the notion of expected value.

Recall that a sample space is a set of all outcomes of a random process or experiment and that an event is a subset of a sample space.

Example 9.8.1 – *Applying the Probability Axioms*

Suppose that A and B are events in a sample space S. If A and B are disjoint, could $P(A) = 0.6$ and $P(B) = 0.8$?

Example 9.8.1 – *Solution*

No. Probability axiom 3 would imply that $P(A \cup B) = P(A) + P(B) = 0.6 + 0.8 = 1.4$, and since $1.4 > 1$, this result would violate probability axiom 1.

Example 9.8.2 – *The Probability of the Complement of an Event*

Suppose that *A* is an event in a sample space *S*. Deduce that $P(A^c) = 1 - P(A)$.

Example 9.8.2 – *Solution*

S representing the universal set *U*,

$$A \cap A^c = \varnothing \quad \text{and} \quad A \cup A^c = S.$$

Thus *S* is the disjoint union of *A* and A^c, and so

$$P(A \cup A^c) = P(A) + P(A^c) = P(S) = 1.$$

So $P(A) + P(A^c) = 1$ and subtracting *P*(*A*) from both sides gives the result that $P(A^c) = 1 - P(A)$.

Probability Axioms and Expected Value (4/4)

Probability of the Complement of an Event

If A is any event in a sample space S, then

$$P(A^c) = 1 - P(A). \qquad 9.8.1$$

Probability of a General Union of Two Events

If S is any sample space and A and B are any events in S, then

$$P(A \cup B) = P(A) + P(B) - P(A \cap B). \qquad 9.8.2$$

Example 9.8.4 – *Computing the Probability of a General Union of Two Events*

Suppose a card is chosen at random from an ordinary 52-card deck. What is the probability that the card is a face card (jack, queen, or king) or is from one of the red suits (hearts or diamonds)?

Example 9.8.4 – *Solution (1/2)*

Let A be the event that the chosen card is a face card, and let B be the event that the chosen card is from one of the red suits. The event that the card is a face card or is from one of the red suits is $A \cup B$.

Now $N(A) = 4 \cdot 3 = 12$ (because each of the four suits has three face cards), and so $P(A) = 12/52$.

Also $N(B) = 26$ (because half the cards are red), and so $P(B) = 26/52$.

Example 9.8.4 – *Solution (2/2)*

continued

Finally, $N(A \cap B) = 6$ (because there are three face cards in hearts and another three in diamonds), and so $P(A \cap B) = 6/52$. It follows from the formula for the probability of a union of any two events that

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) = \frac{12}{52} + \frac{26}{52} - \frac{6}{52} = \frac{32}{52} \cong 61.5\%.$$

Thus the probability that the chosen card is a face card or is from one of the red suits is approximately 61.5%.

Expected Value

Expected Value (1/1)

Definition

Suppose the possible outcomes of an experiment, or random process, are real numbers $a_1, a_2, a_3, \ldots, a_n$, which occur with probabilities $p_1, p_2, p_3, \ldots, p_n$. The **expected value** of the process is

$$\sum_{k=1}^{n} a_k p_k = a_1 p_1 + a_2 p_2 + a_3 p_3 + \cdots + a_n p_n.$$

Example 9.8.5 – *Expected Value of a Lottery*

Suppose that 500,000 people pay $5 each to play a lottery game with the following prizes: one grand prize of $1,000,000, 10 second prizes of $1,000 each, 1,000 third prizes of $500 each, and 10,000 fourth prizes of $10 each.

What is the expected value of a ticket?

Example 9.8.5 – *Solution (1/3)*

Each of the 500,000 lottery tickets has the same chance as any other of containing a winning lottery number, and so $p_k = \frac{1}{500{,}000}$ for each k = 1, 2, 3, . . . , 500,000. Let a_i be the net gain for an individual ticket a_i, where a_1 = 999,995 (the net gain for the grand-prize ticket, which is one million dollars minus the \$5 cost of the winning ticket), $a_2 = a_3 = \ldots = a_{11}$ = 995 (the net gain for each of the 10 second-prize tickets), $a_{12} = a_{13} = \ldots = a_{1,011}$ = 495 (the net gain for each of the 1,000 third-prize tickets), and $a_{1,012} = a_{1,013} = \ldots = a_{11,011}$ = 5 (the net gain for each of the 10,000 fourth-prize tickets).

Example 9.8.5 – *Solution (2/3)*

continued

Since the remaining 488,989 tickets just lose \$5, $a_{11,012} = a_{11,013} = \ldots = a_{500,000} = -5$. The expected value of a ticket is therefore

$$\sum_{k=1}^{500,000} a_k p_k = \sum_{k=1}^{500,000} \left(a_k \cdot \frac{1}{500,000} \right) \quad \text{because each } p_k = 1/500,000$$

$$= \frac{1}{500,000} \sum_{k=1}^{500,000} a_k \quad \text{by theorem 5.1.1(2)}$$

Example 9.8.5 – *Solution (3/3)*

continued

$$= \frac{1}{500{,}000}(999{,}995 + 10 \cdot 995 + 1{,}000 \cdot 495 + 10{,}000 \cdot 5 + (-5) \cdot 488{,}989)$$

$$= \frac{1}{500{,}000}(999{,}995 + 9{,}950 + 495{,}000 + 50{,}000 - 2{,}444{,}945)$$

$= -1.78.$

In other words, a person who continues to play this lottery for a very long time will probably win some money occasionally but on average will lose $1.78 per ticket.

CHAPTER 9

COUNTING AND PROBABILITY

Copyright © Cengage Learning. All rights reserved.

9.9 Conditional Probability, Bayes' Formula, and Independent Events

Copyright © Cengage Learning. All rights reserved.

Conditional Probability

Conditional Probability (1/2)

Definition

Let A and B be events in a sample space S. If $P(A) \neq 0$, then the **conditional probability of B given A**, denoted $\boldsymbol{P(B|A)}$, is

$$P(B|A) = \frac{P(A \cap B)}{P(A)}. \qquad 9.9.1$$

Example 9.9.1 – *Computing a Conditional Probability*

A pair of fair dice, one blue and the other gray, are rolled. What is the probability that the sum of the numbers showing face up is 8, given that both of the numbers are even?

Example 9.9.1 – *Solution (1/2)*

The sample space is the set of all 36 outcomes obtained from rolling the two dice and noting the numbers showing face up on each.

As in earlier section, denote by ab the outcome that the number showing face up on the blue die is a and the one on the gray die is b.

Let A be the event that both numbers are even and B the event that the sum of the numbers is 8.

Example 9.9.1 – *Solution (2/2)*

continued

Then $A = \{22, 24, 26, 42, 44, 46, 62, 64, 66\}$, $B = \{26, 35, 44, 53, 62\}$, and $A \cap B = \{26, 44, 62\}$.

Because the dice are fair (so all outcomes are equally likely), $P(A) = 9/36$, $P(B) = 5/36$, and $P(A \cap B) = 3/36$. By definition of conditional probability,

$$P(B|A) = \frac{P(A \cap \mathrm{B})}{P(A)} = \frac{\frac{3}{36}}{\frac{9}{36}} = \frac{3}{9} = \frac{1}{3}.$$

Conditional Probability (2/2)

Note that when both sides of the formula for conditional probability (formula 9.9.1) are multiplied by $P(A)$, a formula for $P(A \cap B)$ is obtained:

$$P(A \cap B) = P(B|A) \cdot P(A). \qquad 9.9.2$$

And dividing both sides of formula (9.9.2) by $P(B|A)$ gives a formula for $P(A)$:

$$P(A) = \frac{P(A \cap B)}{P(B|A)}. \qquad 9.9.3$$

Bayes’ Theorem

Bayes' Theorem (1/1)

Thomas Bayes was an English Presbyterian minister who devoted much of his energies to mathematics. The theorem that bears his name was published posthumously in 1763. The portrait at the left is the only one attributed to him, but its authenticity has recently come into question.

Theorem 9.9.1 Bayes' Theorem

Suppose a sample space S is a union of mutually disjoint events $B_1, B_2, B_3, \ldots, B_n$, suppose A is an event in S, and suppose both A and each B_k have nonzero probabilities for every integer k with $1 \le k \le n$. Then

$$P(B_k|A) = \frac{P(A|B_k)P(B_k)}{P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \cdots + P(A|B_n)P(B_n)}.$$

Example 9.9.3 – *Applying Bayes' Theorem (1/3)*

Most medical tests occasionally produce incorrect results, called false positives and false negatives.

When a test is designed to determine whether a patient has a certain disease, a **false positive** result indicates that a patient has the disease when the patient does not have it.

A **false negative** result indicates that a patient does not have the disease when the patient does have it.

Example 9.9.3 – *Applying Bayes' Theorem (2/3)*

When large-scale health screenings are performed for diseases with relatively low incidence, those who develop the screening procedures have to balance several considerations: the per-person cost of the screening, follow-up costs for further testing of false positives, and the possibility that people who have the disease will develop unwarranted confidence in the state of their health.

Consider a medical test that screens for a disease found in 5 people in 1,000. Suppose that the false positive rate is 3% and the false negative rate is 1%.

Example 9.9.3 – *Applying Bayes' Theorem (3/3)*

Then 99% of the time a person who has the condition tests positive for it, and 97% of the time a person who does not have the condition tests negative for it.

a. What is the probability that a randomly chosen person who tests positive for the disease actually has the disease?

b. What is the probability that a randomly chosen person who tests negative for the disease does not in fact have the disease?

Example 9.9.3 – *Solution (1/5)*

Consider a person chosen at random from among those screened.

Let A be the event that the person tests positive for the disease, B_1 the event that the person actually has the disease, and B_2 the event that the person does not have the disease. Then

$$P(A|B_1) = 0.99, \quad P(A^c|B_1) = 0.01, \quad P(A^c|B_2) = 0.97,$$
$$\text{and} \quad P(A|B_2) = 0.03.$$

Example 9.9.3 – *Solution (2/5)* continued

Also, because 5 people in 1,000 have the disease,

$P(B_1) = 0.005$ and $P(B_2) = 0.995$.

a. By Bayes' theorem,

$$P(B_1|A) = \frac{P(A|B_1)P(B_1)}{P(A|B_1)P(B_1) + P(A|B_2)P(B_2)}$$

$$= \frac{(0.99)(0.005)}{(0.99)(0.005) + (0.03)(0.995)}$$

$\cong 0.1422 \cong 14.2\%$.

Example 9.9.3 – *Solution (3/5)* continued

Thus the probability that a person with a positive test result actually has the disease is approximately 14.2%.

b. By Bayes' theorem,

$$P(B_2|A^c) = \frac{P(A^c|B_2)P(B_2)}{P(A^c|B_1)P(B_1) + P(A^c|B_2)P(B_2)}$$

$$= \frac{(0.97)(0.995)}{(0.01)(0.005) + (0.97)(0.995)}$$

$$\cong 0.999948 \cong 99.995\%.$$

Example 9.9.3 – *Solution (4/5)* continued

Thus the probability that a person with a negative test result does not have the disease is approximately 99.995%.

You might be surprised by these numbers, but they are fairly typical of the situation where the screening test is significantly less expensive than a more accurate test for the same disease yet produces positive results for nearly all people with the disease.

Example 9.9.3 – *Solution (5/5)*

continued

Using the screening test limits the expense of unnecessarily using the more costly test to a relatively small percentage of the population being screened, while only rarely indicating that a person who has the disease is free of it.

Independent Events

Independent Events (1/5)

Suppose a fair coin is tossed twice. It seems intuitively clear that the outcome of the first toss does not depend in any way on the outcome of the second toss, and conversely, the outcome of the second toss does not depend on the outcome of the first toss.

In other words, if, for instance, *A* is the event that a head is obtained on the first toss and *B* is the event that a head is obtained on the second toss, then if the coin is tossed randomly both times, events *A* and *B* should be *independent* in the sense that $P(A|B) = P(A)$ and $P(B|A) = P(B)$.

Independent Events (2/5)

Definition

If A and B are events in a sample space S, then A and B are **independent** if, and only if,

$$P(A \cap B) = P(A) \cdot P(B).$$

Example 9.9.4 – *Disjoint Events and Independence*

Let A and B be events in a sample space S, and suppose $A \cap B = \varnothing$, $P(A) \neq 0$, and $P(B) \neq 0$. Show that

$$P(A \cap B) \neq P(A) \cdot P(B).$$

Example 9.9.4 – *Solution*

Because

$$A \cap B = \varnothing, P(A \cap B) = 0$$

by probability axiom 2. But $P(A) \cdot P(B) \neq 0$ because neither $P(A)$ nor $P(B)$ equals zero.

Thus $P(A \cap B) \neq P(A) \cdot P(B)$.

Example 9.9.6 – *Computing Probabilities of Intersections of Two Independent Events (1/2)*

A coin is loaded so that the probability of heads is 0.6. Suppose the coin is tossed twice.

Although the probability of heads is greater than the probability of tails, there is no reason to believe that whether the coin lands heads or tails on one toss will affect whether it lands heads or tails on the other toss.

Thus it is reasonable to assume that the results of the tosses are independent.

Example 9.9.6 – *Computing Probabilities of Intersections of Two Independent Events (2/2)*

continued

a. What is the probability of obtaining two heads?

b. What is the probability of obtaining one head?

c. What is the probability of obtaining no heads?

d. What is the probability of obtaining at least one head?

Example 9.9.6 – *Solution (1/5)*

The sample space S consists of the four outcomes $\{HH, HT, TH, TT\}$, which are not equally likely.

Let E be the event that a head is obtained on the first toss, and let F be the event that a head is obtained on the second toss.

Then $P(E) = P(F) = 0.6$, and it is to be assumed that E and F are independent.

Example 9.9.6 – *Solution (2/5)* continued

a. The event of obtaining two heads is $E \cap F$. And because E and F are independent,

$$P(\text{two heads}) = P(E \cap F) = P(E) \cdot P(F) = (0.6)(0.6)$$
$$= 0.36 = 36\%.$$

b. One head can be obtained in two mutually exclusive ways: head on the first toss and tail on the second, or tail on the first toss and head on the second. Thus, the event of obtaining exactly one head is $(E \cap F^c) \cup (E^c \cap F)$.

Example 9.9.6 – *Solution (3/5)* continued

Also $(E \cap F^c) \cap (E^c \cap F) = \varnothing$, and, moreover, by the formula for the probability of the complement of an event, $P(E^c) = P(F^c) = 1 - 0.6 = 0.4$. Hence

$$P(\text{one head}) = P((E \cap F^c) \cup (E^c \cap F))$$

$$= P(E) \cdot P(F^c) + P(E^c) \cdot P(F)$$

$$= (0.6)(0.4) + (0.4)(0.6)$$

$$= 0.48 = 48\%.$$

c. The event of obtaining no heads is $E^c \cap F^c$.

Example 9.9.6 – *Solution (4/5)* continued

Thus,

$$P(\text{no heads}) = P(E^c \cap F^c) = P(E^c) \cdot P(F^c)$$

$$= (0.4)(0.4) = 0.16 = 16\%.$$

d. There are two ways to solve this problem. One is to observe that because the event of obtaining one head and the event of obtaining two heads are mutually disjoint,

$$P(\text{at least one head}) = P(\text{one head}) + P(\text{two heads})$$

Example 9.9.6 – *Solution (5/5)* continued

$= 0.48 + 0.36$ by parts (a) and (b)

$= 0.84$

$= 84\%.$

The second way is to use the fact that the event of obtaining at least one head is the complement of the event of obtaining no heads. So

$P(\text{at least one head}) = 1 - P(\text{no heads})$

$= 1 - 0.16$ by part (c)

$= 0.84 = 84\%.$

Example 9.9.7 – *Expected Value of Tossing a Loaded Coin Twice*

Suppose that a coin is loaded so that the probability of heads is 0.6, and suppose the coin is tossed twice.

If this experiment is repeated many times, what is the expected value of the number of heads?

Example 9.9.7 – *Solution*

Think of the outcomes of the coin tosses as just 0, 1, or 2 heads.

Example 9.9.6 showed that the probabilities of these outcomes are 0.16, 0.48, and 0.36, respectively.

Thus, by definition of expected value, the

$$\text{expected number of heads} = 0 \cdot (0.16) + 1 \cdot (0.48) + 2 \cdot (0.36)$$
$$= 1.2.$$

Independent Events (3/5)

Four conditions must be included in the definition of independence for three events.

Definition

Let A, B, and C be events in a sample space S. A, B, and C are **pairwise independent** if, and only if, they satisfy conditions 1–3 below. They are **mutually independent** if, and only if, they satisfy all four conditions below.

1. $P(A \cap B) = P(A) \cdot P(B)$
2. $P(A \cap C) = P(A) \cdot P(C)$
3. $P(B \cap C) = P(B) \cdot P(C)$
4. $P(A \cap B \cap C) = P(A) \cdot P(B) \cdot P(C)$

Independent Events (4/5)

The definition of mutual independence for any collection of *n* events with $n \geq 2$ generalizes the two definitions given previously.

Definition

Events $A_1, A_2, A_3, \ldots, A_n$ in a sample space S are **mutually independent** if, and only if, the probability of the intersection of any subset of the events is the product of the probabilities of the events in the subset.

Example 9.9.9 – *Tossing a Loaded Coin Ten Times*

A coin is loaded so that the probability of heads is 0.6 (and thus the probability of tails is 0.4). Suppose the coin is tossed ten times. As in Example 9.9.6, it is reasonable to assume that the results of the tosses are mutually independent.

a. What is the probability of obtaining eight heads?

b. What is the probability of obtaining at least eight heads?

Example 9.9.9 – *Solution (1/6)*

a. For each $i = 1, 2, \ldots, 10$, let H_i be the event that a head is obtained on the i th toss, and let T_i be the event that a tail is obtained on the ith toss.

Suppose that the eight heads occur on the first eight tosses and that the remaining two tosses are tails. This is the event

$$H_1 \cap H_2 \cap H_3 \cap H_4 \cap H_5 \cap H_6 \cap H_7 \cap H_8 \cap T_9 \cap T_{10}.$$

For simplicity, we denote it as *HHHHHHHHTT*.

Example 9.9.9 – *Solution (2/6)*

continued

By definition of mutually independent events,

$$P(HHHHHHHHTT) = (0.6)^8(0.4)^2.$$

Because of the commutative law for multiplication, if the eight heads occur on any other of the ten tosses, the same number is obtained. For instance, if we denote the event $H_1 \cap H_2 \cap T_3 \cap H_4 \cap H_5 \cap H_6 \cap H_7 \cap H_8 \cap T_9 \cap H_{10}$ by $H\ H\ T\ H\ H\ H\ H\ H\ T\ H$, then

$$P(HHTHHHHHTH) = (0.6)^2(0.4)(0.6)^5(0.4)(0.6) = (0.6)^8(0.4)^2.$$

Example 9.9.9 – *Solution (3/6)*

continued

Now there are as many different ways to obtain eight heads in ten tosses as there are subsets of eight elements (the toss numbers on which heads are obtained) that can be chosen from a set of ten elements.

This number is $\binom{10}{8}$. It follows that, because the different ways of obtaining eight heads are all mutually exclusive,

$$P(\text{eight heads}) = \binom{10}{8}(0.6)^8(0.4)^2.$$

Example 9.9.9 – *Solution (4/6)*

continued

b. By reasoning similar to that in part (a),

$$P(\text{nine heads}) = \begin{bmatrix} \text{the number of different} \\ \text{ways nine heads can be} \\ \text{obtained in ten tosses} \end{bmatrix} \cdot (0.6)^9(0.4)^1$$

$$= \binom{10}{9}(0.6)^9(0.4),$$

and

$$P(\text{ten heads}) = \begin{bmatrix} \text{the number of different} \\ \text{ways ten heads can be} \\ \text{obtained in ten tosses} \end{bmatrix} \cdot (0.6)^{10}(0.4)^0$$

Example 9.9.9 – *Solution (5/6)*

continued

$$= \binom{10}{10}(0.6)^{10}.$$

Because obtaining eight, obtaining nine, and obtaining ten heads are mutually disjoint events,

P(at least eight heads) = *P*(eight heads) +

P(nine heads) + *P*(ten heads)

$$= \binom{10}{8}(0.6)^8(0.4)^2 + \binom{10}{9}(0.6)^9(0.4) + \binom{10}{10}(0.6)^{10}$$

Example 9.9.9 – *Solution (6/6)*

continued

$\cong 0.167$

$= 16.7\%.$

Independent Events (5/5)

Probabilities of the form

$$\binom{n}{k} p^{n-k}(1-p)^k,$$

where $0 \leq p \leq 1$, are called **binomial probabilities**.

CHAPTER 10

THEORY OF GRAPHS AND TREES

Copyright © Cengage Learning. All rights reserved.

10.1 Trails, Paths, and Circuits

Copyright © Cengage Learning. All rights reserved.

Trails, Paths, and Circuits (1/6)

The subject of graph theory began in the year 1736 when the great mathematician Leonhard Euler published a paper giving the solution to the following puzzle:

The town of Königsberg in Prussia (now Kaliningrad in Russia) was built at a point where two branches of the Pregel River came together. It consisted of an island and some land along the river banks.

Trails, Paths, and Circuits (2/6)

These were connected by seven bridges as shown in Figure 10.1.1.

The Seven Bridges of Königsberg

Figure 10.1.1

Trails, Paths, and Circuits (3/6)

The question is this: Is it possible for a person to take a walk around town, starting and ending at the same location and crossing each of the seven bridges exactly once?

To solve this puzzle, Euler translated it into a graph theory problem. He noticed that all points of a given land mass can be identified with each other since a person can travel from any one point to any other point of the same land mass without crossing a bridge.

Trails, Paths, and Circuits (4/6)

Thus for the purpose of solving the puzzle, the map of Königsberg can be identified with the graph shown in Figure 10.1.2, in which the vertices *A*, *B*, *C*, and *D* represent land masses and the seven edges represent the seven bridges.

Graph Version of Königsberg Map

Figure 10.1.2

Trails, Paths, and Circuits (5/6)

Definition

Let G be a graph, and let v and w be vertices in G.

A **walk from v to w** is a finite alternating sequence of adjacent vertices and edges of G. Thus a walk has the form

$$v_0 e_1 v_1 e_2 \cdots v_{n-1} e_n v_n,$$

where the v's represent vertices, the e's represent edges, $v_0 = v$, $v_n = w$, and for each $i = 1, 2, \ldots n$, v_{i-1} and v_i are the endpoints of e_i. The **trivial walk from v to v** consists of the single vertex v.

A **trail from v to w** is a walk from v to w that does not contain a repeated edge.

A **path from v to w** is a trail that does not contain a repeated vertex.

A **closed walk** is a walk that starts and ends at the same vertex.

A **circuit** is a closed walk that contains at least one edge and does not contain a repeated edge.

A **simple circuit** is a circuit that does not have any other repeated vertex except the first and last.

Trails, Paths, and Circuits (6/6)

For ease of reference, these definitions are summarized in the following table:

	Repeated Edge?	Repeated Vertex?	Starts and Ends at the Same Point?	Must Contain at Least One Edge?
Walk	allowed	allowed	allowed	no
Trail	no	allowed	allowed	no
Path	no	no	no	no
Closed walk	allowed	allowed	yes	no
Circuit	no	allowed	yes	yes
Simple circuit	no	first and last only	yes	yes

Example 10.1.1 – *Notation for Walks (1/2)*

a. In the graph below, the notation $e_1e_2e_4e_3$ refers unambiguously to the following walk: $v_1e_1v_2e_2v_3e_4v_3e_3v_2$. On the other hand, the notation e_1 is ambiguous if used by itself to refer to a walk. It could mean either $v_1e_1v_2$ or $v_2e_1v_1$.

Example 10.1.1 – *Notation for Walks (2/2)*

b. In the graph of part (a), the notation v_2v_3 is ambiguous if used to refer to a walk. It could mean $v_2e_2v_3$ or $v_2e_3v_3$. On the other hand, in the graph below, the notation $v_1v_2v_2v_3$ refers unambiguously to the walk $v_1e_1v_2e_2v_2e_3v_3$.

Example 10.1.2 – *Walks, Trails, Paths, and Circuits*

In the graph below, determine which of the following walks are trails, paths, circuits, or simple circuits.

a. $v_1e_1v_2e_3v_3e_4v_3e_5v_4$

b. $e_1e_3e_5e_5e_6$

c. $v_2v_3v_4v_5v_3v_6v_2$

d. $v_2v_3v_4v_5v_6v_2$

e. $v_1e_1v_2e_1v_1$

f . v_1

Example 10.1.2 – *Solution (1/2)*

a. This walk has a repeated vertex but does not have a repeated edge, so it is a trail from v_1 to v_4 but not a path.

b. This is just a walk from v_1 to v_5. It is not a trail because it has a repeated edge.

c. This walk starts and ends at v_2, contains at least one edge, and does not have a repeated edge, so it is a circuit. Since the vertex v_3 is repeated in the middle, it is not a simple circuit.

Example 10.1.2 – *Solution (2/2)* continued

d. This walk starts and ends at v_2, contains at least one edge, does not have a repeated edge, and does not have a repeated vertex. Thus it is a simple circuit.

e. This is just a closed walk starting and ending at v_1. It is not a circuit because edge e_1 is repeated.

f. The first vertex of this walk is the same as its last vertex, but it does not contain an edge, and so it is not a circuit. It is a closed walk from v_1 to v_1. (It is also a trail from v_1 to v_1.)

Subgraphs

Subgraphs (1/1)

Definition

A graph H is said to be a **subgraph** of a graph G if, and only if, every vertex in H is also a vertex in G, every edge in H is also an edge in G, and every edge in H has the same endpoints as it has in G.

Example 10.1.3 – *Subgraphs*

List all subgraphs of the graph G with vertex set $\{v_1, v_2\}$ and edge set $\{e_1, e_2, e_3\}$, where the endpoints of e_1 are v_1 and v_2, the endpoints of e_2 are v_1 and v_2, and e_3 is a loop at v_1.

Example 10.1.3 – *Solution (1/2)*

G can be drawn as shown below.

There are 11 subgraphs of *G*, which can be grouped according to those that do not have any edges, those that have one edge, those that have two edges, and those that have three edges.

Example 10.1.3 – *Solution (2/2)* continued

The 11 subgraphs are shown in Figure 10.1.3.

Figure 10.1.3

Connectedness

Connectedness (1/4)

It is easy to understand the concept of connectedness on an intuitive level. Roughly speaking, a graph is connected if it is possible to travel from any vertex to any other vertex along a sequence of adjacent edges of the graph.

The formal definition of connectedness is stated in terms of walks.

Connectedness (2/4)

Definition

Let G be a graph. Two **vertices v and w of G are connected** if, and only if, there is a walk from v to w. The **graph G is connected** if, and only if, given *any* two vertices v and w in G, there is a walk from v to w. Symbolically:

$$G \text{ is connected} \quad \Leftrightarrow \quad \forall \text{ vertices } v \text{ and } w \text{ in } G, \exists \text{ a walk from } v \text{ to } w.$$

If you take the negation of this definition, you will see that a graph G is *not connected* if, and only if, there exist two vertices of G that are not connected by any walk.

Example 10.1.4 – *Connected and Disconnected Graphs*

Which of the following graphs are connected?

Example 10.1.4 – *Solution*

The graph represented in (a) is connected, whereas those of (b) and (c) are not. To understand why (c) is not connected, recall that in a drawing of a graph, two edges may cross at a point that is not a vertex. Thus the graph in (c) can be redrawn as follows:

Connectedness (3/4)

Some useful facts relating circuits and connectedness are collected in the following lemma.

Lemma 10.1.1

Let G be a graph.

a. If G is connected, then any two distinct vertices of G can be connected by a path.

b. If vertices v and w are part of a circuit in G and one edge is removed from the circuit, then there still exists a trail from v to w in G.

c. If G is connected and G contains a circuit, then an edge of the circuit can be removed without disconnecting G.

Connectedness (4/4)

A *connected component* of a graph is a connected subgraph of largest possible size.

Definition

A graph H is a **connected component** of a graph G if, and only if,

1. H is subgraph of G;
2. H is connected; and
3. no connected subgraph of G has H as a subgraph and contains vertices or edges that are not in H.

Example 10.1.5 – *Connected Components*

Find all connected components of the following graph *G*.

Example 10.1.5 – *Solution*

G has three connected components: H_1, H_2, and H_3 with vertex sets V_1, V_2, and V_3 and edge sets E_1, E_2, and E_3, where

$$V_1 = \{v_1, v_2, v_3\}, \quad E_1 = \{e_1, e_2\},$$
$$V_2 = \{v_4\}, \quad E_2 = \varnothing,$$
$$V_3 = \{v_5, v_6, v_7, v_8\}, \quad E_3 = \{e_3, e_4, e_5\}.$$

Euler Circuits

Euler Circuits (1/5)

Now we return to consider general problems similar to the puzzle of the Königsberg bridges. The following definition is made in honor of Euler.

Definition

Let G be a graph. An **Euler circuit** for G is a circuit that contains every vertex and every edge of G. That is, an Euler circuit for G is a sequence of adjacent vertices and edges in G that has at least one edge, starts and ends at the same vertex, uses every vertex of G at least once, and uses every edge of G exactly once.

Euler Circuits (2/5)

The analysis used earlier to solve the puzzle of the Königsberg bridges generalizes to prove the following theorem:

Theorem 10.1.2

If a graph has an Euler circuit, then every vertex of the graph has positive even degree.

Contrapositive Version of Theorem 10.1.2

If some vertex of a graph has odd degree, then the graph does not have an Euler circuit.

Example 10.1.6 – *Showing That a Graph Does Not Have an Euler Circuit*

Show that the graph below does not have an Euler circuit.

Example 10.1.6 – *Solution*

Vertices v_1 and v_3 both have degree 3, which is odd. Hence by (the contrapositive form of) Theorem 10.1.2, this graph does not have an Euler circuit.

Theorem 10.1.2

If a graph has an Euler circuit, then every vertex of the graph has positive even degree.

Contrapositive Version of Theorem 10.1.2

If some vertex of a graph has odd degree, then the graph does not have an Euler circuit.

Euler Circuits (3/5)

Theorem 10.1.3

If a graph G is connected and the degree of every vertex of G is a positive even integer, then G has an Euler circuit.

Example 10.1.7 – *Finding an Euler Circuit*

Use Theorem 10.1.3 to check that the graph below has an Euler circuit. Then use the algorithm from the proof of the theorem to find an Euler circuit for the graph.

Example 10.1.7 – *Solution (1/5)*

Observe that

$\deg(a) = \deg(b) = \deg(c) = \deg(f) = \deg(g) = \deg(i) = \deg(j) = 2$

and that $\deg(d) = \deg(e) = \deg(h) = 4$.

Hence all vertices have even degree. Also, the graph is connected. Thus, by Theorem 10.1.3, the graph has an Euler circuit.

Example 10.1.7 – *Solution (2/5)* continued

To construct an Euler circuit using the algorithm of Theorem 10.1.3, let $v = a$ and let C be

$$C: abcda.$$

C is represented by the labeled edges shown below.

Example 10.1.7 – *Solution (3/5)* continued

Observe that C is not an Euler circuit for the graph but that C intersects the rest of the graph at d. Let C' be C': *deghjid*.

Patch C' into C to obtain C'': *abcdeghjida*.

Set $C = C''$. Then C is represented by the labeled edges shown below.

Example 10.1.7 – *Solution (4/5)* continued

Observe that C is not an Euler circuit for the graph but that it intersects the rest of the graph at e and h. Let C' be

$$C': efhe.$$

Patch C' into C to obtain

$$C'': abcdefheghjida.$$

Example 10.1.7 – *Solution (5/5)* continued

Set $C = C''$. Then C is represented by the labeled edges shown below.

Since C includes every edge of the graph exactly once, C is an Euler circuit for the graph.

Euler Circuits (4/5)

Theorem 10.1.4

A graph G has an Euler circuit if, and only if, G is connected and every vertex of G has positive even degree.

A corollary to Theorem 10.1.4 gives a criterion for determining when it is possible to find a walk from one vertex of a graph to another, passing through every vertex of the graph at least once and every edge of the graph exactly once.

Euler Circuits (5/5)

Definition

Let G be a graph, and let v and w be two distinct vertices of G. An **Euler trail from v to w** is a sequence of adjacent edges and vertices that starts at v, ends at w, passes through every vertex of G at least once, and traverses every edge of G exactly once.

Corollary 10.1.5

Let G be a graph, and let v and w be two distinct vertices of G. There is an Euler trail from v to w if, and only if, G is connected, v and w have odd degree, and all other vertices of G have positive even degree.

Example 10.1.8 – *Finding an Euler Trail*

The floor plan shown below is for a house that is open for public viewing. Is it possible to find a trail that starts in room *A*, ends in room *B*, and passes through every interior doorway of the house exactly once? If so, find such a trail.

Example 10.1.8 – *Solution*

Let the floor plan of the house be represented by the graph below, where the edges indicate the openings between the rooms.

Each vertex of this graph has even degree except for *A* and *B*, each of which has degree 1. Hence by Corollary 10.1.5, there is an Euler trail from *A* to *B*. One such trail is

A G H F E I H E K J D C B.

Hamiltonian Circuits

Hamiltonian Circuits (1/6)

In 1859 the Irish mathematician Sir William Rowan Hamilton introduced a puzzle in the shape of a dodecahedron (DOH-dek-a-HEE-dron). (Figure 10.1.7 contains a drawing of a dodecahedron, which is a solid figure with 12 identical pentagonal faces.)

Dodecahedron

Figure 10.1.7

Hamiltonian Circuits (2/6)

Each vertex was labeled with the name of a city—London, Paris, Hong Kong, New York, and so on.

The problem Hamilton posed was to start at one city and tour the world by visiting each other city exactly once and returning to the starting city.

Hamiltonian Circuits (3/6)

One way to solve the puzzle is to imagine the surface of the dodecahedron stretched out and laid flat in the plane, as follows:

One solution is the circuit

A B C D E F G H I J K L M N O P Q R S T A,

whose edges are indicated with black lines.

Hamiltonian Circuits (4/6)

Note that although every city is visited, many edges are omitted from the circuit. (More difficult versions of the puzzle required that certain cities be visited in a certain order.)

The following definition is made in honor of Hamilton.

Definition

Given a graph G, a **Hamiltonian circuit** for G is a simple circuit that includes every vertex of G. That is, a Hamiltonian circuit for G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once, except for the first and the last, which are the same.

Hamiltonian Circuits (5/6)

Proposition 10.1.6

If a graph G has a Hamiltonian circuit, then G has a subgraph H with the following properties:

1. H contains every vertex of G.
2. H is connected.
3. H has the same number of edges as vertices.
4. Every vertex of H has degree 2.

Example 10.1.9 – *Showing That a Graph Does Not Have a Hamiltonian Circuit*

Prove that the graph *G* shown below does not have a Hamiltonian circuit.

Example 10.1.9 – *Solution (1/2)*

If G has a Hamiltonian circuit, then by Proposition 10.1.6, G has a subgraph H that (1) contains every vertex of G, (2) is connected, (3) has the same number of edges as vertices, and (4) is such that every vertex has degree 2. Suppose such a subgraph H exists.

In other words, suppose there is a connected subgraph H of G such that H has five vertices (a, b, c, d, e) and five edges and such that every vertex of H has degree 2.

Example 10.1.9 – *Solution (2/2)* continued

Since the degree of b in G is 4 and every vertex of H has degree 2, two edges incident on b must be removed from G to create H. Edge $\{a, b\}$ cannot be removed because if it were, vertex a would have degree less than 2 in H. Similar reasoning shows that edges $\{e, b\}$, $\{b, a\}$, and $\{b, d\}$ cannot be removed either.

It follows that the degree of b in H must be 4, which contradicts the condition that every vertex in H has degree 2 in H. Hence no such subgraph H exists, and so G does not have a Hamiltonian circuit.

Hamiltonian Circuits (6/6)

The next example illustrates a type of problem known as a **traveling salesman problem**. It is a variation of the problem of finding a Hamiltonian circuit for a graph.

Example 10.1.10 – *A Traveling Salesman Problem*

Imagine that the drawing below is a map showing four cities and the distances in kilometers between them. Suppose that a salesman must travel to each city exactly once, starting and ending in city *A.* Which route from city to city will minimize the total distance that must be traveled?

Example 10.1.10 – *Solution*

This problem can be solved by writing all possible Hamiltonian circuits starting and ending at *A* and calculating the total distance traveled for each.

Route	Total Distance (In Kilometers)	
A B C D A	$30 + 30 + 25 + 40 = 125$	
A B D C A	$30 + 35 + 25 + 50 = 140$	
A C B D A	$50 + 30 + 35 + 40 = 155$	
A C D B A	140	[*A B D C A* backward]
A D B C A	155	[*A C B D A* backward]
A D C B A	125	[*A B C D A* backward]

Thus either route *A B C D A* or *A D C B A* gives a minimum total distance of 125 kilometers.

CHAPTER 10

THEORY OF GRAPHS AND TREES

Copyright © Cengage Learning. All rights reserved.

10.2 Matrix Representations of Graphs

Copyright © Cengage Learning. All rights reserved.

Matrices

Matrices (1/4)

Matrices are two-dimensional analogues of sequences. They are also called two-dimensional arrays.

Definition

An $m \times n$ (read "m by n") **matrix A over a set** S is a rectangular array of elements of S arranged into m rows and n columns:

$$\mathbf{A} = \begin{array}{c} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix} \begin{array}{l} \\ \\ \\ \leftarrow i\text{th row of } \mathbf{A} \\ \\ \\ \end{array} \\ \uparrow \\ j\text{th column of } \mathbf{A} \end{array}$$

We write $\mathbf{A} = (a_{ij})$.

Matrices (2/4)

The ***i* th row of A** is

$$[a_{i1} \quad a_{i2} \quad \ldots \quad a_{in}]$$

and the ***j* th column of A** is

$$\begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}.$$

Matrices (3/4)

The entry a_{ij} in the i th row and j th column of **A** is called the ***i j* th entry of A**. An $m \times n$ matrix is said to have **size** $\boldsymbol{m \times n}$.

If **A** and **B** are matrices, then **A** = **B** if, and only if, **A** and **B** have the same size and the corresponding entries of **A** and **B** are all equal; that is,

$$a_{ij} = b_{ij} \quad \text{for every } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n.$$

Matrices (4/4)

A matrix for which the numbers of rows and columns are equal is called a **square matrix**.

If **A** is a square matrix of size $n \times n$, then the **main diagonal of A** consists of all the entries $a_{11}, a_{22}, \ldots, a_{nn}$:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1i} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2i} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{ni} & \cdots & a_{nn} \end{bmatrix}$$

main diagonal of A

Example 10.2.1 – *Matrix Terminology*

The following is a 3×3 matrix over the set of integers.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -3 \\ 4 & -1 & 5 \\ -2 & 2 & 0 \end{bmatrix}$$

a. What is a_{23}, the entry in row 2, column 3?

b. What is the second column of **A**?

c. What are the entries in the main diagonal of **A**?

Example 10.2.1 – *Solution*

a. $a_{23} = 5$

b. $\begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}$

c. $1, -1$, and 0

Matrices and Directed Graphs

Matrices and Directed Graphs (1/2)

Consider the directed graph shown in Figure 10.2.1. This graph can be represented by the matrix $\mathbf{A} = (a_{ij})$ for which a_{ij} = the number of arrows from v_i to v_j, for every $i = 1, 2, 3$ and $j = 1, 2, 3$.

Directed Graph G

(a)

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

Adjacency Matrix of G

(b)

A Directed Graph and Its Adjacency Matrix

Figure 10.2.1

Matrices and Directed Graphs (2/2)

Thus $a_{11} = 1$ because there is one arrow from v_1 to v_1; $a_{12} = 0$ because there is no arrow from v_1 to v_2, $a_{23} = 2$ because there are two arrows from v_2 to v_3, and so forth. **A** is called the *adjacency matrix* of the directed graph. For convenient reference, the rows and columns of **A** are often labeled with the vertices of the graph G.

Definition

Let G be a directed graph with ordered vertices $v_1, v_2, \ldots, v_n$. The **adjacency matrix of G** is the $n \times n$ matrix $\mathbf{A} = (a_{ij})$ over the set of nonnegative integers such that

$$a_{ij} = \text{the number of arrows from } v_i \text{ to } v_j \quad \text{for all } i, j = 1, 2, \ldots, n.$$

Example 10.2.2 – *The Adjacency Matrix of a Graph*

The two directed graphs shown below differ only in the ordering of their vertices.

Find their adjacency matrices.

(a)

(b)

Example 10.2.2 – *Solution (1/2)*

Since both graphs have three vertices, both adjacency matrices are 3×3 matrices.

For (a), all entries in the first row are 0 since there are no arrows from v_1 to any other vertex.

For (b), the first two entries in the first row are 1 and the third entry is 0 since from v_1 there are single arrows to v_1 and to v_2 and no arrows to v_3.

Example 10.2.2 – *Solution (2/2)* continued

Continuing the analysis in this way, you obtain the following two adjacency matrices:

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix} \end{array} \qquad \mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

(a) (b)

Example 10.2.3 – *Obtaining a Directed Graph from a Matrix*

Let

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 \end{bmatrix}.$$

Draw a directed graph that has **A** as its adjacency matrix.

Example 10.2.3 – *Solution (1/2)*

Let G be the graph corresponding to **A**, and let v_1, v_2, v_3, and v_4 be the vertices of G. Label **A** across the top and down the left side with these vertex names, as shown below.

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Example 10.2.3 – *Solution (2/2)* continued

Then, for instance, the 2 in the fourth row and the first column means that there are two arrows from v_4 to v_1. The 0 in the first row and the fourth column means that there is no arrow from v_1 to v_4. A corresponding directed graph is shown here (without edge labels because the matrix does not determine those).

Matrices and Undirected Graphs

Matrices and Undirected Graphs (1/2)

Once you know how to associate a matrix with a directed graph, the definition of the matrix corresponding to an undirected graph should seem natural to you. As before, you must order the vertices of the graph, but in this case you simply set the *i j* th entry of the adjacency matrix equal to the number of edges connecting the *i* th and *j* th vertices of the graph.

Definition

Let G be an undirected graph with ordered vertices $v_1, v_2, \ldots, v_n$. The **adjacency matrix of G** is the $n \times n$ matrix $\mathbf{A} = (a_{ij})$ over the set of nonnegative integers such that

$$a_{ij} = \text{the number of edges connecting } v_i \text{ and } v_j$$

for every i and $j = 1, 2, \ldots, n$.

Example 10.2.4 – *Finding the Adjacency Matrix of a Graph*

Find the adjacency matrix for the graph *G* shown below.

Example 10.2.4 – *Solution*

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Matrices and Undirected Graphs (2/2)

The entries of **A** satisfy the condition, $a_{ij} = a_{ji}$, for every i, $j = 1, 2, \ldots, n$. This implies that the appearance of **A** remains the same if the entries of **A** are flipped across its main diagonal. A matrix, like **A**, with this property is said to be *symmetric*.

Definition

An $n \times n$ square matrix $\mathbf{A} = (a_{ij})$ is called **symmetric** if, and only if, for every i and $j = 1, 2, \ldots, n$,

$$a_{ij} = a_{ji}.$$

Example 10.2.5 – *Symmetric Matrices*

Which of the following matrices are symmetric?

a. $\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$

b. $\begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 3 \end{bmatrix}$

c. $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

Example 10.2.5 – *Solution*

Only (b) is symmetric.

In (a) the entry in the first row and the second column differs from the entry in the second row and the first column; the matrix in (c) is not even square.

Matrices and Connected Components

Matrices and Connected Components (1/3)

Consider a graph *G*, as shown below, that consists of several connected components.

Matrices and Connected Components (2/3)

The adjacency matrix of *G* is

$$\mathbf{A} = \left[\begin{array}{ccc:cc:cc} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{array}\right]$$

Matrices and Connected Components (3/3)

Theorem 10.2.1

Let G be a graph with connected components $G_1, G_2, \ldots, G_k$. If there are n_i vertices in each connected component G_i and these vertices are numbered consecutively, then the adjacency matrix of G has the form

$$\begin{bmatrix} A_1 & O & O & \cdots & O & O \\ O & A_2 & O & \cdots & O & O \\ O & O & A_3 & \cdots & O & O \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ O & O & O & \cdots & O & A_k \end{bmatrix}$$

where each A_i is the $n_i \times n_i$ adjacency matrix of G_i, for every $i = 1, 2, \ldots, k$, and the O's represent matrices whose entries are all 0.

Matrix Multiplication

Matrix Multiplication (1/9)

Matrix multiplication is an enormously useful operation that arises in many contexts, including the investigation of walks in graphs.

Although matrix multiplication can be defined in quite abstract settings, the definition for matrices whose entries are real numbers will be sufficient for our applications.

Matrix Multiplication (2/9)

The product of two matrices is built up of *scalar* or *dot* products of their individual rows and columns.

Definition

Suppose that all entries in matrices **A** and **B** are real numbers. If the number of element, n, in the ith row of **A** equals the number of elements in the jth column of **B**, then the **scalar product** or **dot product** of the ith row of **A** and jth column of **B** is the real number obtained as follows:

$$\begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{bmatrix} \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}.$$

Example 10.2.6 – *Multiplying a Row and a Column*

$$\begin{bmatrix} 3 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ 3 \\ 0 \end{bmatrix} = 3 \cdot (-1) + 0 \cdot 2 + (-1) \cdot 3 + 2 \cdot 0$$

$$= -3 + 0 - 3 + 0 = -6$$

Matrix Multiplication (3/9)

More generally, if **A** and **B** are matrices whose entries are real numbers and if **A** and **B** have *compatible sizes* in the sense that the number of columns of **A** equals the number of rows of **B**, then the product **AB** is defined.

It is the matrix whose *i j* th entry is the scalar product of the *i* th row of **A** times the *j* th column of **B**, for all possible values of *i* and *j*.

Matrix Multiplication (4/9)

Definition

Let $\mathbf{A} = (a_{ij})$ be an $m \times k$ matrix and $\mathbf{B} = (b_{ij})$ a $k \times n$ matrix with real entries. The (matrix) product of $\mathbf{A}$ times $\mathbf{B}$, denoted $\mathbf{AB}$, is that matrix (c_{ij}) defined as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ik} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mk} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2n} \\ & \cdot & & \cdot & & \cdot \\ & \cdot & & \cdot & & \cdot \\ & \cdot & & \cdot & & \cdot \\ b_{k1} & b_{k2} & \cdots & b_{kj} & \cdots & b_{kn} \end{bmatrix}$$

$$= \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1j} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2j} & \cdots & c_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ c_{i1} & c_{i2} & \cdots & c_{ij} & \cdots & c_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mj} & \cdots & c_{mn} \end{bmatrix}$$

where

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ik}b_{kj} = \sum_{r=1}^{k} a_{ir}b_{rj},$$

for each $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

Example 10.2.7 – *Computing a Matrix Product*

Let $\mathbf{A} = \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$. Compute $\mathbf{AB}$.

Example 10.2.7 – *Solution (1/3)*

A has size 2×3 and **B** has size 3×2, so the number of columns of **A** equals the number of rows of **B** and the matrix product of **A** and **B** can be computed. Then

$$\begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix},$$

where

$$c_{11} = 2 \cdot 4 + 0 \cdot 2 + 3 \cdot (-2) = 2$$

$$\begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$$

Example 10.2.7 – *Solution (2/3)* continued

$$c_{12} = 2\cdot 3 + 0\cdot 2 + 3\cdot(-1) = 3 \qquad \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$$

$$c_{21} = (-1)\cdot 4 + 1\cdot 2 + 0\cdot(-2) = -2 \qquad \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$$

$$c_{22} = (-1)\cdot 3 + 1\cdot 2 + 0\cdot(-1) = -1 \qquad \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}.$$

Example 10.2.7 – *Solution (3/3)* continued

Hence

$$\mathbf{AB} = \begin{bmatrix} 2 & 3 \\ -2 & -1 \end{bmatrix}.$$

Matrix Multiplication (5/9)

Matrix multiplication is both similar to and different from multiplication of real numbers. One difference is that although the product of any two numbers can be formed, only matrices with compatible sizes can be multiplied. For example, if **A** is a 3×2 matrix and **B** is a 2×4 matrix, then **AB** can be computed because the number of columns of **A** equals the number of rows of **B**.

But **BA** does not exist because **B** has 4 columns, **A** has 3 rows, and 4 ≠ 3.

Matrix Multiplication (6/9)

Another difference is that multiplication of real numbers is commutative (for all real numbers *a* and *b*, *ab* = *ba*), whereas matrix multiplication is not. For instance,

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix} \quad \text{but} \quad \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

On the other hand, both real number and matrix multiplications are associative: $(ab)c = a(bc)$, for all elements *a*, *b*, and *c* for which the products are defined.

Matrix Multiplication (7/9)

As far as multiplicative identities are concerned, there are both similarities and differences between real numbers and matrices. You know that the number 1 acts as a multiplicative identity for products of real numbers.

It turns out that there are certain matrices, called *identity matrices*, that act as multiplicative identities for certain matrix products.

Matrix Multiplication (8/9)

Definition

For each positive integer n, the $\boldsymbol{n \times n}$ **identity matrix**, denoted $\mathbf{I}_n = (\boldsymbol{\delta}_{ij})$ or just $\mathbf{I}$ (if the size of the matrix is obvious from context), is the $n \times n$ matrix in which all the entries in the main diagonal are 1's and all other entries are 0's. In other words,

$$\delta_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}, \quad \text{for every } i, j = 1, 2, \ldots, n.$$

Matrix Multiplication (9/9)

There are also similarities and differences between real numbers and matrices with respect to the computation of powers. Any number can be raised to a nonnegative integer power, but a matrix can be multiplied by itself only if it has the same number of rows as columns.

Definition

For any $n \times n$ matrix $\mathbf{A}$, the powers of $\mathbf{A}$ are defined as follows:

$$\mathbf{A}^0 = \mathbf{I} \quad \text{where } \mathbf{I} \text{ is the } n \times n \text{ identity matrix}$$

$$\mathbf{A}^n = \mathbf{A}\mathbf{A}^{n-1} \quad \text{for every integer } n \geq 1.$$

Example 10.2.10 – *Powers of a Matrix*

Let $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}$. Compute $\mathbf{A}^0$, $\mathbf{A}^1$, $\mathbf{A}^2$, and $\mathbf{A}^3$.

Example 10.2.10 – *Solution*

$$\mathbf{A}^0 = \text{the } 2 \times 2 \text{ identity matrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{A}^1 = \mathbf{A}\mathbf{A}^0 = \mathbf{A}\mathbf{I} = \mathbf{A}$$

$$\mathbf{A}^2 = \mathbf{A}\mathbf{A}^1 = \mathbf{A}\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\mathbf{A}^3 = \mathbf{A}\mathbf{A}^2 = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 9 & 10 \\ 10 & 4 \end{bmatrix}$$

Counting Walks of Length *N*

Counting Walks of Length *N (1/2)*

A walk in a graph consists of an alternating sequence of vertices and edges. If repeated edges are counted each time they occur, then the number of edges in the sequence is called the **length** of the walk. For instance, the walk $v_2e_3v_3e_4v_2e_2v_2e_3v_3$ has length 4 (counting e_3 twice). Consider the following graph G:

Counting Walks of Length *N* (2/2)

If **A** is the adjacency matrix of a graph *G*, the *i j* th entry of $\mathbf{A}^2$ equals the number of walks of length 2 connecting the *i* th vertex to the *j* th vertex of *G*. Even more generally, if *n* is any positive integer, the *i j* th entry of $\mathbf{A}^n$ equals the number of walks of length *n* connecting the *i*th and the *j* th vertices of *G*.

Theorem 10.2.2

If G is a graph with vertices $v_1, v_2, \ldots, v_m$ and $\mathbf{A}$ is the adjacency matrix of G, then for each positive integer n and for all integers $i, j = 1, 2, \ldots, m$,

$$\text{the } ij\text{th entry of } \mathbf{A}^n = \text{the number of walks of length } n \text{ from } v_i \text{ to } v_j.$$

CHAPTER 10

THEORY OF GRAPHS AND TREES

Copyright © Cengage Learning. All rights reserved.

10.3 Isomorphisms of Graphs

Copyright © Cengage Learning. All rights reserved.

Isomorphisms of Graphs (1/3)

Two graphs that are the same except for the labeling of their vertices and edges are called *isomorphic.* The word *isomorphism* comes from the Greek, meaning "same form." Isomorphic graphs are those that have essentially the same form.

Definition

Let G and G' be graphs with vertex sets $V(G)$ and $V(G')$ and edge sets $E(G)$ and $E(G')$, respectively. **G is isomorphic to G'** if, and only if, there exist one-to-one correspondences $g: V(G) \to V(G')$ and $h: E(G) \to E(G')$ that preserve the edge-endpoint functions of G and G' in the sense that for each $v \in V(G)$ and $e \in E(G)$,

$$v \text{ is an endpoint of } e \quad \Leftrightarrow \quad g(v) \text{ is an endpoint of } h(e). \qquad 10.3.1$$

Isomorphisms of Graphs (2/3)

In words, G is isomorphic to G' if, and only if, the vertices and edges of G and G' can be matched up by one-to-one, onto functions in such a way that the edges between corresponding vertices correspond to each other.

Example 10.3.1 – *Showing That Two Graphs Are Isomorphic*

Show that the following two graphs are isomorphic.

G

G'

Example 10.3.1 – *Solution (1/4)*

To solve this problem, you must find functions $g : V(G) \to V(G')$ and $h: E(G) \to E(G')$ such that for each $v \in V(G)$ and $e \in E(G)$, v is an endpoint of e if, and only if, $g(v)$ is an endpoint of $h(e)$.

Setting up such functions is partly a matter of trial and error and partly a matter of deduction.

For instance, since e_2 and e_3 are parallel *[have the same endpoints]*, $h(e_2)$ and $h(e_3)$ must be parallel also.

Example 10.3.1 – *Solution (2/4)* continued

So $h(e_2) = f_1$ and $h(e_3) = f_2$ or $h(e_2) = f_2$ and $h(e_3) = f_1$. Also, the endpoints of e_2 and e_3 must correspond to the endpoints of f_1 and f_2, and so $g(v_3) = w_1$ and $g(v_4) = w_5$ or $g(v_3) = w_5$ and $g(v_4) = w_1$.

Similarly, since v_1 is the endpoint of four distinct edges (e_1, e_7, e_5, and e_4), $g(v_1)$ must also be the endpoint of four distinct edges *[because every edge incident on $g(v_1)$ is the image under h of an edge incident on v_1 and h is one-to-one and onto]*.

Example 10.3.1 – *Solution (3/4)* continued

But the only vertex in G' that has four edges coming out of it is w_2, and so $g(v_1) = w_2$. Now if $g(v_3) = w_1$, then since v_1 and v_3 are endpoints of e_1 in G, $g(v_1) = w_2$ and $g(v_3) = w_1$ must be endpoints of $h(e_1)$ in G'. This implies that $h(e_1) = f_3$.

By continuing in this way, possibly making some arbitrary choices as you go, you eventually can find functions g and h to define the isomorphism between G and G'.

Example 10.3.1 – *Solution (4/4)* continued

One pair of functions (there are several) is the following:

Isomorphisms of Graphs (3/3)

Theorem 10.3.1 Graph Isomorphism Is an Equivalence Relation

Let S be a set of graphs and let R be the relation of graph isomorphism on S. Then R is an equivalence relation on S.

Definition

A property P is called an **invariant for graph isomorphism** if, and only if, given any graphs G and G', if G has property P and G' is isomorphic to G, then G' has property P.

Theorem 10.3.2

Each of the following properties is an invariant for graph isomorphism, where n, m, and k are all nonnegative integers:

1. has n vertices
2. has m edges
3. has a vertex of degree k
4. has m vertices of degree k
5. has a circuit of length k
6. has a simple circuit of length k
7. has m simple circuits of length k
8. is connected
9. has an Euler circuit
10. has a Hamiltonian circuit.

Example 10.3.3 – *Showing That Two Graphs Are Not Isomorphic*

Show that the following pairs of graphs are not isomorphic by finding an isomorphic invariant that they do not share.

a.

b.

Example 10.3.3 – *Solution*

a. G has nine edges; G' has only eight.

b. H has a vertex of degree 4; H' does not.

Graph Isomorphism for Simple Graphs

Graph Isomorphism for Simple Graphs (1/1)

When graphs G and G' are both simple, the definition of G being isomorphic to G' can be written without referring to the correspondence between the edges of G and the edges of G'.

Definition

If G and G' are simple graphs, then **G is isomorphic to G'** if, and only if, there exists a one-to-one correspondence g from the vertex set $V(G)$ of G to the vertex set $V(G')$ of G' that preserves the edge-endpoint functions of G and G' in the sense that for all vertices u and v of G,

$$\{u, v\} \text{ is an edge in } G \quad \Leftrightarrow \quad \{g(u), g(v)\} \text{ is an edge in } G'. \qquad 10.3.2$$

Example 10.3.5 – *Isomorphism of Simple Graphs*

Are the two graphs shown below isomorphic? If so, define an isomorphism.

Example 10.3.5 – *Solution (1/2)*

Yes. Define $g : V(G) \to V(G')$ by the arrow diagram shown below.

Then g is one-to-one and onto by inspection.

Example 10.3.5 – *Solution (2/2)* continued

The fact that g preserves the edge-endpoint functions of G and G' is shown by the following table:

Edges of G	Edges of G'
$\{a, b\}$	$\{y, w\} = \{g(a), g(b)\}$
$\{a, c\}$	$\{y, x\} = \{g(a), g(c)\}$
$\{a, d\}$	$\{y, z\} = \{g(a), g(d)\}$
$\{c, d\}$	$\{x, z\} = \{g(c), g(d)\}$

CHAPTER 10

THEORY OF GRAPHS AND TREES

Copyright © Cengage Learning. All rights reserved.

10.4 Trees: Examples and Basic Properties

Copyright © Cengage Learning. All rights reserved.

Trees: Examples and Basic Properties

In mathematics, a tree is a connected graph that does not contain any circuits. Mathematical trees are similar in certain ways to their botanical namesakes.

Definition

A graph is said to be **circuit-free** if, and only if, it has no circuits. A graph is called a **tree** if, and only if, it is circuit-free and connected. A **trivial tree** is a graph that consists of a single vertex. A graph is called a **forest** if, and only if, it is circuit-free and not connected.

Example 10.4.1 – *Trees and Non-trees (1/2)*

All the graphs shown in Figure 10.4.1 are trees, whereas those in Figure 10.4.2 are not.

Trees. All the graphs in (a)–(d) are connected and circuit-free.

Figure 10.4.1

Example 10.4.1 – *Trees and Non-trees (2/2)*

Non-trees. The graphs in (a), (b), and (c) all have circuits, and the graph in (d) is not connected.

Figure 10.4.2

Examples of Trees

Example 10.4.2 – *A Decision Tree (1/2)*

During orientation week, a college administers a mathematics placement exam to all entering students. The exam consists of two parts, and placement recommendations are made as indicated by the tree shown in Figure 10.4.3.

Figure 10.4.3

Example 10.4.2 – *A Decision Tree (2/2)*

Read the tree from left to right to decide what course should be recommended for a student who scored 9 on part I and 7 on part II.

Example 10.4.2 – *Solution*

Since the student scored 9 on part I, the score on part II is checked.

Since it is greater than 6, the student should be advised to take Math 110.

Example 10.4.3 – *A Parse Tree (1/7)*

In the last 30 years, Noam Chomsky and others have developed new ways to describe the syntax (or grammatical structure) of natural languages such as English.

In the study of grammars, trees are often used to show the derivation of grammatically correct sentences from certain basic rules. Such trees are called **syntactic derivation trees** or **parse trees**.

Example 10.4.3 – *A Parse Tree (2/7)*

A very small subset of English grammar, for example, specifies that

1. a sentence can be produced by writing first a noun phrase and then a verb phrase;
2. a noun phrase can be produced by writing an article and then a noun;
3. a noun phrase can also be produced by writing an article, then an adjective, and then a noun;
4. a verb phrase can be produced by writing a verb and then a noun phrase;

Example 10.4.3 – *A Parse Tree (3/7)*

5. one article is “the”;
6. one adjective is “young”;
7. one verb is “caught”;
8. one noun is “man”;
9. one (other) noun is “ball.”

The rules of a grammar are called **productions**. It is customary to express them using the shorthand notation illustrated on the next slide.

Example 10.4.3 – *A Parse Tree (4/7)*

This notation, introduced by John Backus in 1959 and modified by Peter Naur in 1960, was used to describe the computer language Algol and is called the **Backus–Naur notation**.

In the notation, the symbol | represents the word *or*, and angle brackets ⟨⟩ are used to enclose terms to be defined (such as a sentence or noun phrase).

1. ⟨sentence⟩ → ⟨noun phrase⟩ ⟨verb phrase⟩
2., 3. ⟨noun phrase⟩ → ⟨article⟩ ⟨noun⟩ | ⟨article⟩ ⟨adjective⟩ ⟨noun⟩

Example 10.4.3 – *A Parse Tree (5/7)*

4. ⟨verb phrase⟩ → ⟨verb⟩ ⟨noun phrase⟩

5. ⟨article⟩ → the

6. ⟨adjective⟩ → young

7. ⟨verb⟩ → caught

8., 9. ⟨noun⟩ → man | ball

Example 10.4.3 – *A Parse Tree (6/7)*

The derivation of the sentence “The young man caught the ball” from the above rules is described by the tree shown below.

Example 10.4.3 – *A Parse Tree (7/7)*

In the study of linguistics, **syntax** refers to the grammatical structure of sentences, and **semantics** refers to the meanings of words and their interrelations.

A sentence can be syntactically correct but semantically incorrect, as in the nonsensical sentence “The young ball caught the man,” which can be derived from the rules given on the previous slides.

Or a sentence can contain syntactic errors but not semantic ones, as, for instance, when a two-year-old child says, “Me hungry!”

Characterizing Trees

Characterizing Trees (1/6)

There is a somewhat surprising relation between the number of vertices and the number of edges of a tree.

It turns out that if n is a positive integer, then any tree with n vertices (no matter what its shape) has $n - 1$ edges.

Perhaps even more surprisingly, a partial converse to this fact is also true—namely, any connected graph with n vertices and $n - 1$ edges is a tree.

Characterizing Trees (2/6)

It follows from these facts that if even one new edge (but no new vertex) is added to a tree, the resulting graph must contain a circuit.

Also, from the fact that removing an edge from a circuit does not disconnect a graph, it can be shown that every connected graph has a subgraph that is a tree.

It follows that if n is a positive integer, any graph with n vertices and *fewer* than $n - 1$ edges is not connected.

Characterizing Trees (3/6)

A small but very important fact necessary to derive the first main theorem about trees is that any nontrivial tree must have at least one vertex of degree 1.

Lemma 10.4.1

Any tree that has more than one vertex has at least one vertex of degree 1.

Definition

Let T be a tree. If T has at least two vertices, then a vertex of degree 1 in T is called a **leaf** (or a **terminal vertex**), and a vertex of degree greater than 1 in T is called an **internal vertex** (or a **branch vertex**). The unique vertex in a trivial tree is also called a **leaf** or **terminal vertex**.

Example 10.4.5 – *Leaves and Internal Vertices in Trees*

Find all leaves (or terminal vertices) and all internal (or branch) vertices in the following tree:

Example 10.4.5 – *Solution*

The leaves (or terminal vertices) are v_0, v_2, v_4, v_5, v_7, and v_8.
The internal (or branch) vertices are v_6, v_1, and v_3.

Characterizing Trees (4/6)

Theorem 10.4.2

For any positive integer n, any tree with n vertices has $n - 1$ edges.

Example 10.4.6 – *Determining Whether a Graph Is a Tree*

A graph *G* has ten vertices and twelve edges. Is it a tree?

Example 10.4.6 – *Solution*

No. By Theorem 10.4.2, any tree with ten vertices has nine edges, not twelve.

Theorem 10.4.2

For any positive integer n, any tree with n vertices has $n - 1$ edges.

Example 10.4.7 – *Finding Trees Satisfying Given Conditions*

Find all nonisomorphic trees with four vertices.

Example 10.4.7 – *Solution*

By Theorem 10.4.2, any tree with four vertices has three edges. Thus the total degree of a tree with four vertices must be 6. Also, every tree with more than one vertex has at least two vertices of degree 1.

Thus the following combinations of degrees for the vertices are the only ones possible: 1, 1, 1, 3 and 1, 1, 2, 2. There are two nonisomorphic trees corresponding to both of these possibilities, as shown below.

and

Characterizing Trees (5/6)

Lemma 10.4.3

If G is any connected graph, C is any circuit in G, and any one of the edges of C is removed from G, then the graph that remains is connected.

Theorem 10.4.4

For any positive integer n, if G is a connected graph with n vertices and $n - 1$ edges, then G is a tree.

Example 10.4.8 – *A Graph with n Vertices and n − 1 Edges That Is Not a Tree*

Give an example of a graph with five vertices and four edges that is not a tree.

Example 10.4.8 – *Solution*

By Theorem 10.4.4, such a graph cannot be connected. One example of such an unconnected graph is shown below.

Characterizing Trees (6/6)

Corollary 10.4.5

If G is any graph with n vertices and m edges, where m and n are positive integers and $m \geq n$, then G has a circuit.

CHAPTER 10

THEORY OF GRAPHS AND TREES

Copyright © Cengage Learning. All rights reserved.

10.5 Rooted Trees

Copyright © Cengage Learning. All rights reserved.

Rooted Trees (1/3)

In mathematics, a rooted tree is a tree in which one vertex has been distinguished from the others and is designated the *root*. Given any other vertex *v* in the tree, there is a unique path from the root to *v*.

The number of edges in such a path is called the level of *v*, and the *height* of the tree is the length of the longest such path. It is traditional in drawing rooted trees to place the root at the top and show the branches descending from it.

Rooted Trees (2/3)

Definition

A **rooted tree** is a tree in which there is one vertex that is distinguished from the others and is called the **root**. The **level** of a vertex is the number of edges along the unique path between it and the root. The **height** of a rooted tree is the maximum level of any vertex of the tree. Given the root or any internal vertex v of a rooted tree, the **children** of v are all those vertices that are adjacent to v and are one level farther away from the root than v. If w is a child of v, then v is called the **parent** of w, and two distinct vertices that are both children of the same parent are called **siblings**. Given two distinct vertices v and w, if v lies on the unique path between w and the root, then v is an **ancestor** of w and w is a **descendant** of v.

Rooted Trees (3/3)

These terms are illustrated in Figure 10.5.1.

A Rooted Tree

Figure 10.5.1

Example 10.5.1 – *Rooted Trees (1/2)*

Consider the tree with root v_0 shown below.

a. What is the level of v_5?

b. What is the level of v_0?

c. What is the height of this rooted tree?

d. What are the children of v_3?

e. What is the parent of v_2?

f. What are the siblings of v_8?

g. What are the descendants of v_3?

Example 10.5.1 – *Rooted Trees (2/2)*

h. How many leaves (terminal vertices) are on the tree?

Example 10.5.1 – *Solution*

a. 2

b. 0

c. 3

d. v_5 and v_6

e. v_0

f. v_7 and v_9

g. v_5, v_6, v_{10}

h. 6

Binary Trees

Binary Trees (1/5)

When every vertex in a rooted tree has at most two children and each child is designated either the (unique) left child or the (unique) right child, the result is a *binary tree.*

Definition

A **binary tree** is a rooted tree in which every parent has at most two children. Each child in a binary tree is designated either a **left child** or a **right child** (but not both), and every parent has at most one left child and one right child. A **full binary tree** is a binary tree in which each parent has exactly two children.

Given any parent v in a binary tree T, if v has a left child, then the **left subtree** of v is the binary tree whose root is the left child of v, whose vertices consist of the left child of v and all its descendants, and whose edges consist of all those edges of T that connect the vertices of the left subtree. The **right subtree** of v is defined analogously.

Binary Trees (2/5)

These terms are illustrated in Figure 10.5.2.

A Binary Tree

Figure 10.5.2

Example 10.5.2 – *Representation of Algebraic Expressions (1/2)*

Binary trees are used in many ways in computer science. One use is to represent algebraic expressions with arbitrary nesting of balanced parentheses. For instance, the following (labeled) binary tree represents the expression *a/b*: The operator is at the root and acts on the left and right children of the root in left-to-right order.

Example 10.5.2 – *Representation of Algebraic Expressions (2/2)*

More generally, the binary tree shown below represents the expression $a/(c + d)$. In such a representation, the internal vertices are arithmetic operators, the leaves are variables, and the operator at each vertex acts on its left and right subtrees in left-to-right order.

Draw a binary tree to represent the expression $((a - b) \cdot c) + (d/e)$.

Example 10.5.2 – *Solution*

Binary Trees (3/5)

An interesting theorem about binary trees says that if you know the number of internal vertices of a full binary tree, then you can calculate both the total number of vertices and the number of leaves, and conversely. More specifically, a full binary tree with k internal vertices has a total of $2k + 1$ vertices of which $k + 1$ are leaves.

Theorem 10.5.1

If k is a positive integer and T is a full binary tree with k internal vertices, then (1) T has a total of $2k + 1$ vertices, and (2) T has $k + 1$ leaves.

Example 10.5.3 – *Determining Whether a Certain Full Binary Tree Exists*

Is there a full binary tree that has 10 internal vertices and 13 terminal vertices?

Example 10.5.3 – *Solution*

No. By Theorem 10.5.1, a full binary tree with 10 internal vertices has 10 + 1 = 11 leaves, not 13.

Theorem 10.5.1

If k is a positive integer and T is a full binary tree with k internal vertices, then (1) T has a total of $2k + 1$ vertices, and (2) T has $k + 1$ leaves.

Binary Trees (4/5)

Another interesting theorem about binary trees specifies the maximum number of leaves of a binary tree of a given height. Specifically, the maximum number of leaves of a binary tree of height *h* is 2^h. Another way to say this is that a binary tree with *t* leaves has height of at least $\log_2 t$.

Theorem 10.5.2

For every integer $h \geq 0$, if T is any binary tree with height h and t leaves, then

$$t \leq 2^h.$$

Equivalently: $\log_2 t \leq h.$

Example 10.5.4 – *Determining Whether a Certain Binary Tree Exists*

Is there a binary tree that has height 5 and 38 leaves?

Example 10.5.4 – *Solution*

No. By Theorem 10.5.2, any binary tree T with height 5 has at most $2^5 = 32$ leaves, so such a tree cannot have 38 leaves.

Theorem 10.5.2

For every integer $h \geq 0$, if T is any binary tree with height h and t leaves, then

$$t \leq 2^h.$$

Equivalently: $\log_2 t \leq h.$

Binary Trees (5/5)

Corollary 10.5.3

A full binary tree of height h has 2^h leaves.

Binary Search Trees

Binary Search Trees (1/6)

A binary search tree is a kind of binary tree in which data records, such as customer information, can be stored, searched, and processed very efficiently. To place records into a binary search tree, it must be possible to arrange them in a total order.

In case they do not have a natural total order of their own, an element of a totally ordered set, such as a number or a word and called a **key**, may be added to each record. The keys are inserted into the vertices of the tree and provide access to the records to which they are attached.

Binary Search Trees (2/6)

Once it is built, a binary search tree has the following property: for every internal vertex *v*, all the keys in the left subtree of *v* are less than the key in *v*, and all the keys in the right subtree of *v* are greater than the key in *v*.

For example, check that the following is a binary search tree for the set of records with the following keys: 15, 10, 19, 25, 12, 4.

Binary Search Trees (3/6)

To build a binary search tree, start by making a root and insert a key into it.

To add a new key, compare it to the key at the root. If the new key is less than the key at the root, give the root a left child and insert the new key into it.

If the key is greater than the key at the root, give the root a right child and insert the new key into it.

Binary Search Trees (4/6)

After the first couple of keys have been added, the root and other vertices may already have left and right children.

So to add a key at a subsequent stage, work down the tree to find a place to put the new key, starting at the root and either moving left or right depending on whether the new key is less or greater than the key at the vertex to which it is currently being compared.

Binary Search Trees (5/6)

Algorithm 10.5.1 Building a Binary Search Tree

Input: A totally ordered, nonempty set K of keys

Algorithm Body:
Initialize T to have one vertex, the root, and no edges. Choose a key from K to insert into the root.

while (there are still keys to be added)
  Choose a key, *newkey*, from K to add. Let the root be called v, let $key(v)$ be the key at the root, and let $success = 0$.
    while ($success = 0$)
      if ($newkey < key(v)$)
        then if (v has a left child), call the left child v_L and let $v := v_L$
        else do 1. add a vertex v_L to T as the left child for v
          2. add an edge to T to join v to v_L
          3. insert *newkey* as the key for v_L
          4. let $success := 1$ **end do**

Binary Search Trees (6/6)

continued

Algorithm 10.5.1 Building a Binary Search Tree

```
            if (newkey > key(v))
                then if v has a right child
                    then call the right child v_R, and let v := v_R
                    else do 1. add a vertex v_R to T as the right child for v
                            2. add an edge to T to join v to v_R
                            3. insert newkey as the key for v_R
                            4. let success := 1 end do
        end while
    end while
Output: A binary search tree T for the set K of keys
```

Example 10.5.5 – *Steps for Building a Binary Search Tree*

Go through the steps to build a binary search tree for the keys 15, 10, 19, 25, 12, 4, and insert the keys in the order in which they are listed. For simplicity, use the same names for the vertices and their associated keys.

Example 10.5.5 – *Solution (1/2)*

Insert 15: Make 15 the root.

Insert 10: Compare 10 to 15.

Since 10 < 15 and 15 does not have a left child, make 10 the left child of 15 and add an edge joining 15 and 10.

Insert 19: Compare 19 to 15.

Since 19 > 15 and 15 does not have a right child, make 19 the right child of 15 and add an edge joining 15 and 19.

Example 10.5.5 – *Solution (2/3)* continued

Insert 25: Compare 25 to 15.

Since 25 > 15 and 15 has a right child, namely 19, compare 25 to 19.

Since 25 > 19 and 19 does not have a right child, make 25 the right child of 19 and add an edge joining 19 and 25.

Insert 12: Compare 12 to 15.

Since 12 < 15 and 15 has a left child, namely 10, compare 12 to 10.

Since 12 > 10 and 10 does not have a right child, make 12 the right child of 10 and add an edge joining 10 and 12.

Example 10.5.5 – *Solution (3/3)* continued

Insert 4: Compare 4 to 15.

Since 4 < 15 and 15 has a left child, namely 10, compare 4 to 10.

Since 4 < 10 and 10 does not have a left child, make 4 the left child of 10 and add an edge joining 10 and 4.

The sequence of steps is shown in the following diagrams.

CHAPTER 10

THEORY OF GRAPHS AND TREES

Copyright © Cengage Learning. All rights reserved.

10.6 Spanning Trees and a Shortest Path Algorithm

Copyright © Cengage Learning. All rights reserved.

Spanning Trees and a Shortest Path Algorithm

Definition

A **spanning tree** for a graph G is a subgraph of G that contains every vertex of G and is a tree.

Proposition 10.6.1

1. Every connected graph has a spanning tree.
2. Any two spanning trees for a graph have the same number of edges.

Example 10.6.1 – *Spanning Trees*

Find all spanning trees for the graph *G* pictured below.

Example 10.6.1 – *Solution*

The graph G has one circuit $v_2v_1v_4v_2$, and removing any edge of the circuit gives a tree. Thus, as shown below, there are three spanning trees for G.

Minimum Spanning Trees

Minimum Spanning Trees (1/4)

The graph of the routes allowed by the U.S. Federal Aviation Authority shown in Figure 10.6.1 can be annotated by adding the distances (in miles) between each pair of cities.

Figure 10.6.1

Minimum Spanning Trees (2/4)

This is done in Figure 10.6.3.

Figure 10.6.3

Minimum Spanning Trees (3/4)

Now suppose the airline company wants to serve all the cities shown, but with a route system that minimizes the total mileage of the system as a whole.

Note that such a system is a tree, because if the system contained a circuit, removal of an edge from the circuit would not affect a person’s ability to reach every city in the system from every other, but it would reduce the total mileage of the system.

Minimum Spanning Trees (4/4)

More generally, a graph whose edges are labeled with numbers (known as *weights*) is called a *weighted graph*. A *minimum-weight spanning tree*, or simply a *minimum spanning tree*, is a spanning tree for which the sum of the weights of all the edges is as small as possible.

Definition and Notation

A **weighted graph** is a graph for which each edge has an associated positive real number **weight**. The sum of the weights of all the edges is the **total weight** of the graph. A **minimum spanning tree** for a connected, weighted graph is a spanning tree that has the least possible total weight compared to all other spanning trees for the graph.

If G is a weighed graph and e is an edge of G, then $\boldsymbol{w(e)}$ denotes the weight of e and $\boldsymbol{w(G)}$ denotes the total weight of G.

Kruskal’s Algorithm

Kruskal's Algorithm (1/3)

In Kruskal's algorithm, the edges of a connected, weighted graph are examined one by one in order of increasing weight. At each stage the edge being examined is added to what will become the minimum spanning tree, provided that this addition does not create a circuit.

After $n - 1$ edges have been added (where n is the number of vertices of the graph), these edges, together with the vertices of the graph, form a minimum spanning tree for the graph.

Kruskal's Algorithm (2/3)

Algorithm 10.6.1 Kruskal

Input: *G [a connected, weighted graph with n vertices, where n is a positive integer]*

Algorithm Body:

[Build a subgraph T of G to consist of all the vertices of G with edges added in order of increasing weight. At each stage, let m be the number of edges of T.]

1. Initialize T to have all the vertices of G and no edges.
2. Let E be the set of all the edges of G, and let $m := 0$.
3. **while** $(m < n - 1)$
   - 3a. Find an edge e in E of least weight.
   - 3b. Delete e from E.
   - 3c. **if** addition of e to the edge set of T does not produce a circuit **then** add e to the edge set of T and set $m := m + 1$

   end while

Output: *T [T is a minimum spanning tree for G.]*

Example 10.6.2 – *Action of Kruskal's Algorithm*

Describe the action of Kruskal's algorithm on the graph shown in Figure 10.6.4, where $n = 8$.

Figure 10.6.4

Example 10.6.2 – *Solution (1/2)*

Iteration Number	Edge Considered	Weight	Action Taken
1	Chicago–Milwaukee	74	added
2	Louisville–Cincinnati	83	added
3	Louisville–Nashville	151	added
4	Cincinnati–Detroit	230	added
5	St. Louis–Louisville	242	added
6	St. Louis–Chicago	262	added
7	Chicago–Louisville	269	not added
8	Louisville–Detroit	306	not added
9	Louisville–Milwaukee	348	not added
10	Minneapolis–Chicago	355	added

Example 10.6.2 – *Solution (2/2)* continued

The tree produced by Kruskal's algorithm is shown in Figure 10.6.5.

Figure 10.6.5

Kruskal's Algorithm (3/3)

Theorem 10.6.2 Correctness of Kruskal's Algorithm

When a connected, weighted graph is input to Kruskal's algorithm, the output is a minimum spanning tree.

Prim’s Algorithm

Prim's Algorithm (1/3)

Prim's algorithm works differently from Kruskal's. It builds a minimum spanning tree *T* by expanding outward in connected links from some vertex. One edge and one vertex are added at each stage.

The edge added is the one of least weight that connects the vertices already in *T* with those not in *T*, and the vertex is the endpoint of this edge that is not already in *T*.

Prim's Algorithm (2/3)

Algorithm 10.6.2

Input: *G [a connected, weighted graph with n vertices where n is a positive integer]*

Algorithm Body:
[Build a subgraph T of G by starting with any vertex v of G and attaching edges (with their endpoints) one by one to an as-yet-unconnected vertex of G, each time choosing an edge of least weight that is adjacent to a vertex of T.]

1. Pick a vertex v of G and let T be the graph with one vertex, v, and no edges.
2. Let V be the set of all vertices of G except v.
3. **for** $i := 1$ **to** $n - 1$
   - 3a. Find an edge e of G such that (1) e connects T to one of the vertices in V, and (2) e has the least weight of all edges connecting T to a vertex in V. Let w be the endpoint of e that is in V.
   - 3b. Add e and w to the edge and vertex sets of T, and delete w from V.

   next i

Output: *T [T is a minimum spanning tree for G.]*

Example 10.6.3 – *Action of Prim's Algorithm*

Describe the action of Prim's algorithm for the graph in Figure 10.6.6 using the Minneapolis vertex as a starting point.

Figure 10.6.6

Example 10.6.3 – *Solution*

Iteration Number	Vertex Added	Edge Added	Weight
0	Minneapolis		
1	Chicago	Minneapolis–Chicago	355
2	Milwaukee	Chicago–Milwaukee	74
3	St. Louis	Chicago–St. Louis	262
4	Louisville	St. Louis–Louisville	242
5	Cincinnati	Louisville–Cincinnati	83
6	Nashville	Louisville–Nashville	151
7	Detroit	Cincinnati–Detroit	230

Prim's Algorithm (3/3)

Theorem 10.6.3 Correctness of Prim's Algorithm

When a connected, weighted graph G is input to Prim's algorithm, the output is a minimum spanning tree for G.

Example 10.6.4 – *Finding Minimum Spanning Trees*

Find all minimum spanning trees for the following graph. Use Kruskal's algorithm and Prim's algorithm starting at vertex *a*. Indicate the order in which edges are added to form each tree.

Example 10.6.4 – *Solution (1/2)*

When Kruskal's algorithm is applied, edges are added in one of the following two orders:

1. $\{d, f\}$, $\{a, c\}$, $\{a, b\}$, $\{c, d\}$, $\{d, e\}$
2. $\{d, f\}$, $\{a, c\}$, $\{b, c\}$, $\{c, d\}$, $\{d, e\}$

When Prim's algorithm is applied starting at a, edges are added in one of the following two orders:

1. $\{a, c\}$, $\{a, b\}$, $\{c, d\}$, $\{d, f\}$, $\{d, e\}$
2. $\{a, c\}$, $\{b, c\}$, $\{c, d\}$, $\{d, f\}$, $\{d, e\}$

Example 10.6.4 – *Solution (2/2)* continued

Thus, as shown below, there are two distinct minimum spanning trees for this graph.

(a)

(b)

Dijkstra’s Shortest Path Algorithm

Dijkstra’s Shortest Path Algorithm (1/5)

Although the trees produced by Kruskal’s and Prim’s algorithms have the least possible total weight compared to all other spanning trees for the given graph, they do not always reveal the shortest distance between any two points on the graph.

In 1959 the computing pioneer, Edsger Dijkstra developed an algorithm to find the shortest path between a starting vertex and an ending vertex in a weighted graph in which all the weights are positive.

Dijkstra's Shortest Path Algorithm (2/5)

It is somewhat similar to Prim's algorithm in that it works outward from a starting vertex *a*, adding vertices and edges one by one to construct a tree *T*.

However, it differs from Prim's algorithm in the way it chooses the next vertex to add, ensuring that for each added vertex *v*, the length of the shortest path from *a* to *v* has been identified.

Dijkstra's Shortest Path Algorithm (3/5)

Algorithm 10.6.3 Dijkstra

Input: *G [a connected simple graph with a positive weight for every edge], ∞ [a number greater than the sum of the weights of all the edges in the graph], $w(u, v)$ [the weight of edge $\{u, v\}$], a [the starting vertex], z [the ending vertex]*

Algorithm Body:

1. Initialize T to be the graph with vertex a and no edges. Let $V(T)$ be the set of vertices of T, and let $E(T)$ be the set of edges of T.
2. Let $L(a) = 0$, and for all vertices in G except a, let $L(u) = \infty$.
   [The number $L(x)$ is called the label of x.]
3. Initialize v to equal a and F to be $\{a\}$.
   [The symbol v is used to denote the vertex most recently added to T.]

Dijkstra's Shortest Path Algorithm (4/5)

Algorithm 10.6.3 Dijkstra

4. **while** ($z \notin V(T)$)

4a. $F := (F - \{v\}) \cup$ {vertices that are adjacent to v and are not in $V(T)$}
[The set F is called the fringe. Each time a vertex is added to T, it is removed from the fringe and the vertices adjacent to it are added to the fringe if they are not already in the fringe or the tree T.]

4b. For each vertex u that is adjacent to v and is not in $V(T)$,
if $L(v) + w(v, u) < L(u)$ **then**

$$L(u) := L(v) + w(v, u)$$
$$D(u) := v$$

[Note that adding v to T does not affect the labels of any vertices in the fringe F except those adjacent to v. Also, when L(u) is changed to a smaller value, the notation D(u) is introduced to keep track of which vertex in T gave rise to the smaller value.]

4c. Find a vertex x in F with the smallest label
Add vertex x to $V(T)$, and add edge $\{D(x), x\}$ to $E(T)$
$v := x$ *[This statement sets up the notation for the next iteration of the loop.]*

end while

Output: $L(z)$ *[L(z), a nonnegative integer, is the length of the shortest path from a to z.]*

Example 10.6.5 – *Action of Dijkstra's Algorithm*

Show the steps in the execution of Dijkstra's shortest path algorithm for the graph shown below with starting vertex *a* and ending vertex *z*.

Example 10.6.5 – *Solution (1/7)*

Step 1: Going into the **while** loop: $V(T) = \{a\}$, $E(T) = \varnothing$, and $F = \{a\}$

During iteration:

$F = \{b, c\}$, $L(b) = 3$, $L(c) = 4$.

Since $L(b) < L(c)$, b is added to $V(T)$, $D(b) = a$, and $\{a, b\}$ is added to $E(T)$.

Example 10.6.5 – *Solution (2/7)* continued

Step 2: Going into the **while** loop: $V(T) = \{a, b\}$,

$E(T) = \{\{a, b\}\}$

During iteration:

$F = \{c, d, e\}$, $L(c) = 4$, $L(d) = 9$, $L(e) = 8$.

Since $L(c) < L(d)$ and $L(c) < L(e)$, c is added to $V(T)$, $D(c) = a$, and $\{a, c\}$ is added to $E(T)$.

Example 10.6.5 – *Solution (3/7)* continued

Step 3: Going into the **while** loop: $V(T) = \{a, b, c\}$,

$E(T) = \{\{a, b\}, \{a, c\}\}$

During iteration:

$F = \{d, e\}$, $L(d) = 9$, $L(e) = 5$

$L(e)$ becomes 5 because *ace*, which has length 5, is a shorter path to *e* than *abe*, which has length 8.

Since $L(e) < L(d)$, *e* is added to $V(T)$, $D(e) = c$, and $\{c, e\}$ is added to $E(T)$.

Example 10.6.5 – *Solution (4/7)* continued

Step 4: Going into the **while** loop: $V(T) = \{a, b, c, e\}$,

$E(T) = \{\{a, b\}, \{a, c\}, \{c, e\}\}$

During iteration:

$F = \{d, z\}$, $L(d) = 7$, $L(z) = 17$

$L(d)$ becomes 7 because *aced*, which has length 7, is a shorter path to d than *abd*, which has length 9.

Since $L(d) < L(z)$, d is added to $V(T)$, $D(d) = e$, and $\{e, d\}$ is added to $E(T)$.

Example 10.6.5 – *Solution (5/7)* continued

Step 5: Going into the **while** loop: $V(T) = \{a, b, c, e, d\}$,

$E(T) = \{\{a, b\}, \{a, c\}, \{c, e\}, \{e, d\}\}$

During iteration:

$F = \{z\}$, $L(z) = 14$

$L(z)$ becomes 14 because *acedz*, which has length 14, is a shorter path to *d* than *abdz*, which has length 17.

Since *z* is the only vertex in *F*, its label is a minimum, and so *z* is added to $V(T)$, $D(z) = d$, and $\{d, z\}$ is added to $E(T)$.

Example 10.6.5 – *Solution (6/7)* continued

Execution of the algorithm terminates at this point because $z \in V(T)$. The shortest path from a to z has length $L(z) = 14$.

Keeping track of the steps in a table is a convenient way to show the action of the algorithm. Table 10.6.1 does this for the g

Step	$V(T)$	$E(T)$	F	$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$	$L(z)$
0	$\{a\}$	$\varnothing$	$\{a\}$	**0**	∞	∞	∞	∞	∞
1	$\{a\}$	$\varnothing$	$\{b, c\}$	0	**3**	4	∞	∞	∞
2	$\{a, b\}$	$\{\{a, b\}\}$	$\{c, d, e\}$	0	3	**4**	9	8	∞
3	$\{a, b, c\}$	$\{\{a, b\}, \{a, c\}\}$	$\{d, e\}$	0	3	4	9	**5**	∞
4	$\{a, b, c, e\}$	$\{\{a, b\}, \{a, c\}, \{c, e\}\}$	$\{d, z\}$	0	3	4	**7**	5	17
5	$\{a, b, c, e, d\}$	$\{\{a, b\}, \{a, c\}, \{c, e\}, \{e, d\}\}$	$\{z\}$	0	3	4	7	5	**14**
6	$\{a, b, c, e, d, z\}$	$\{\{a, b\}, \{a, c\}, \{c, e\}, \{e, d\}, \{e, z\}\}$							

Table 10.6.1

Example 10.6.5 – *Solution (7/7)* continued

In step 1, $D(b) = a$; in step 2, $D(c) = a$; in step 3, $D(e) = c$; in step 4, $D(d) = e$; and in step 5, $D(z) = e$.

Working backward gives the vertices in the shortest path. Because $D(z) = d$, $D(d) = e$, $D(e) = c$, and $D(c) = a$, the shortest path from a to z is $acedz$.

Dijkstra's Shortest Path Algorithm (5/5)

Theorem 10.6.4 Correctness of Dijkstra's Algorithm

When a connected, simple graph with a positive weight for every edge is input to Dijkstra's algorithm with starting vertex a and ending vertex z, the output is the length of a shortest path from a to z.

CHAPTER 11

ANALYSIS OF ALGORITHM EFFICIENCY

Copyright © Cengage Learning. All rights reserved.

11.1 Real-Valued Functions of a Real Variable and Their Graphs

Copyright © Cengage Learning. All rights reserved.

Real-Valued Functions of a Real Variable and Their Graphs (1/4)

A **Cartesian plane** or **two-dimensional Cartesian coordinate system** is a pictorial representation of **R** × **R**, obtained by setting up a one-to-one correspondence between ordered pairs of real numbers and points in a Euclidean plane.

To obtain it, two perpendicular lines, called the **horizontal** and **vertical axes**, are drawn in the plane. Their point of intersection is called the **origin**, and a unit of distance is chosen for each axis.

Real-Valued Functions of a Real Variable and Their Graphs (2/4)

An ordered pair (x, y) of real numbers corresponds to the point P that lies $|x|$ units to the right or left of the vertical axis and $|y|$ units above or below the horizontal axis. On each axis the positive direction is marked with an arrow.

Definition

A **real-valued function of a real variable** is a function from one set of real numbers to another. If f is a real-valued function of a real variable, then for each real number x in the domain of f there is a unique corresponding real number $f(x)$. The **graph of f** is the set of all points (x, y) in the Cartesian coordinate plane with the property that x is in the domain of f and $y = f(x)$.

Real-Valued Functions of a Real Variable and Their Graphs (3/4)

The definition of graph (see Figure 11.1.1) means that for each *x* in the domain of *f*:

$$y = f(x) \quad \Leftrightarrow \quad \text{the point } (x, y) \text{ lies on the graph of } f.$$

Graph of a Function *f*

Figure 11.1.1

Real-Valued Functions of a Real Variable and Their Graphs (4/4)

Note that if $f(x)$ can be written as an algebraic expression in x, the graph of the function f is the same as the graph of the equation $y = f(x)$ where x is restricted to lie in the domain of f.

Power Functions

Power Functions (1/1)

A function that sends a real number *x* to a particular power, x^a, is called a *power function.* For applications in computer science, we are almost invariably concerned with situations where *x* and *a* are nonnegative, and so we restrict our definition to these cases.

Definition

Let a be any nonnegative real number. Define p_a, the **power function with exponent *a***, as follows:

$$p_a(x) = x^a \quad \text{for each nonnegative real number } x.$$

Example 11.1.1 – *Graphs of Power Functions*

Sketch the graphs of the power functions p_0, $p_{1/2}$, p_1, and p_2 on the same coordinate axes.

Example 11.1.1 – *Solution (1/3)*

Because the power function with exponent zero satisfies $p_0(x) = x^0 = 1$ for every nonnegative number x, all points of the form $(x, 1)$ lie on the graph of p_0 for every $x \geq 0$. So the graph is just a horizontal half-line of height 1 lying above the horizontal axis.

Similarly, $p_1(x) = x$ for every nonnegative number x, and so the graph of p_1 consists of all points of the form (x, x) where x is nonnegative. The graph is therefore the half-line of slope 1 that emanates from (0, 0).

Example 11.1.1 – *Solution (2/3)* continued

Since for each nonnegative number x, $p_{1/2}(x) = x^{1/2} = \sqrt{x}$, any point with coordinates $(x, \sqrt{x})$, where x is nonnegative, is on the graph of $p_{1/2}$. For instance, the graph of $p_{1/2}$ contains the points (0, 0), (1, 1), (4, 2), and (9, 3). Similarly, since $p_2(x) = x^2$, any point with coordinates (x, x^2) lies on the graph of p_2.

Thus, for instance, the graph of p_2 contains the points (0, 0), (1, 1), (2, 4), and (3, 9).

Example 11.1.1 – *Solution (3/3)* continued

The graphs of all four functions are shown in Figure 11.1.2.

Graphs of Some Power Functions

Figure 11.1.2

The Floor Function

The Floor Function (1/1)

The floor and ceiling functions arise in many computer science contexts. Example 11.1.2 illustrates the graph of the floor function.

Example 11.1.2 – *Graph of the Floor Function (1/2)*

Recall that each real number either is an integer itself or sits between two consecutive integers: For each real number x, there exists a unique integer n such that $n \leq x < n + 1$.

The floor of a number is the integer immediately to its left on the number line.

Example 11.1.2 – *Graph of the Floor Function (2/2)*

More formally, the floor function F is defined by the rule
For each real number x,

$$F(x) = \lfloor x \rfloor$$

= the greatest integer that is less than or equal to x

= the unique integer n such that $n \leq x < n + 1$.

Sketch a graph of the floor function.

Example 11.1.2 – *Solution (1/2)*

If n is any integer, then for each real number x in the interval $n \leq x < n + 1$, the floor of x, $\lfloor x \rfloor$, equals n.

Thus on each such interval, the graph of the floor function is horizontal; for each x in the interval, the height of the graph is n.

Example 11.1.2 – *Solution (2/2)*

It follows that the graph of the floor function consists of horizontal line segments, like a staircase, as shown in Figure 11.1.3.

Graph of the Floor Function

Figure 11.1.3

The open circles at the right-hand edge of each step are used to show that those points are *not* on the graph.

Graphing Functions Defined on Sets of Integers

Graphing Functions Defined on Sets of Integers (1/1)

Many real-valued functions used in computer science are defined on sets of integers rather than on intervals of real numbers.

But if you know what the graph of a function looks like when it is given by a formula on an interval of real numbers, you can obtain the graph of the function that is defined on the integers in the interval using the same formula by selecting only the points on the known graph whose first coordinates are integers.

Example 11.1.3 – *Graph of a Function Defined on a Set of Integers*

Consider an integer version of the power function $p_{1/2}$. In other words, define a function g by the formula $g(n)=n^{1/2}$ for each nonnegative integer n. Sketch the graph of g.

Example 11.1.3 – *Solution (1/2)*

Look back at the graph of $p_{1/2}$ in Figure 11.1.2.

Graphs of Some Power Functions

Figure 11.1.2

Example 11.1.3 – *Solution (2/2)* continued

Draw the graph of g by reproducing only those points on the graph of $p_{1/2}$ with integer first coordinates. Thus for each nonnegative integer n, the point $(n, n^{1/2})$ is on the graph of g.

Graph of g where $g(n) = n^{1/2}$ for each nonnegative integer n

Graph of a Multiple of a Function

Graph of a Multiple of a Function (1/1)

A *multiple* of a function is obtained by multiplying every value of the function by a fixed number. To understand the concept of *O*-notation, it is helpful to understand the relation between the graph of a function and the graph of a multiple of the function.

Definition

Let f be a real-valued function of a real variable and let M be any real number. The function $M f$, called the **multiple of f by M** or **M times f**, is the real-valued function with the same domain as f that is defined by the rule

$$(M f)(x) = M \cdot (f(x)) \quad \text{for each } x \in \text{domain of } f.$$

Example 11.1.4 – *Graph of a Multiple of a Function*

Let *f* be the function whose graph is shown below. Sketch the graph of 2*f*.

Example 11.1.4 – *Solution (1/2)*

At each real number x, you obtain the height of the graph of $2f$ by measuring the height of the graph of f at x and multiplying that number by 2. The result is the following graph.

Example 11.1.4 – *Solution (2/2)* continued

Note that the general shapes of f and $2f$ are very similar, but the graph of $2f$ is “stretched out”: the “highs” are twice as high and the “lows” are twice as low.

Increasing and Decreasing Functions

Increasing and Decreasing Functions (1/4)

Consider the *absolute value function*, *A*, which is defined as follows:

$$A(x) = |x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases} \qquad \text{for each real number } x.$$

When $x \geq 0$, the graph of A is the same as the graph of $y = x$, the straight line with slope 1 that passes through the origin (0, 0).

Increasing and Decreasing Functions (2/4)

For $x < 0$, the graph of A is the same as the graph of $y = -x$, which is the straight line with slope -1 that passes through (0, 0). (See Figure 11.1.4.)

Graph of the Absolute Value Function

Figure 11.1.4

Increasing and Decreasing Functions (3/4)

Note that as you trace from left to right along the graph to the left of the origin, the height of the graph continually *decreases.* For this reason, the absolute value function is said to be *decreasing* on the set of real numbers less than 0.

On the other hand, as you trace from left to right along the graph to the right of the origin, the height of the graph continually *increases.* Consequently, the absolute value function is said to be *increasing* on the set of real numbers greater than 0.

Increasing and Decreasing Functions (4/4)

Since the height of the graph of a function f at a point x is $f(x)$, these geometric concepts translate to the following analytic definition.

Definition

Let f be a real-valued function defined on a set of real numbers, and suppose the domain of f contains a set S. We say that f is **increasing on the set S** if, and only if,

$$\text{for all real numbers } x_1 \text{ and } x_2 \text{ in } S, \text{ if } x_1 < x_2 \text{ then } f(x_1) < f(x_2).$$

We say that f is **decreasing on the set S** if, and only if,

$$\text{for all real numbers } x_1 \text{ and } x_2 \text{ in } S, \text{ if } x_1 < x_2 \text{ then } f(x_1) > f(x_2).$$

We say that f is an **increasing** (or **decreasing**) **function** if, and only if, f is increasing (or decreasing) on its entire domain.

Example 11.1.5 – *A Positive Multiple of an Increasing Function Is Increasing*

Suppose that *f* is a real-valued function of a real variable that is increasing on a set *S* of real numbers, and suppose *M* is any positive real number. Show that *M f* is also increasing on *S.*

Figure 11.1.5

Example 11.1.5 – *Solution (1/2)*

Suppose x_1 and x_2 are particular but arbitrarily chosen elements of S such that

$$x_1 < x_2.$$

[We must show that $(M\ f)(x_1) < (M\ f)\ (x_2)$*.]*

From the facts that $x_1 < x_2$ and f is increasing, it follows that

$$f(x_1) < f(x_2).$$

Example 11.1.5 – *Solution (2/2)* continued

Then

$$M f(x_1) < M f(x_2),$$

since multiplying both sides of the inequality by a positive number does not change the direction of the inequality.

Hence, by definition of $M f$,

$$(M f)(x_1) < (M f)(x_2),$$

and, consequently, $M f$ is increasing on S.

CHAPTER 11

ANALYSIS OF ALGORITHM EFFICIENCY

Copyright © Cengage Learning. All rights reserved.

11.2 Big-*O*, Big-Omega, and Big-Theta Notations

Copyright © Cengage Learning. All rights reserved.

Big-*O*, Big-Omega, and Big-Theta Notations (1/7)

The symbols Θ and Ω are the uppercase Greek letters theta and omega. The Θ-notation introduced in this section is determined by ranges such as those shown in Figure 11.2.1.

Figure 11.2.1

Big-*O*, Big-Omega, and Big-Theta Notations (2/7)

It is related to two other notations: Ω-notation and *O*-notation. The oldest of the notations, *O*-notation (read “big-*O* notation”), was introduced by the German mathematician Paul Bachmann in 1894 in a book on analytic number theory, and it was the first to be used to compare efficiencies of algorithms.

As you will see, however, using *O*-notation alone can produce ambiguous or even misleading results.

Big-*O*, Big-Omega, and Big-Theta Notations (3/7)

As a response, in 1976 Donald Knuth, a pioneer in the analysis of algorithms, introduced the Ω- and Θ-notations so that the growth of functions could be compared with greater precision.

The idea of the notations is this: Suppose f and g are real-valued functions of an integer variable n and suppose $g(n) \geq 0$ for every integer n greater than some positive real number.

Big-*O*, Big-Omega, and Big-Theta Notations (4/7)

1. If, for sufficiently large values of n, the values of f are greater than those of a positive multiple of g, then f is of order *at least* g, written "$f(n)$ is $\Omega(g(n))$."

2. If, for sufficiently large values of n, the values of f are positive and less than those of a positive multiple of g, then f is of order *at most* g, written "$f(n)$ is $O(g(n))$."

3. If, for sufficiently large values of n, the values of f are bounded both above and below by those of positive multiples of g, then f is of order g, written "$f(n)$ is $\Theta(g(n))$."

Big-*O*, Big-Omega, and Big-Theta Notations (5/7)

These relationships are illustrated in Figure 11.2.2.

Figure 11.2.2

Big-O, Big-Omega, and Big-Theta Notations (6/7)

Definition

Let f and g be real-valued functions defined on the same set of nonnegative integers, with $g(n) \geq 0$ for every integer $n \geq r$, where r is a positive real number. Then

1. $\boldsymbol{f}$ is of order at least $\boldsymbol{g}$, written $\boldsymbol{f(n)}$ **is** $\boldsymbol{\Omega(g(n))}$ (f of n is big-Omega of g of n), if, and only if, there exist positive real numbers A and $a \geq r$ such that

$$Ag(n) \leq f(n) \quad \text{for every integer } n \geq a.$$

2. $\boldsymbol{f}$ is of order at most $\boldsymbol{g}$, written $\boldsymbol{f(n)}$ **is** $\boldsymbol{O(g(n))}$ (f of n is big-O of g of n), if, and only if, there exist positive real numbers B and $b \geq r$ such that

$$0 \leq f(n) \leq Bg(n) \quad \text{for every integer } n \geq b.$$

3. $\boldsymbol{f}$ is of order $\boldsymbol{g}$, written $\boldsymbol{f(n)}$ **is** $\boldsymbol{\Theta(g(n))}$ (f of n is big-Theta of g of n), if, and only if, there exist positive real numbers A, B, and $k \geq r$ such that

$$Ag(n) \leq f(n) \leq Bg(n) \quad \text{for every integer } n \geq k.$$

Example 11.2.1 – *Translating to Θ-Notation*

Use Θ-notation to express the statement

$4n^6 \leq 17n^6 - 45n^3 + 2n + 8 \leq 30n^6$ for every integer $n \geq 3$.

Example 11.2.1 – *Solution*

Let $A = 4$, $B = 30$, and $k = 3$. Then the statement translates to

$$An^6 \leq 17n^6 - 45n^3 + 2n + 8 \leq Bn^6 \quad \text{for every integer } n \geq k.$$

So, by definition of Θ-notation,

$$17n^6 - 45n^3 + 2n + 8 \quad \text{is} \quad \Theta(n^6).$$

Example 11.2.2 – *Translating to Ω- and O-Notations and Deducing Θ-Notation*

a. Use Ω-notation to express the statement

$$\frac{11}{4}n^2 \le 3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \qquad \text{for every integer } n \ge 2.$$

b. Use *O*-notation to express the statement

$$0 \le 3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \le 6n^2 \qquad \text{for every integer } n \ge 1.$$

c. Justify the statement: $3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \text{ is } \Theta(n^2).$

Example 11.2.2 – *Solution (1/3)*

a. Let $A = \frac{11}{4}$ and *a* = 2. The statement in (a) translates to

$$An^2 \leq 3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \qquad \text{for every integer } n \geq a.$$

So, by definition of Ω-notation,

$$3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \text{ is } \Omega(n^2).$$

Example 11.2.2 – *Solution (2/3)* continued

b. Let $B = 6$ and $b = 1$. The statement in (b) translates to

$$0 \le 3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \le Bn^2 \qquad \text{for every integer } n \ge b.$$

So, by definition of O-notation,

$$3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \ \text{ is } \ O(n^2).$$

Example 11.2.2 – *Solution (3/3)* continued

c. Let $A = \frac{11}{4}$, $a = 2$, $B = 6$, and $b = 1$, and let k be the larger of a and b. Then when $n \geq k$, both inequalities in parts (a) and (b) are satisfied, and so

$$An^2 \leq 3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \leq Bn^2 \quad \text{for every integer } n \geq k.$$

Hence by definition of Θ-notation,

$$3\left(\left\lfloor \frac{n}{4} \right\rfloor\right)^2 + 5n^2 \quad \text{is} \quad \Theta(n^2).$$

Big-O, Big-Omega, and Big-Theta Notations (7/7)

Theorem 11.2.1 Relation among O-, Ω-, and Θ-Notations

If f and g are real-valued functions defined on the same set of nonnegative integers, and if $f(n) \geq 0$ and $g(n) \geq 0$ for every integer $n \geq r$, where r is a positive real number,

then $f(n)$ is $\Theta(g(n))$ if, and only if, $f(n)$ is $\Omega(g(n))$ and $f(n)$ is $O(g(n))$.

Orders of Power Functions

Orders of Power Functions (1/2)

The functions that are most commonly used for comparing algorithm efficiencies are power functions, such as $n^{1/2}$, n, n^2, and n^3, and combinations involving a power function and an exponential or logarithmic function, such as 2^n, log(n), n log(n), and $n^2 \log(n)$.

These functions arise naturally in the analysis of algorithms, but, for large values of n, their sizes are dramatically different. In this section we focus on power functions.

Orders of Power Functions (2/2)

Theorem 11.2.2 For any positive rational numbers *r* and *s* and any integer $n \geq 1$,

$$\text{if } r \leq s, \quad \text{then} \quad n^r \leq n^s.$$

Orders of Polynomial Functions

Orders of Polynomial Functions (1/6)

The following two examples show how to use Theorem 11.2.2 to find a big-Ω and a big-*O* for some polynomial functions.

Graphs of Powers of *n* for *n* ≥ 1

Figure 11.2.3

Example 11.2.3 – *Finding a Big-Omega and a Big-O for a Polynomial Function with Nonnegative Coefficients*

a. Show that $15n^3 + 11n^2 + 9$ is $\Omega(n^3)$.

b. Without using part (a), show that $15n^3 + 11n^2 + 9$ is $O(n^3)$.

Example 11.2.3 – *Solution (1/4)*

a. To show that $15n^3 + 11n^2 + 9$ is $\Omega(n^3)$, you need to show that $15n^3 + 11n^2 + 9$ is greater than or equal to a positive multiple of n^3 for all values of *n* that are sufficiently large. Now

$$15n^3 \leq 15n^3 + 11n^2 + 9 \qquad \text{for every integer } n \geq 1.$$

because when $n \geq 1$, $11n^2 + 9$ is positive. Thus you can let $A = 15$ and $a = 1$ to obtain

$$An^3 \leq 15n^3 + 11n^2 + 9 \qquad \text{for every integer } n \geq a.$$

and conclude, by definition of Ω-notation, that

$$15n^3 + 11n^2 + 9 \text{ is } \Omega(n^3).$$

Example 11.2.3 – *Solution (2/4)* continued

b. To show that $15n^3 + 11n^2 + 9$ is $O(n^3)$, you need to show that $15n^3 + 11n^2 + 9$ is greater than or equal to 0 and less than or equal to some positive multiple of n^3 for all values of *n* that are sufficiently large.

First note that because all terms of $15n^3 + 11n^2 + 9$ are positive,

$$0 \leq 15n^3 + 11n^2 + 9 \qquad \text{for every integer } n \geq 1.$$

Example 11.2.3 – *Solution (3/4)* continued

Next observe that for every integer $n \geq 1$,

$$15n^3 + 11n^2 + 9 \leq 15n^3 + 11n^3 + 9n^3$$

because, by Theorem 11.2.2, since $n \geq 1$, then $n \leq n^3$ and $1 \leq n^3$, and so $11n \leq 11n^3$ and $9 \leq 9n^3$

$$= 35n^3$$

because 15 + 11 + 9 = 35.

Thus, by transitivity of order and equality,

$$0 \leq 15n^3 + 11n + 7 \leq 35n^3$$

for every integer $n \geq 1$.

Example 11.2.3 – *Solution (4/4)* continued

Let $B = 35$ and $b = 1$. Then

$$0 \le 15n^3 + 11n^2 + 9 \le Bn^3 \qquad \text{for every integer } n \ge b.$$

and so, by definition of O-notation,

$$15n^3 + 11n^2 + 9 \text{ is } O(n^3).$$

Orders of Polynomial Functions (2/6)

Let m be a nonnegative integer, let $P(n)$ be a polynomial of degree m, and suppose the coefficient a_m of n^m is positive.

To find big-Omega for $P(n)$: Let $A = \frac{1}{2}a_m$, and let a be the number obtained as follows:

1. Find the sum of the absolute values of all the coefficients of $P(n)$ except for a_m.
2. Multiply the result of step 1 by $\dfrac{2}{a_m}$.
3. Let a be the larger of the number 1 and the result of step 2.

Show that $An^m \leq P(n)$ for every integer $n \geq a$.

Example 11.2.5 – *Finding a Big-Theta for a Polynomial Function*

a. Show that $15n^3 + 11n^2 + 9$ is $\Theta(n^3)$.

b. Show that $n^4 - 5n - 8$ is $\Theta(n^4)$.

Example 11.2.5 – *Solution*

a. By Example 11.2.3, $15n^3 + 11n^2 + 9$ is $O(n^3)$ and by Example 11.2.4, $15n^3 + 11n^2 + 9$ is $\Omega(n^3)$. Thus by Theorem 11.2.1, $15n^3 + 11n^2 + 9$ is $\Theta(n^3)$.

b. By Example 11.2.4, $n^4 - 5n - 8$ is both $\Omega(n^4)$ and $O(n^4)$. Thus, by Theorem 11.2.1, $n^4 - 5n - 8$ is $\Theta(n^4)$.

Orders of Polynomial Functions (3/6)

Theorem 11.2.3 A Limit on What Can Be Inferred from Big-*O*

For any function f and positive real numbers r and s with $r < s$,

$$\text{if } f(n) \text{ is } O(n^r) \text{ then } f(n) \text{ is } O(n^s).$$

Example 11.2.6 – *A Caution about O-notation*

Suppose a person finds that $f(n)$ is $O(n^5)$ and that $g(n)$ is $O(n^4)$. Because 4 < 5, this person might conclude that the graph of g lies below the graph of f for large values of n. Show that this is not necessarily the case by showing that there exist functions f and g such that $f(n)$ is $O(n^5)$ and $g(n)$ is $O(n^4)$, yet for large values of n the graph of f lies below, not above, the graph of g.

Example 11.2.6 – *Solution (1/4)*

Let $f(n) = 15n^3 + 11n^2 + 9$ and $g(n) = n^4 - 5n - 8$. Since $f(n)$ is $O(n^3)$, it follows from Theorem 11.2.3 that $f(n)$ is $O(n^5)$. Also by Example 11.2.3, $g(n)$ is $O(n^4)$. However, Figure 11.2.4, which shows both f and g, suggests that for values of $n \geq 16$, the graph of g lies above the graph of f.

Figure 11.2.4

Example 11.2.6 – *Solution (2/4)* continued

This result can be confirmed analytically by noting that

$$(*) \qquad g(n) \geq f(n)$$

$$\Leftrightarrow \qquad n^4 - 5n - 8 \geq 15n^3 + 11n^2 + 9$$

$$\Leftrightarrow \qquad n^4 \geq 15n^3 + 11n^2 + 5n + 17$$

$$(**) \Leftrightarrow \qquad n \geq 15 + \frac{11}{n} + \frac{5}{n^2} + \frac{17}{n^3}$$

Example 11.2.6 – *Solution (3/4)* continued

So any value of n that satisfies inequality (**) also satisfies the inequality (*). Now when $n \geq 16$, then $\frac{1}{n} \leq \frac{1}{16}$, and so inequality (**) is satisfied because

$$15 + \frac{11}{n} + \frac{5}{n^2} + \frac{1}{n^3} \leq 15 + \frac{11}{16} + \frac{5}{16^2} + \frac{1}{16^3} \leq 16$$ for every integer $n \geq 16$.

Therefore

$$n^4 - 5n - 8 \geq 15n^3 + 11n^2 + 9$$ for every integer $n \geq 16$.

Example 11.2.6 – *Solution (4/4)* continued

Thus it is possible to find bands around each graph so that the band around the graph of $g(n)$ lies entirely above the band around the graph of $f(n)$ for every integer $n \geq 16$.

In fact, the second derivative test from calculus shows that $g(n)$ is growing at a faster rate than $f(n)$, which explains why the graph of $g(n)$ bends upward more steeply than the graph of $f(n)$. Readers who have studied calculus can check this result themselves.

Example 11.2.7 – *Showing That One Function Is Not Big-O of Some Other Function*

Let $g(n) = n^4 - 5n - 8$. By Example 11.2.4, $g(n)$ is $\Omega(n^4)$. Show that $g(n)$ is not $O(n^r)$ for any positive real number $r < 4$.

Example 11.2.7 – *Solution (1/3)*

Suppose by way of contradiction that there exists a positive real number r such that $r < 4$ and $g(n)$ is $O(n^r)$. Then there exist positive real numbers B and b such that

$$0 \leq g(n) \leq Bn^r \qquad \text{for every integer } n \geq b.$$

Since it is also the case that $g(n)$ is $\Omega(n^4)$, there exist positive real numbers A and a such that

$$An^4 \leq g(n) \qquad \text{for every integer } n \geq a.$$

Example 11.2.7 – *Solution (2/3)* continued

Let t be the larger of a and b. Then

$$An^4 \le g(n) \le Bn^r \qquad \text{for every integer } n \ge t.$$

Let $\Rightarrow$ stand for the words "which implies that." It follows from the inequalities above that for every integer $n \ge t$,

$$\begin{aligned}
& An^4 \le Bn^r && \text{by transitivity of order} \\
\Rightarrow\quad & \frac{n^4}{n^r} \le \frac{B}{A} && \text{by dividing both sides by } An^r \\
\Rightarrow\quad & n^{4-r} \le \frac{B}{A} && \text{by algebra} \\
\Rightarrow\quad & n \le \sqrt[4-r]{B/A} && \text{by taking the } (4-r)\text{th root of both sides. (*)}
\end{aligned}$$

Example 11.2.7 – *Solution (3/3)* continued

Since $r < 4$, then $4 - r > 0$, which implies that $\sqrt[4-r]{B/A}$ is a fixed positive real number. But n can be greater than any fixed number, which implies that condition (*) is contradictory.

In other words, the supposition results in a contradiction, and hence it is false. Therefore, for each positive real number r with $r < 4$, $g(n)$ is not $O(n^r)$.

Orders of Polynomial Functions (4/6)

Theorem 11.2.4 Showing That a Big-*O* Relationship Does Not Hold

If f is a real-valued function defined on a set of nonnegative integers and $f(n)$ is $\Omega(n^m)$, where m is a positive integer, then $f(n)$ is not $O(n^p)$ for any positive real number $p < m$.

Theorem 11.2.5 On Polynomial Orders

If m is any integer with $m \geq 0$ and $a_0, a_1, a_2, \ldots, a_m$ are real numbers with $a_m > 0$, then $a_m n^m + a_{m-1} n^{m-1} + \cdots + a_1 n + a_0$ is $\Theta(n^m)$.

Example 11.2.8 – *Calculating Polynomial Orders Using the Theorem on Polynomial Orders*

Use the theorem on polynomial orders to find orders for the functions given by the following formulas.

a. $f(n) = 7n^5 + 5n^3 - n + 4$ for each positive integer *n.*

b. $g(n) = \dfrac{(n-1)(n+1)}{4}$ for each positive integer *n.*

Example 11.2.8 – *Solution*

a. By direct application of the theorem on polynomial orders, $7n^5 + 5n^3 - n + 4$ is $\Theta(n^5)$.

b. $$g(n) = \frac{(n-1)(n+1)}{4}$$

$$= \frac{1}{4}(n^2 - 1)$$

$$= \frac{1}{4}n^2 - \frac{1}{4}$$ by algebra.

Thus $g(n)$ is $\Theta(n^2)$

Orders of Polynomial Functions (5/6)

Theorem 11.2.6 Reciprocal Relationship between Ω- and O-notations

Let f and g be real-valued functions defined on the same set of nonnegative integers, and suppose there is a positive real number r such that $f(n) \geq 0$ and $g(n) \geq 0$ for every integer $n \geq r$. Then:

a. If $f(n)$ is $\Omega(g(n))$, then $g(n)$ is $O(f(n))$.

b. If $g(n)$ is $O(f(n))$, then $f(n)$ is $\Omega(g(n))$.

Theorem 11.2.7 Reflexive, Symmetric, and Transitive Properties of Θ-notation

Let f, g, and h be real-valued functions defined on the same set of nonnegative integers, and suppose there is a positive real number r such that $f(n) \geq 0$, $g(n) \geq 0$ and $h(n) \geq 0$, for every integer $n \geq r$. Then:

a. $f(n)$ is $\Theta(f(n))$.

b. If $f(n)$ is $\Theta(g(n))$, then $g(n)$ is $\Theta(f(n))$.

c. If $f(n)$ is $\Theta(g(n))$ and $g(n)$ is $\Theta(h(n))$, then $f(n)$ is $\Theta(h(n))$.

Orders of Polynomial Functions (6/6)

Theorem 11.2.8 Effect of Constants on Order Notations

Let f and g be real-valued functions defined on the same set of nonnegative integers, and suppose there is a positive real number r such that $f(n) \geq 0$ and $g(n) \geq 0$ for every integer $n \geq r$.

Then for every positive real number c:

a. If $f(n)$ is $\Omega(g(n))$, then $cf(n)$ is $\Omega(g(n))$;

b. If $f(n)$ is $O(g(n))$, then $cf(n)$ is $O(g(n))$;

c. If $f(n)$ is $\Theta(g(n))$, then $cf(n)$ is $\Theta(g(n))$.

Theorem 11.2.9 Orders of Sums and Products of Functions

Let f_1, f_2, g_1, and g_2 be real-valued functions defined on the same set of nonnegative integers, and suppose there is a positive real number r such that $f_1(n) \geq 0$, $f_2(n) \geq 0$, $g_1(n) \geq 0$, and $g_2(n) \geq 0$ for every integer $n \geq r$. Then:

a. If $f_1(n)$ is $\Theta(g(n))$ and $f_2(n)$ is $\Theta(g(n))$, then $(f_1(n) + f_2(n))$ is $\Theta(g(n))$.

b. If $f_1(n)$ is $\Theta(g_1(n))$ and $f_2(n)$ is $\Theta(g_2(n))$, then $(f_1(n)f_2(n))$ is $\Theta(g_1(n)g_2(n))$.

c. If $f_1(n)$ is $\Theta(g_1(n))$ and $f_2(n)$ is $\Theta(g_2(n))$ and if there is a real number s so that $g_1(n) \leq g_2(n)$ for every integer $n \geq s$, then $(f_1(n) + f_2(n))$ is $\Theta(g_2(n))$.

CHAPTER 11

ANALYSIS OF ALGORITHM EFFICIENCY

Copyright © Cengage Learning. All rights reserved.

11.3 Application Analysis of Algorithm Efficiency:I

Copyright © Cengage Learning. All rights reserved.

Application Analysis of Algorithm Efficiency:I

Starting in the late 1940s a number of mathematicians and computer scientists contributed to developing formal techniques for analyzing computer algorithms.

Alan Turing may have been the first to suggest a concrete way for doing this.

The Sequential Search Algorithm

The Sequential Search Algorithm (1/2)

The object of a search algorithm is to hunt through an array of data in an attempt to find a particular item *x*. In a sequential search, *x* is compared to the first item in the array, then to the second, then to the third, and so on.

The search is stopped if a match is found at any stage. On the other hand, if the entire array is processed without finding a match, then *x* is not in the array.

The Sequential Search Algorithm (2/2)

An example of a sequential search is shown diagrammatically in Figure 11.3.1.

Sequential Search of $a[1], a[2], \ldots, a[7]$ for x where $x = a[5]$

Figure 11.3.1

Example 11.3.1 – *Best-, Worst, and Average-Case Orders for Sequential Search*

Find best, worst-, and average-case orders for the sequential search algorithm from among the set of power functions.

Example 11.3.1 – *Solution (1/2)*

Suppose the sequential search algorithm is applied to an input array $a[1], a[2], \ldots, a[n]$ to find an item x.

In the best case, the algorithm requires only one comparison between x and the items in $a[1], a[2], \ldots, a[n]$.

This occurs when x is the first item in the array.

Example 11.3.1 – *Solution (2/2)* continued

Thus in the best case, the sequential search algorithm is $\Theta(1)$. (Note that $\Theta(1) = \Theta(n^0)$.) In the worst case, however, the algorithm requires *n* comparisons. This occurs when *x* = *a*[*n*] or when *x* does not appear in the array at all. Thus in the worst case, the sequential search algorithm is $\Theta(n)$.

Finally, because *x* is as likely to be in the first half of the array as in the second half, the algorithm requires an average of $\frac{n}{2}$ comparisons. Since $\frac{n}{2}$ is $\Theta(n)$, the algorithm's average-case performance is also $\Theta(n)$.

Measuring the Efficiency of an Algorithm

Measuring the Efficiency of an Algorithm (1/7)

Two aspects of algorithm efficiency are important: the amount of time required to execute the algorithm and the amount of memory space needed when it is run. In this chapter we introduce basic techniques for calculating time efficiency.

Similar techniques exist for calculating space efficiency. Occasionally, one algorithm may make more efficient use of time but less efficient use of memory space than another, forcing a trade-off based on the resources available to the user.

Measuring the Efficiency of an Algorithm (2/7)

How can the time efficiency of an algorithm be calculated? The answer depends on several factors. One is the size of the set of data that is input to the algorithm; for example, it takes longer for a sort algorithm to process 1,000,000 items than 100 items. Consequently, the execution time of an algorithm is generally expressed as a function of its input size.

The analysis of an algorithm for time efficiency begins by trying to count the number of elementary operations that must be performed when the algorithm is executed with an input of size *n* (in the best case, worst case, or average case).

Measuring the Efficiency of an Algorithm (3/7)

For instance, to compare two algorithms for evaluating a polynomial, the crucial issue is the number of additions and multiplications that are needed, whereas to compare two algorithms for searching a list to find a particular element, the important distinction is the number of comparisons that are required.

As is common, we will classify the following as **elementary operations**: addition, subtraction, multiplication, division, and comparisons that are indicated explicitly in an if-then statement using one of the relational symbols $<$, $\leq$, $>$, $\geq$, $=$, or $\neq$.

Measuring the Efficiency of an Algorithm (4/7)

When algorithms are implemented in a particular programming language and run on a particular computer, some operations are executed faster than others, and, of course, there are differences in execution times from one machine to another.

In certain practical situations these factors are taken into account when deciding which algorithm or which machine to use to solve a particular problem.

Measuring the Efficiency of an Algorithm (5/7)

In other cases, however, the machine is fixed, and rough estimates are all that are needed to determine the clear superiority of one algorithm over another.

Since each elementary operation is executed in time no longer than the slowest, the time efficiency of an algorithm is approximately proportional to the number of elementary operations required to execute the algorithm.

Measuring the Efficiency of an Algorithm (6/7)

For example, to accomplish a job with an input size of 1,000,000, algorithm *B* requires almost 2 trillion more elementary operations than algorithm *A.*

One way to look at this is that algorithm *B* has to perform 100,000 operations for each single operation performed by algorithm *A*.

Measuring the Efficiency of an Algorithm (7/7)

Definition

Let A be an algorithm.

1. Suppose the number of elementary operations performed when A is executed for an input of size n depends on n alone and not on the nature of the input data; say it equals $f(n)$. If $f(n)$ is $\Theta(g(n))$, we say that **A is $\Theta(g(n))$** or **A is of order $g(n)$**.
2. Suppose the number of elementary operations performed when A is executed for an input of size n depends on the nature of the input data as well as on n.
   a. Let $b(n)$ be the *minimum* number of elementary operations required to execute A for all possible input sets of size n. If $b(n)$ is $\Theta(g(n))$, we say that **in the best case, A is $\Theta(g(n))$** or **A has a best-case order of $g(n)$**.
   b. Let $w(n)$ be the *maximum* number of elementary operations required to execute A for all possible input sets of size n. If $w(n)$ is $\Theta(g(n))$, we say that **in the worst case, A is $\Theta(g(n))$** or **A has a worst-case order of $g(n)$**.

Example 11.3.2 – *Computing an Order of an Algorithm Segment*

Assume *n* is a positive integer and consider the following algorithm segment:

$p := 0, x := 2$
for $i := 2$ **to** n
  $p := (p + i) \cdot x$
next i

a. Compute the actual number of elementary operations that are performed when this algorithm segment is executed.

b. Use the theorem on polynomial orders to find an order for this algorithm segment.

Example 11.3.2 – *Solution (1/2)*

a. This algorithm segment has just one loop, which goes from 2 to n, and so there are as many iterations of the loop as there are integers from 2 to n, namely $n - 2 + 1 = n - 1$.

During each iteration, one multiplication and one addition are performed. Thus twice as many elementary operations are performed as there are iterations of the loop, and hence $2(n - 1) = 2n - 2$ elementary operations are performed when the algorithm segment is executed.

Example 11.3.2 – *Solution (2/2)* continued

b. By the theorem on polynomial orders,

$$2n - 2 \quad \text{is} \quad \Theta(n),$$

and so this algorithm segment is $\Theta(n)$.

Example 11.3.4 – *When the Number of Iterations Depends on the Floor Function*

Assume *n* is a positive integer and consider the following algorithm segment:

$$\begin{aligned}&\textbf{for } i := \lfloor n/2 \rfloor \textbf{ to } n\\&\quad a := n - i\\&\textbf{next } i\end{aligned}$$

a. Compute the actual number of elementary operations that are performed when this algorithm segment is executed.

b. Use the theorem on polynomial orders to find an order for this algorithm segment.

Example 11.3.4 – *Solution (1/3)*

a. Each iteration of the loop requires one subtraction, and the loop is iterated as many times as there are integers from $\lfloor n/2 \rfloor$ to *n*, namely, $n - \left\lfloor \frac{n}{2} \right\rfloor + 1$ times.

If *n* is even, then $\left\lfloor \frac{n}{2} \right\rfloor = \frac{n}{2}$, and so the number of elementary operations performed is

$$n - \left\lfloor \frac{n}{2} \right\rfloor + 1 = n - \frac{n}{2} + 1 = \frac{n+2}{2}.$$

Example 11.3.4 – *Solution (2/3)* continued

If n is odd, then $\left\lfloor \frac{n}{2} \right\rfloor = \frac{n-1}{2}$, and so the number of elementary operations performed is

$$n - \left\lfloor \frac{n}{2} \right\rfloor + 1 = n - \frac{n-1}{2} + 1 = \frac{2n - (n-1) + 2}{2} = \frac{n+3}{2}.$$

Example 11.3.4 – *Solution (3/3)* continued

b. By the theorem on polynomial orders,

$$\frac{n+2}{2} \quad \text{is} \quad \Theta(n) \quad \text{and} \quad \frac{n+3}{2} \quad \text{is} \quad \Theta(n)$$

also. Hence, regardless of whether *n* is even or odd, this algorithm segment is $\Theta(n)$.

The Insertion Sort Algorithm

The Insertion Sort Algorithm (1/4)

Insertion sort is an algorithm for arranging the items in an array into ascending order. Initially, the second item is compared to the first. If the second item is less than the first, their values are interchanged, and as a result the first two array items are in ascending order.

The idea of the algorithm is gradually to lengthen the section of the array that is known to be in ascending order by inserting each subsequent array item into its correct position relative to the preceding ones.

The Insertion Sort Algorithm (2/4)

When the last item has been placed, the entire array is in ascending order.

Figure 11.3.2 illustrates the action of step *k* of insertion on an array *a*[1], *a*[2], *a*[3], … , *a*[*n*].

Step *k* of Insertion Sort

Figure 11.3.2

The Insertion Sort Algorithm (3/4)

Algorithm 11.3.1 Insertion Sort

[The aim of this algorithm is to take an array $a[1], a[2], a[3], \ldots, a[n]$*, where* $n \geq 1$*, and reorder it. The output array is also denoted* $a[1], a[2], a[3], \ldots, a[n]$*. It has the same values as the input array, but they are in ascending order. In the kth step,* $a[1], a[2], a[3], \ldots, a[k-1]$ *is in ascending order, and* $a[k]$ *is inserted into the correct position with respect to it.]*

Input: n *[a positive integer],* $a[1], a[2], a[3], \ldots, a[n]$ *[an array of data items capable of being ordered]*

The Insertion Sort Algorithm (4/4)

continued

Algorithm 11.3.1 Insertion Sort

Algorithm Body:

for $k := 2$ **to** n

[Compare $a[k]$ to previous items in the array $a[1], a[2], a[3], \ldots, a[k-1]$, starting from the largest and moving downward. Whenever $a[k]$ is less than a preceding array item, the indexes of $a[k]$ and the preceding item are switched. As soon as $a[k]$ is greater than or equal to an array item, the value of $a[k]$ is left unchanged.]

```
    x := a[k]
    j := k − 1
    while (j ≠ 0)
      if x < a[j] then
          a[j + 1] := a[j]
          a[ j] := x
          j := j − 1
          else j := 0
      end if
    end while
next k
```

Output: $a[1], a[2], a[3], \ldots, a[n]$ *[in ascending order]*

Example 11.3.5 – *A Trace Table for Insertion Sort*

Construct a trace table showing the action of insertion sort on the array

$$a[1] = 6,\ a[2] = 3,\ a[3] = 5,\ a[4] = 7,\ a[5] = 2.$$

Example 11.3.5 – *Solution (1/3)*

The first column shows the state of the variables before the first iteration of the **for-next** loop.

n	5											
***a*[1]**	6	3										2
***a*[2]**	3	6		5							2	3
***a*[3]**	5			6						2	5	
***a*[4]**	7								2	6		
***a*[5]**	2								7			
k	2		3			4		5				
x	3		5			7		2				
j	1	0	2	1	0	3	0	4	3	2	1	0

Example 11.3.5 – *Solution (2/3)* continued

When the **for-next** loop is first iterated, k is assigned the value 2; x the value of $a[2]$, which is 3; and j the value of $k - 1$, which is 1.

Because $j \neq 0$, the **while** loop is entered and the condition for the **if-then-else** statement is tested.

Because $a[1] > x$, then $a[2]$ is assigned the value of $a[1]$, which is 6, j is assigned the value of $j - 1$, which is 0, and $a[1]$ is assigned the value of x, which is 3.

Example 11.3.5 – *Solution (3/3)* continued

The condition governing the **while** loop is tested again, but since $j = 0$, it is not satisfied, and so the **while** loop is not entered.

Then the value of k is incremented by 1 (so that it equals 3), and the **for-next** loop is entered a second time. This process continues until the value of k has been incremented to 6.

Because 6 is greater than the top value in the **for-next** loop, execution of the algorithm ceases, and the array items are in ascending order.

Example 11.3.6 – *Finding a Worst-Case Order for Insertion Sort*

a. When insertion sort is applied to the array $a[1]$, $a[2]$, $a[3]$, …, $a[n]$, what is the maximum number of comparisons if those that control the **while** and **for-next** loops are included in the count?

b. Use the theorem on polynomial orders to find a worst-case order for insertion sort.

Example 11.3.6 – *Solution (1/3)*

a. In each iteration of the **while** loop, one explicit comparison is made to test whether $a[j] > x$.

During the time that $a[k]$ is put into position relative to $a[1], a[2], \ldots, a[k-1]$, the maximum number of attempted iterations of the **while** loop is k.

This happens when $a[k]$ is less than every $a[1], a[2], \ldots, a[k-1]$ and results in $k - 1$ comparisons.

Example 11.3.6 – *Solution (2/3)* continued

Then when the *k*th iteration is attempted, a comparison results in setting $j = 0$ and so the condition of the **while** loop is not satisfied. Thus the maximum number of comparisons for a given value of k is $k - 1$.

Now because k goes from 2 to n, the maximum total number of comparisons occurs when the items in the array are in reverse order, and it equals

$2 + 3 + \ldots + n = (1 + 2 + 3 + \ldots + n) - 1$ by adding and subtracting 1

Example 11.3.6 – *Solution (3/3)* continued

$$= \frac{n(n+1)}{2} - 1 \qquad \text{by Theorem 5.2.1}$$

$$= \frac{n(n+1)}{2} - 1$$

$$= \frac{1}{2}n^2 + \frac{1}{2}n - 1 \qquad \text{by algebra.}$$

b. By the theorem on polynomial orders, $\frac{1}{2}n^2 + \frac{1}{2}n - 1$ is $\Theta(n^2)$, and so the insertion sort algorithm has worst-case order $\Theta(n^2)$.

Example 11.3.7 – *Finding an Average-Case Order for Insertion Sort*

a. What is the average number of comparisons that are performed when insertion sort is applied to the array $a[1]$, $a[2]$, $a[3]$, … , $a[n]$?

b. Use the theorem on polynomial orders to find an average-case order for insertion sort.

Example 11.3.7 – *Solution (1/7)*

a. For any positive integer n, let E_n be the average, or expected, number of comparisons used to sort $a[1]$, $a[2]$, … , $a[n]$ with insertion sort. Then for each integer $k = 2, 3, \ldots, n$,

$$\begin{bmatrix}\text{the expected number of}\\ \text{comparisons used to}\\ \text{sort } a[1], a[2], \ldots, a[k]\end{bmatrix}$$

$$= \begin{bmatrix}\text{the expected number of}\\ \text{comparisons used to}\\ \text{sort } a[1], a[2], \ldots, a[k-1]\end{bmatrix} + \begin{bmatrix}\text{the expected number of comparisons}\\ \text{used to place } a[k] \text{ into position}\\ \text{relative to } a[1], a[2], \ldots, a[k-1]\end{bmatrix}.$$

Example 11.3.7 – *Solution (2/7)* continued

Thus

$$E_k = E_{k-1} + \begin{bmatrix} \text{the expected number of comparisons} \\ \text{used to place } a[k] \text{ into position} \\ \text{relative to } a[1], a[2], \ldots, a[k-1] \end{bmatrix}.$$

Also, $E_1 = 0$ because when there is just one item in the array, $n = 1$ and no iterations of the outer loop are performed.

Example 11.3.7 – *Solution (3/7)* continued

Now at the time the **while** loop is used to place $a[k]$ relative to $a[1]$, $a[2]$, … , $a[k - 1]$, on average the value of $a[k]$ is equally likely to belong in any one of the first k positions.

Thus the probability of its belonging in any particular position is $1/k$. If it actually belongs in position j, then $k - j + 1$ comparisons will be used to move it, because there will be $k - j + 1$ attempted iterations of the **while** loop and, as noted in Example 11.3.6, there is one comparison per attempted iteration.

Example 11.3.7 – *Solution (4/7)* continued

According to the definition of expected value given below, the expected number of comparisons used to place $a[k]$ relative to $a[1], a[2], \ldots, a[k-1]$ is therefore

$$\sum_{j=1}^{k} \frac{1}{k}(k-j+1) = \frac{1}{k}\sum_{j=1}^{k}(k-j+1) \qquad \text{by Theorem 5.1.1}$$

Definition

Suppose the possible outcomes of an experiment, or random process, are real numbers $a_1, a_2, a_3, \ldots, a_n$, which occur with probabilities $p_1, p_2, p_3, \ldots, p_n$. The **expected value** of the process is

$$\sum_{k=1}^{n} a_k p_k = a_1 p_1 + a_2 p_2 + a_3 p_3 + \cdots + a_n p_n.$$

Example 11.3.7 – *Solution (5/7)* continued

$$= \frac{1}{k}[k + (k-1) + \cdots + 3 + 2 + 1]$$ by writing the summation in expanded form

$$= \frac{1}{k}\left(\frac{k(k+1)}{2}\right)$$ by Theorem 5.2.1

$$= \frac{k+1}{2}$$ by algebra.

Example 11.3.7 – *Solution (6/7)* continued

Hence

$$E_k = E_{k-1} + \frac{k+1}{2} \quad \text{for every integer } k \geq 2, \text{ and}$$

$E_1 = 0.$

In exercise 27 at the end of the section you are asked to solve this recurrence relation to show that

$$E_n = \frac{n^2 + 3n - 4}{4} \quad \text{for each integer } n \geq 1.$$

Example 11.3.7 – *Solution (7/7)* continued

b. By the theorem on polynomial orders,

$$\frac{n^2 + 3n - 4}{4} = \frac{1}{4}n^2 + \frac{3}{4}n - 1 \text{ is } \Theta(n^2),$$

and so the average-case order of insertion sort is also $\Theta(n^2)$.

CHAPTER 11

ANALYSIS OF ALGORITHM EFFICIENCY

Copyright © Cengage Learning. All rights reserved.

11.4 Exponential and Logarithmic Functions: Graphs and Orders

Copyright © Cengage Learning. All rights reserved.

Exponential and Logarithmic Functions: Graphs and Orders (1/2)

Exponential and logarithmic functions are of great importance in mathematics in general and in computer science in particular.

Several important computer algorithms have execution times that involve logarithmic functions of the size of the input data (which means they are relatively efficient for large data sets), and some have execution times that are exponential functions of the size of the input data (which means they are extremely inefficient for large data sets).

Exponential and Logarithmic Functions: Graphs and Orders (2/2)

In addition, since exponential and logarithmic functions arise naturally in the descriptions of many growth and decay processes and in the computation of many kinds of probabilities, these functions are used in the analysis of computer operating systems, in queuing theory, and in the theory of information.

Graphs of Exponential Functions

Graphs of Exponential Functions (1/4)

The exponential function with base $b > 0$ is the function that sends each real number x to b^x. The graph of the exponential function with base 2 (together with a partial table of its values) is shown in Figure 11.4.1.

x	2^x
0	$2^0 = 1$
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
–1	$2^{-1} = 0.5$
–2	$2^{-2} = 0.25$
–3	$2^{-3} = 0.125$
0.5	$2^{0.5} \cong 1.414$
–0.5	$2^{-0.5} \cong 0.707$

The Exponential Function with Base 2

Figure 11.4.1

Graphs of Exponential Functions (2/4)

Note that the values of this function increase with extraordinary rapidity. If we tried to continue drawing the graph using the scale shown in Figure 11.4.1, we would have to plot the point $(10, 2^{10})$ more than 21 feet above the horizontal axis. And the point $(30, 2^{30})$ would be located more than 610,080 miles higher—well beyond the moon!

The graph of any exponential function with base $b > 1$ has a shape that is similar to the graph of the exponential function with base 2.

Graphs of Exponential Functions (3/4)

The graph of any exponential function with base $b > 1$ has a shape that is similar to the graph of the exponential function with base 2. If $0 < b < 1$, then $1/b > 0$ and the graph of the exponential function with base b is the reflection across the vertical axis of the exponential function with base $1/b$.

Graphs of Exponential Functions (4/4)

These facts are illustrated in Figure 11.4.2.

Graphs of Exponential Functions

Figure 11.4.2

Graphs of Logarithmic Functions

Graphs of Logarithmic Functions (1/4)

Logarithms were first introduced by the Scotsman John Napier. Astronomers and navigators found them so useful for reducing the time needed to do multiplication and division that they quickly gained wide acceptance and played a crucial role in the remarkable development of those areas in the seventeenth century.

Nowadays, however, electronic calculators and computers are available to handle most computations quickly and conveniently, and logarithms and logarithmic functions are used primarily as conceptual tools.

Graphs of Logarithmic Functions (2/4)

Definition

If b is a positive real number not equal to 1, then the **logarithmic function with base b, $\log_b$: $\mathbf{R}^+ \to \mathbf{R}$**, is the function that sends each positive real number x to the number $\log_b x$, which is the exponent to which b must be raised to obtain x.

The logarithmic function with base *b* is, in fact, the inverse of the exponential function with base *b.*

It follows that the graphs of the two functions are symmetric with respect to the line $y = x$.

Graphs of Logarithmic Functions (3/4)

The graph of the logarithmic function with base $b > 1$ is shown in Figure 11.4.3.

The Graph of the Logarithmic Function with Base $b > 1$

Figure 11.4.3

Graphs of Logarithmic Functions (4/4)

If its base *b* is greater than 1, the logarithmic function is increasing. Analytically, this means that

if $b > 1$, then for all positive numbers x_1 and x_2,

$$\text{if } x_1 < x_2, \text{ then } \log_b(x_1) < \log_b(x_2). \qquad 11.4.1$$

If k is an integer and x is a real number with

$$2^k \leq x < 2^{k+1}, \text{ then } \lfloor \log_2 x \rfloor = k. \qquad 11.4.2$$

$$\text{For any odd integer } n > 1, \lfloor \log_2(n-1) \rfloor = \lfloor \log_2 n \rfloor. \qquad 11.4.3$$

Application: Number of Bits Needed to Represent an Integer in Binary Notation

Application: Number of Bits Needed to Represent an Integer in Binary Notation (1/3)

Any positive integer n can be written in a unique way as

$$n = 2^k + c_{k-1} \cdot 2^{k-1} + \cdots + c_2 \cdot 2^2 + c_1 \cdot 2 + c_0,$$

where k is a nonnegative integer and each $c_0, c_1, c_2, \ldots c_{k-1}$ is either 0 or 1. Then the binary representation of n is

$$1c_{k-1}c_{k-2} \ldots c_2c_1c_0,$$

and so the number of binary digits needed to represent n is $k + 1$. What is $k + 1$ as a function of n? Observe that since each $c_i \leq 1$,

$$n = 2^k + c_{k-1} \cdot 2^{k-1} + \cdots + c_2 \cdot 2^2 + c_1 \cdot 2 + c_0 \leq 2^k + 2^{k-1} + \cdots + 2^2 + 2 + 1.$$

Application: Number of Bits Needed to Represent an Integer in Binary Notation (2/3)

Now, by the formula for the sum of a geometric sequence

$$2^k + 2^{k-1} + \cdots + 2^2 + 2 + 1 = \frac{2^{k+1} - 1}{2 - 1} = 2^{k+1} - 1.$$

Hence, by transitivity of order,

$$n \leq 2^{k+1} - 1 < 2^{k+1}. \qquad 11.4.6$$

In addition, because each $c_i \geq 0$,

$$2^k \leq 2^k + c_{k-1} \cdot 2^{k-1} + \cdots + c_2 \cdot 2^2 + c_1 \cdot 2 + c_0 = n. \qquad 11.4.7$$

Application: Number of Bits Needed to Represent an Integer in Binary Notation (3/3)

Putting inequalities (11.4.6) and (11.4.7) together gives the double inequality

$$2^k \leq n < 2^{k+1}.$$

Then, by property (11.4.2),

$$k = \lfloor \log_2 n \rfloor,$$

and so the number of binary digits needed to represent n is $\lfloor \log_2 n \rfloor + 1$.

Example 11.4.3 – *Number of Bits in a Binary Representation*

How many binary digits are needed to represent 52,837 in binary notation?

Example 11.4.3 – *Solution*

If you compute the logarithm with base 2 using the formula $\log_b (xy) = \log_b x + \log_b y$ and a calculator or computer that gives you approximate values of logarithms with base 10, you find that

$$\log_2(52{,}837) \cong \frac{\log_{10}(52{,}837)}{\log_{10}(2)} \cong \frac{4.722938151}{0.3010299957} \cong 15.7.$$

Thus the binary representation of 52,837 has $\lfloor 15.7 \rfloor + 1 = 15 + 1 = 16$ binary digits.

Application: Using Logarithms to Solve Recurrence Relations

Application: Using Logarithms to Solve Recurrence Relations (1/2)

A class of recurrence relations that is very important in computer science has solutions that can be expressed in terms of logarithms. One such recurrence relation is discussed in the next example.

Example 11.4.4 – *A Recurrence Relation with a Logarithmic Solution*

Define a sequence $a_1, a_2, a_3, \ldots$ recursively as follows:

$$a_1 = 1$$

$$a_k = 2a_{\lfloor k/2 \rfloor} \qquad \text{for each integer } k \geq 2.$$

a. Use iteration to guess an explicit formula for this sequence.

b. Use strong mathematical induction to confirm the correctness of the formula obtained in part (a).

Example 11.4.4 – *Solution (1/4)*

a. Begin by iterating to find the values of the first few terms of the sequence.

$$
\begin{aligned}
a_1 &= 1 && 1 = 2^0 \\
a_2 &= 2a_{\lfloor 2/2 \rfloor} = 2a_1 = 2 \cdot 1 = 2 \\
a_3 &= 2a_{\lfloor 3/2 \rfloor} = 2a_1 = 2 \cdot 1 = 2 && 2 = 2^1 \\
a_4 &= 2a_{\lfloor 4/2 \rfloor} = 2a_2 = 2 \cdot 2 = 4 \\
a_5 &= 2a_{\lfloor 5/2 \rfloor} = 2a_2 = 2 \cdot 2 = 4 \\
a_6 &= 2a_{\lfloor 6/2 \rfloor} = 2a_3 = 2 \cdot 2 = 4 && 4 = 2^2 \\
a_7 &= 2a_{\lfloor 7/2 \rfloor} = 2a_3 = 2 \cdot 2 = 4
\end{aligned}
$$

Example 11.4.4 – *Solution (2/4)* continued

$$\left.\begin{aligned} a_8 &= 2a_{\lfloor 8/2 \rfloor} = 2a_4 = 2\cdot 4 = 8 \\ a_9 &= 2a_{\lfloor 9/2 \rfloor} = 2a_4 = 2\cdot 4 = 8 \\ &\vdots \\ a_{15} &= 2a_{\lfloor 15/2 \rfloor} = 2a_7 = 2\cdot 4 = 8 \end{aligned}\right\} \quad 8 = 2^3$$

$$\left.\begin{aligned} a_{16} &= 2a_{\lfloor 16/2 \rfloor} = 2a_8 = 2\cdot 8 = 16 \\ &\vdots \end{aligned}\right\} \quad 16 = 2^4$$

Note that in each case when the subscript *n* is between two powers of 2, a_n equals the smaller power of 2.

Example 11.4.4 – *Solution (3/4)* continued

More precisely:

$$\text{If } 2^i \le n < 2^{i+1}, \text{ then } a_n = 2^i. \qquad 11.4.8$$

Now since *n* satisfies the inequality

$$2^i \le n < 2^{i+1},$$

then (by property 11.4.2)

$$i = \lfloor \log_2 n \rfloor.$$

Substituting into statement (11.4.8) suggests that

$$a_n = 2^{\lfloor \log_2 n \rfloor}.$$

Example 11.4.4 – *Solution (4/4)* continued

b. The following proof shows that the recursively defined sequence defined in this example does in fact satisfy the formula obtained in part (a).

In other words, if $a_1, a_2, a_3, \ldots$ is a sequence of numbers that satisfies

$$a_1 = 1, \text{ and } a_k = 2a_{\lfloor k/2 \rfloor} \qquad \text{for each integer } k \geq 2,$$

then the sequence satisfies the formula

$$a_n = 2^{\lfloor \log_2 n \rfloor} \qquad \text{for every integer } n \geq 1.$$

Exponential and Logarithmic Orders

Exponential and Logarithmic Orders (1/4)

The graph of the logarithmic function with any base $b > 1$ lies *below* the graph of every power function with a positive exponent, and the graph of the exponential function with any base $b > 1$ lies *above* the graph of each of these power functions.

In analytic terms, this says the following:

For all real numbers b and r with $b > 1$ and $r > 0$, there is a positive real number s such that

$$\log_b x \leq x^r \quad \text{for every real number } x \geq s. \qquad 11.4.9$$

and

$$x^r \leq b^x \quad \text{for every real number } x \geq s. \qquad 11.4.10$$

Exponential and Logarithmic Orders (2/4)

These statements have the following implications for *O*-notation.

For all real numbers b and r with $b > 1$ and $r > 0$,

$$\log_b n \quad \text{is} \quad O(n^r) \qquad 11.4.11$$

and

$$n^r \quad \text{is} \quad O(b^n) \qquad 11.4.12$$

Another important function in the analysis of algorithms is the function *f* defined by the formula

$$f(x) = x \log_b x \qquad \text{for every real number } x > 0.$$

Exponential and Logarithmic Orders (3/4)

For large values of *x*, the graph of this function fits in between the graph of the identity function and the graph of the squaring function. More precisely:

For every real number b with $b > 1$, there is a positive real number s such that for every real number $x \geq s$,

$$x \leq x \log_b x \leq x^2. \qquad 11.4.13$$

The *O*-notation versions of these facts are as follows:

For every real number $b > 1$,

$$n \quad \text{is} \quad O(n \log_b n) \quad \text{and} \quad n \log_b n \quad \text{is} \quad O(n^2). \qquad 11.4.14$$

Exponential and Logarithmic Orders (4/4)

Figure 11.4.4 illustrates the relationships among some power functions, the logarithmic function with base 2, the exponential function with base 2, and the function defined by the formula $x \to x \log_2 x$.

Note that different scales are used on the horizontal and vertical axes.

Graphs of Some Logarithmic, Exponential, and Power Functions

Figure 11.4.4

Example 11.4.5 – *Orders That Involve Logarithmic Functions*

a. Show that if b and c are any real numbers with $b > 1$ and $c > 1$, then $\log_b n$ and $\log_c n$ have the same order.

b. Show that $n + n \log_2 n$ is $\Theta(n \log_2 n)$.

Example 11.4.5 – *Solution (1/3)*

a. By the formula $\log_c x = \dfrac{\log_b x}{\log_b c}$

$$\log_b n = \frac{\log_c n}{\log_c b} = \left(\frac{1}{\log_c b}\right)\log_c n.$$

Since both $b > 1$ and $c > 1$, then $\log_c b > 0$, and so $\dfrac{1}{\log_c b} > 0.$

Thus, by Theorem 11.2.8(c), if $f(n)$ is $\Theta(g(n))$, then $cf(n)$ is $\Theta(g(n))$.

$\left(\dfrac{1}{\log_c b}\right)\log_c n$ is $\Theta(\log_c n)$, and so $\log_b n$ is $\Theta(\log_c n)$.

Example 11.4.5 – *Solution (2/3)* continued

Exactly the same reasoning with b and c interchanged gives that $\log_c n$ is $\Theta(\log_b n)$.

Thus $\log_b n$ and $\log_c n$ have the same order.

b. By Theorem 11.2.7, n is $\Theta(n)$ and $n \log_2 n$ is $\Theta(n \log_2 n)$.

Theorem 11.2.7 Reflexive, Symmetric, and Transitive Properties of Θ-notation

Let f, g, and h be real-valued functions defined on the same set of nonnegative integers, and suppose there is a positive real number r such that $f(n) \geq 0$, $g(n) \geq 0$ and $h(n) \geq 0$, for every integer $n \geq r$. Then:

a. $f(n)$ is $\Theta(f(n))$.

b. If $f(n)$ is $\Theta(g(n))$, then $g(n)$ is $\Theta(f(n))$.

c. If $f(n)$ is $\Theta(g(n))$ and $g(n)$ is $\Theta(h(n))$, then $f(n)$ is $\Theta(h(n))$.

Example 11.4.5 – *Solution (3/3)* continued

In addition, by property 11.4.13, there is a positive real number r such that for every integer $n \geq r$, $n \leq n \log_2 n$.

Thus it follows from Theorem 11.2.9(c) (If $f_1(n)$ is $\Theta(g_1(n))$ and $f_2(n)$ is $\Theta(g_2(n))$ and if there is a real number s so that $g_1(n) \leq g_2(n)$ for every integer $n > s$, then $(f_1(n) + f_2(n))$ is $\Theta(g_2(n))$.) that $\Theta(n \log_2 n)$. $n + n \log_2 n$ is $\Theta(n \log_2 n)$.

Example 11.4.6 – *An Exponential Order*

Show that $n!$ is $O(n^n)$.

Example 11.4.6 – *Solution*

For any positive integer *n*,

$$n! = \underbrace{n \cdot (n-1) \cdot (n-2) \cdot 2 \cdot 1}_{n \text{ factors}}$$

$$\leq \underbrace{n \cdot n \cdot n \cdot n \cdot n}_{n \text{ factors}} = n^n$$

because $(n - 1) \leq n$, $(n - 2) \leq n$, … , $2 \leq n$, and $1 \leq n$. Let $B = 1$ and $b = 1$. Then, since $n! \geq 0$ when $n \geq b$,

$$0 \leq n! \leq n^n \quad \text{for every integer } n \geq b.$$

Thus $n!$ is $O(n^n)$ by definition of *O*-notation.

Example 11.4.7 – *Order of a Harmonic Sum (1/3)*

Sums of the form $1 + \frac{1}{2} + \cdots + \frac{1}{n}$ are called *harmonic sums*. They occur in the analysis of various computer algorithms such as quick sort. $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ is $\Omega(\ln n)$ by performing the steps shown on the next slide.

Example 11.4.7 – *Order of a Harmonic Sum (2/3)*

a. Interpret Figure 11.4.6 to show that

$$\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n} \leq \ln n.$$

and

$$\ln n \leq 1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}.$$

Figure 11.4.6

Example 11.4.7 – *Order of a Harmonic Sum (3/3)*

b. Show that if *n* is an integer that is at least 3, then $1 \leq \ln n$.

c. Deduce from (a) and (b) that if the integer *n* is greater than or equal to 3, then

$$\ln n \leq 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \leq 2 \ln n.$$

d. Deduce from (c) that

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \text{ is } \Theta(\ln n).$$

Example 11.4.7 – *Solution (1/6)*

a. Figure 11.4.6(a) shows rectangles whose bases are the intervals between each pair of integers from 1 to n and whose heights are the heights of the graph of $y = 1/x$ above the right-hand endpoints of the intervals.

Figure 11.4.6(b) shows rectangles with the same bases but whose heights are the heights of the graph above the left-hand endpoints of the intervals.

Example 11.4.7 – *Solution (2/6)* continued

Now the area of each rectangle is its base times its height. Since all the rectangles have base 1, the area of each rectangle equals its height.

Thus in Figure 11.4.6(a),

$$\text{the area of the rectangle from 1 to 2 is } \frac{1}{2};$$

$$\text{the area of the rectangle from 2 to 3 is } \frac{1}{3};$$

$$\vdots$$

$$\text{the area of the rectangle from } n-1 \text{ to } n \text{ is } \frac{1}{n}.$$

Example 11.4.7 – *Solution (3/6)* continued

So the sum of the areas of all the rectangles is

$$\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}.$$

From the picture it is clear that this sum is less than the area underneath the graph of *f* between *x* = 1 and *x* = *n*, which is known to equal ln *n*. Hence

$$\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n} \leq \ln n.$$

Example 11.4.7 – *Solution (4/6)* continued

A similar analysis of the areas of the combined blue and gray rectangles in Figure 11.4.6(b) shows that

$$\ln n \leq 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

b. Suppose n is an integer and $n \geq 3$. Since $e \cong 2.718$, then $n \geq e$. Now the logarithmic function with base e is strictly increasing.

Thus since $e \leq n$, then $1 = \ln e \leq \ln n$.

Example 11.4.7 – *Solution (5/6)* continued

c. By part (a),

$$\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n} \leq \ln n,$$

and by part (b),

$$1 \leq \ln n.$$

Adding these two inequalities together gives

$$1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n} \leq 2\ln n \quad \text{for each integer } n \geq 3.$$

Example 11.4.7 – *Solution (6/6)* continued

d. Combining the results of parts (a) and (c) leads to the conclusion that for every integer $n \geq 3$,

$$\ln n \leq 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \leq 2 \ln n.$$

Let $A = 1$, $B = 2$, and $k = 3$. Then $\ln n > 0$ for $n \geq 3$ and

$$A \ln n \leq 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \leq B \ln n \text{ for every integer } n > k.$$

Hence by definition of Θ-notation,

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \text{ is } \Theta(\ln n).$$

CHAPTER 11

ANALYSIS OF ALGORITHM EFFICIENCY

Copyright © Cengage Learning. All rights reserved.

11.5 Application: Analysis of Algorithm Efficiency II

Copyright © Cengage Learning. All rights reserved.

Application: Analysis of Algorithm EfficiencyII(1/5)

Have you ever played the “guess my number” game? A person thinks of a number between two other numbers—say, 1 and 10 or 1 and 100 for example—and you try to figure out what it is, using the least possible number of guesses. Each time you guess a number, the person tells you whether you are correct, too low, or too high.

If you have played this game, you have probably already hit upon the most efficient strategy: Begin by guessing a number as close to the middle of the two given numbers as possible.

Application: Analysis of Algorithm EfficiencyII(2/5)

If your guess is too high, then the number is between the lower of the two given numbers and the one you first chose. If your guess is too low, then the number is between the number you first chose and the higher of the two given numbers.

In either case, you take as your next guess a number as close as possible to the middle of the new range in which you now know the number lies. You repeat this process as many times as necessary until you have found the person’s number.

Application: Analysis of Algorithm EfficiencyII(3/5)

The technique described previously is an example of a general strategy called **divide and conquer**, which works as follows: To solve a problem, reduce it to a fixed number of smaller problems of the same kind, which can themselves be reduced to the same fixed number of smaller problems of the same kind, and so forth until easily resolved problems are obtained.

In this case, the problem of finding a particular number in a given range of numbers is reduced at each stage to finding a particular number in a range of numbers approximately half as long.

Application: Analysis of Algorithm EfficiencyII(4/5)

It turns out that algorithms using a divide-and-conquer strategy are generally quite efficient and nearly always have orders involving logarithmic functions.

In this section we define the *binary search* algorithm, which is the formalization of the “guess my number” game described previously, and we compare the efficiency of binary search to the sequential search. Then we develop a divide-and-conquer algorithm for sorting, *merge sort*, and compare its efficiency with that of insertion sort and selection sort.

Application: Analysis of Algorithm EfficiencyII(5/5)

Almost a hundred years before the first computer was built and working only from a description of how such a machine might function, Ada Augusta, Countess of Lovelace, foresaw both the general nature of the problems it could address and the importance of designing an efficient arrangement of computations to solve them.

Binary Search

Binary Search (1/5)

Whereas a sequential search can be performed on an array whose elements are in any order, a binary search can be performed only on an array whose elements are arranged in ascending (or descending) order. Given an array $a[1], a[2], \ldots, a[n]$ of distinct elements arranged in ascending order, consider the problem of trying to find a particular element x in the array.

To use binary search, first compare x to the “middle element” of the array. If the two are equal, the search is successful.

Binary Search (2/5)

If the two are not equal, then because the array elements are in ascending order, comparing the values of *x* and the middle array element narrows the search either to the lower subarray (consisting of all the array elements below the middle element) or to the upper subarray (consisting of all array elements above the middle element).

The search continues by repeating this basic process over and over on successively smaller subarrays. It terminates either when a match occurs or when the subarray to which the search has been narrowed contains no elements.

Binary Search (3/5)

The efficiency of the algorithm is a result of the fact that at each step, the length of the subarray to be searched is roughly half the length of the array of the previous step. This process is illustrated in Figure 11.5.1.

One Iteration of the Binary Search Process

Figure 11.5.1

Binary Search (4/5)

Algorithm 11.5.1 Binary Search

[The aim of this algorithm is to search for an element x in an ascending array of elements $a[1], a[2], \ldots, a[n]$*. If x is found, the variable index is set equal to the index of the array element where x is located. If x is not found, index is not changed from its initial value, which is 0. The variables bot and top denote the bottom and top indexes of the array currently being examined.]*

Input: *n [a positive integer],* $a[1], a[2], \ldots, a[n]$ *[an array of data items given in ascending order], x [a data item of the same data type as the elements of the array]*

Binary Search (5/5)

continued

Algorithm 11.5.1 Binary Search

Algorithm Body:

$index := 0$, $bot := 1$, $top := n$

[Compute the middle index of the array, mid. Compare x to a[mid]. If the two are equal, the search is successful. If not, repeat the process either for the lower or for the upper subarray, either giving top the new value mid −1 or giving bot the new value mid +1. Each iteration of the loop either decreases the value of top or increases the value of bot. Thus, if the looping is not stopped by success in the search process, the value of top eventually becomes less than the value of bot, which stops the looping process and shows that x is not an element of the array.]

while ($top \geq bot$ and $index = 0$)

$$mid := \left\lfloor \frac{bot + top}{2} \right\rfloor$$

if $a[mid] = x$ **then** $index := mid$

if $a[mid] > x$

then $top := mid - 1$

else $bot := mid + 1$

end while

[If index has the value 0 at this point, then the **while** *loop was not entered because top < bot, and so x is not in the array. Otherwise, index gives the index of the array where x is located.]*

Output: *index [a nonnegative integer]*

Example 11.5.1 – *Tracing the Binary Search Algorithm*

Trace the action of Algorithm 11.5.1 on the variables *index*, *bot*, *top*, *mid*, and the values of *x* given in (a) and (b) below for the input array

a[1] = Ann, *a*[2] = Dawn, *a*[3] = Erik, *a*[4] = Gail, *a*[5] = Juan, *a*[6] = Matt, *a*[7] = Max, *a*[8] = Rita, *a*[9] = Tsuji, *a*[10] = Yuen,

where alphabetical ordering is used to compare elements of the array.

a. *x* = Erik b. *x* = Sara

Example 11.5.1 – *Solution*

a.

index	0			3
bot	1		3	
top	10	4		
mid		5	2	3
x	Erik			

b.

index	0			
bot	1	6	9	
top	10			8
mid		5	8	9
x	Sara			

The Efficiency of the Binary Search Algorithm

The Efficiency of the Binary Search Algorithm (1/6)

Here briefly is the idea for how to derive the efficiency of the binary search algorithm. At each stage of the binary search process, the length of the new subarray to be searched is approximately half that of the previous one, and in the worst case, every subarray down to a subarray with a single element must be searched.

Consequently, in the worst case, the maximum number of iterations of the algorithm's **while** loop is 1 more than the number of times the original input array can be cut approximately in half.

The Efficiency of the Binary Search Algorithm (2/6)

Now if the length *n* of this array is a power of 2 ($n = 2^k$ for some integer *k*), then *n* can be halved exactly $k = \log_2 n = \lfloor \log_2 n \rfloor$ times before an array of length 1 is reached. If *n* is not a power of 2, then $n = 2^k + m$ for some integer *k* (where $m < 2^k$), and thus *n* can be split approximately in half *k* times also, and so in this case, $k = \lfloor \log_2 n \rfloor$ also.

Thus in the worst case, the maximum number of iterations of the **while** loop in the binary search algorithm (which is proportional to the number of comparisons required to execute it) is $\lfloor \log_2 n \rfloor + 1$.

The Efficiency of the Binary Search Algorithm (3/6)

The derivation is concluded by noting that $\lfloor \log_2 n \rfloor + 1$ is $O(\log_2 n)$. The details of the derivation are developed in Examples 11.5.2-11.5.6. Throughout the derivation, for each integer $n \geq 1$, let

> w_n = the number of iterations of the **while** loop in a *worst-case* execution of the binary search algorithm for an input array of length n.

The first issue to consider is this. If the length of the input array for one iteration of the **while** loop is known, what is the greatest possible length of the array input to the next iteration?

Example 11.5.2 – *The Length of the Input Array to the Next Iteration of the Loop*

Prove that if an array of length k is input to the **while** loop of the binary search algorithm, then after one unsuccessful iteration of the loop, the input to the next iteration is an array of length at most $\lfloor k/2 \rfloor$.

Example 11.5.2 – *Solution (1/4)*

Consider what occurs when an array of length *k* is input to the **while** loop in the case where $x \neq a[mid]$:

$$\underbrace{a[bot],\ a[bot+1],\ \ldots,\ a[mid-1],}_{\text{new input to the while loop if } x < a[mid]}\quad \underset{\text{“middle element”}}{a[mid]},\quad \underbrace{a[mid+1],\ \ldots,\ a[top-1],\ a[top].}_{\text{new input to the while loop if } x > a[mid]}$$

Since the input array has length *k*, the value of *mid* depends on whether *k* is odd or even.

Example 11.5.2 – *Solution (2/4)* continued

In both cases we match up the array elements with the integers from 1 to k and analyze the lengths of the left and right subarrays.

In case k is odd, both the left and the right subarrays have length $\lfloor k/2 \rfloor$. In case k is even, the left subarray has length $\lfloor k/2 \rfloor - 1$ and the right subarray has length $\lfloor k/2 \rfloor$.

Example 11.5.2 – *Solution (3/4)* continued

The reasoning behind these results is shown in Figure 11.5.3.

$$\textbf{k odd:}\quad \underbrace{\begin{array}{ccc} a[bot] & \cdots & a[mid-1] \\ \updownarrow & & \updownarrow \\ 1 & \cdots & \frac{k+1}{2}-1 \end{array}}_{\text{length} = \frac{k+1}{2}-1 = \frac{k-1}{2} = \left\lfloor \frac{k}{2} \right\rfloor} \quad \begin{array}{c} a[mid] \\ \updownarrow \\ \frac{k+1}{2} \end{array} \quad \underbrace{\begin{array}{ccc} a[mid+1] & \cdots & a[top] \\ \updownarrow & & \updownarrow \\ \frac{k+1}{2}+1 & \cdots & k \end{array}}_{\text{length} = k-\left(\frac{k+1}{2}+1\right)+1 = \frac{k-1}{2} = \left\lfloor \frac{k}{2} \right\rfloor}$$

$$\textbf{k even:}\quad \underbrace{\begin{array}{ccc} a[bot] & \cdots & a[mid-1] \\ \updownarrow & & \updownarrow \\ 1 & \cdots & \frac{k}{2}-1 \end{array}}_{\text{length} = \frac{k}{2}-1 = \left\lfloor \frac{k}{2} \right\rfloor - 1} \quad \begin{array}{c} a[mid] \\ \updownarrow \\ \frac{k}{2} \end{array} \quad \underbrace{\begin{array}{ccc} a[mid+1] & \cdots & a[top] \\ \updownarrow & & \updownarrow \\ \frac{k}{2}+1 & \cdots & k \end{array}}_{\text{length} = k-\left(\frac{k}{2}+1\right)+1 = \frac{k}{2} = \left\lfloor \frac{k}{2} \right\rfloor}$$

Lengths of the Left and Right Subarrays

Figure 11.5.3

Example 11.5.2 – *Solution (4/4)* continued

Because the maximum of the numbers $\lfloor k/2 \rfloor$ and $\lfloor k/2 \rfloor - 1$ is $\lfloor k/2 \rfloor$, in the worst case this will be the length of the array input to the next iteration of the loop.

The Efficiency of the Binary Search Algorithm (4/6)

To find the order of the algorithm, an explicit formula for w_1, w_2, w_3, … is needed. The next example derives a recurrence relation for the sequence.

Example 11.5.3 – *A Recurrence Relation for $w_1, w_2, w_3, \ldots$ (1/2)*

Prove that the sequence $w_1, w_2, w_3, \ldots$ satisfies the recurrence relation and initial condition

$$w_1 = 1,$$

$$w_k = 1 + w_{\lfloor k/2 \rfloor} \quad \text{for every integer } k > 1.$$

Example 11.5.3 – *A Recurrence Relation for w_1, w_2, w_3, . . . (2/2)*

***k* odd:**

$$\underbrace{\begin{array}{ccc} a[bot] & \cdots & a[mid-1] \\ \updownarrow & & \updownarrow \\ 1 & \cdots & \frac{k+1}{2}-1 \end{array}}_{\text{length} = \frac{k+1}{2} - 1 = \frac{k-1}{2} = \left\lfloor \frac{k}{2} \right\rfloor} \quad \begin{array}{c} a[mid] \\ \updownarrow \\ \frac{k+1}{2} \end{array} \quad \underbrace{\begin{array}{ccc} a[mid+1] & \cdots & a[top] \\ \updownarrow & & \updownarrow \\ \frac{k+1}{2}+1 & \cdots & k \end{array}}_{\text{length} = k - \left(\frac{k+1}{2}+1\right) + 1 = \frac{k-1}{2} = \left\lfloor \frac{k}{2} \right\rfloor}$$

***k* even:**

$$\underbrace{\begin{array}{ccc} a[bot] & \cdots & a[mid-1] \\ \updownarrow & & \updownarrow \\ 1 & \cdots & \frac{k}{2}-1 \end{array}}_{\text{length} = \frac{k}{2} - 1 = \left\lfloor \frac{k}{2} \right\rfloor - 1} \quad \begin{array}{c} a[mid] \\ \updownarrow \\ \frac{k}{2} \end{array} \quad \underbrace{\begin{array}{ccc} a[mid+1] & \cdots & a[top] \\ \updownarrow & & \updownarrow \\ \frac{k}{2}+1 & \cdots & k \end{array}}_{\text{length} = k - \left(\frac{k}{2}+1\right) + 1 = \frac{k}{2} = \left\lfloor \frac{k}{2} \right\rfloor}$$

Lengths of the Left and Right Subarrays

Figure 11.5.3

Example 11.5.3 – *Solution (1/2)*

In Example 11.5.2 it was shown that given an input array of length k to the **while** loop, the worst that can happen is that the next iteration of the loop will have to search an array of length $\lfloor k/2 \rfloor$.

Hence the maximum number of iterations of the loop is 1 more than the maximum number necessary to execute it for an input array of length $\lfloor k/2 \rfloor$.

Example 11.5.3 – *Solution (2/2)* continued

In symbols,

$$w_k = 1 + w_{\lfloor k/2 \rfloor}.$$

Also

$$w_1 = 1$$

because for an input array of length 1 (*bot* = *top*), the **while** loop iterates only one time.

The Efficiency of the Binary Search Algorithm (5/6)

Now that a recurrence relation for w_1, w_2, w_3, … has been found, iteration can be used to come up with a good guess for an explicit formula.

Example 11.5.4 – *An Explicit Formula for w_1, w_2, w_3, . . .*

Apply iteration to the recurrence relation found in Example 11.5.3 to conjecture an explicit formula for w_1, w_2, w_3, …

Example 11.5.4 – *Solution (1/2)*

Begin by iterating to find the values of the first few terms of the sequence.

$$
\begin{aligned}
w_1 &= 1 && 1 = 2^0 : 1 = 0 + 1 \\
w_2 &= 1 + w_{\lfloor 2/2 \rfloor} = 1 + w_1 = 1 + 1 = 2 \\
w_3 &= 1 + w_{\lfloor 3/2 \rfloor} = 1 + w_1 = 1 + 1 = 2 && 2 = 2^1 : 2 = 1 + 1 \\
w_4 &= 1 + w_{\lfloor 4/2 \rfloor} = 1 + w_2 = 1 + 2 = 3 \\
w_5 &= 1 + w_{\lfloor 5/2 \rfloor} = 1 + w_2 = 1 + 2 = 3 \\
w_6 &= 1 + w_{\lfloor 6/2 \rfloor} = 1 + w_3 = 1 + 2 = 3 \\
w_7 &= 1 + w_{\lfloor 7/2 \rfloor} = 1 + w_3 = 1 + 2 = 3 && 4 = 2^2 : 3 = 2 + 1 \\
w_8 &= 1 + w_{\lfloor 8/2 \rfloor} = 1 + w_4 = 1 + 3 = 4 \\
w_9 &= 1 + w_{\lfloor 9/2 \rfloor} = 1 + w_4 = 1 + 3 = 4 \\
&\vdots \\
w_{15} &= 1 + w_{\lfloor 15/2 \rfloor} = 1 + w_7 = 1 + 3 = 4 && 8 = 2^3 : 4 = 3 + 1 \\
w_{16} &= 1 + w_{\lfloor 16/2 \rfloor} = 1 + w_8 = 1 + 4 = 5 \\
&\vdots && 16 = 2^4 : 5 = 4 + 1
\end{aligned}
$$

Example 11.5.4 – *Solution (2/2)* continued

Note that in each case when the subscript n is between two powers of 2, w_n is 1 more than the exponent of the lower power of 2. In other words:

$$\text{If } 2^i \le n < 2^{i+1}, \text{ then } w_n = i + 1. \qquad 11.5.1$$

Now if $$2^i \le n < 2^{i+1},$$

then *[by property* $2^k \le x < 2^{k+1}$, then $\lfloor \log_2 x \rfloor = k$. *]*

$$i = \lfloor \log_2 n \rfloor.$$

Substituting into statement (11.5.1) gives the conjecture that

$$w_n = \lfloor \log_2 n \rfloor + 1$$

The Efficiency of the Binary Search Algorithm (6/6)

The final example shows how to use the formula for w_1, w_2, w_3, … to find a worst-case order for the algorithm.

Example 11.5.6 – *The Binary Search Algorithm Is Logarithmic*

By Example 11.5.5,

$$w_n = \lfloor \log_2 n \rfloor + 1 \qquad \text{for every integer } n \geq 1.$$

Use this result to show that in the worst case, the binary search algorithm is $\Theta(\log_2 n)$.

Example 11.5.6 – *Solution (1/2)*

Let $\Longrightarrow$ stand for the words "which implies that." Then for each integer $n \geq 2$,

$$w_n = \lfloor \log_2 n \rfloor + 1$$

by Example 11.5.5

$$\Longrightarrow \quad \log_2 n \leq w_n \leq \log_2 n + 1$$

because $x < \lfloor x \rfloor + 1$ and $\lfloor x \rfloor \leq x$ for every real number x

$$\Longrightarrow \quad \log_2 n \leq w_n \leq \log_2 n + \log_2 n$$

since the logarithm with base 2 is increasing, when $2 \leq n$, then $1 = \log_2 2 \leq \log_2 n$.

$$\Longrightarrow \quad \log_2 n \leq w_n \leq 2\log_2 n.$$

Example 11.5.6 – *Solution (2/2)* continued

Let $A = 1$, $B = 2$, and $k = 2$. Then

$$A \log_2 n \le w_n \le B \log_2 n \quad \text{for every integer } n \ge k.$$

Hence by definition of Θ-notation,

$$w_n \text{ is } \Theta(\log_2 n).$$

Finally, because w_n, the number of iterations of the **while** loop, is proportional to the number of comparisons performed when the binary search algorithm is executed, the binary search algorithm is $\Theta(\log_2 n)$.

Merge Sort

Merge Sort (1/16)

Note that it is much easier to write a detailed algorithm for sequential search than for binary search. Yet binary search is much more efficient than sequential search. Such trade-offs often occur in computer science.

Frequently, the straightforward “obvious” solution to a problem is less efficient than a clever solution that is more complicated to describe.

Merge Sort (2/16)

We know two methods for sorting, insertion sort and selection sort, both of which are formalizations of methods human beings often use in ordinary situations.

Can a divide-and-conquer approach be used to find a sorting method more efficient than these? It turns out that the answer is an emphatic “yes.” In fact, over the past few decades, computer scientists have developed several divide-and-conquer sorting methods all of which are somewhat more complex to describe but are significantly more efficient than either insertion sort or selection sort.

Merge Sort (3/16)

One of these methods, **merge sort**, is obtained by thinking recursively. Imagine that an efficient way for sorting arrays of length less than *k* is already known. How can such knowledge be used to sort an array of length *k*?

One way is to suppose the array of length *k* is split into two roughly equal parts and each part is sorted using the known method. Is there an efficient way to combine the parts into a sorted array? Sure. Just “merge” them.

Merge Sort (4/16)

Figure 11.5.4 illustrates how a merge works.

Merging Two Sorted Subarrays to Obtain a Sorted Array

Figure 11.5.4

Merge Sort (5/16)

Imagine that the elements of two ordered subarrays—2, 5, 6, 8 and 3, 6, 7, 9—are written on slips of paper (to make them easy to move around). Place the slips for each subarray in two columns on a tabletop, one at the left and one at the right.

Along the bottom of the tabletop, set up eight positions into which the slips will be moved. Then, one-by-one, bring down the slips from the bottoms of the columns.

Merge Sort (6/16)

At each stage compare the numbers on the slips currently at the column bottoms, and move the slip containing the smaller number down into the next position in the array as a whole. If at any stage the two numbers are equal, take, say, the slip on the left to move into the next position.

And if one of the columns is empty at any stage, just move all the slips from the other column into the correct positions one-by-one in order.

Merge Sort (7/16)

The process of merging is efficient because the total number of comparisons needed to merge two subarrays into an array of length *k* is just *k* − 1. You can see why by analyzing Figure 11.5.4.

Merging Two Sorted Subarrays to Obtain a Sorted Array

Figure 11.5.4

Merge Sort (8/16)

Observe that at each stage, the decision about which slip to move is made by comparing the numbers on the slips currently at the bottoms of the two columns—except when one of the columns is empty, in which case no comparisons are made at all.

Thus in the worst case there will be one comparison for each of the k positions in the final array except the very last one (because when the last slip is placed into position, the other column is sure to be empty), or a total of $k - 1$ comparisons in all.

Merge Sort (9/16)

The merge sort algorithm is recursive because its defining statements include references to itself. The algorithm is well defined, however, because at each stage the length of the array that is input to the algorithm is shorter than at the previous stage, so that, ultimately, the algorithm has to deal only with arrays of length 1, which are already sorted. Specifically, merge sort works as follows.

Given an array of elements that can be put into order, if the array consists of a single element, leave it as it is. It is already sorted. Otherwise:

1. Divide the array into two subarrays of as nearly equal length as possible.
2. Use merge sort to sort each subarray.
3. Merge the two subarrays together.

Merge Sort (10/16)

Figure 11.5.5 illustrates a merge sort in a particular case.

Applying Merge Sort to the Array 5, 2, 4, 6, 1, 3, 2, 6

Figure 11.5.5

Merge Sort (11/16)

As in the case of the binary search algorithm, in order to formalize merge sort we must decide at exactly what point to split each array. Given an array denoted by $a[bot], a[bot + 1], \ldots, a[top]$, let $mid = \lfloor (bot + top)/2 \rfloor$.

Take the left subarray to be $a[bot], a[bot + 1], \ldots, a[mid]$ and the right subarray to be $a[mid + 1], a[mid + 2], \ldots, a[top]$.

Merge Sort (12/16)

The following is a formal version of merge sort.

Algorithm 11.5.2 Merge Sort

[The aim of this algorithm is to take an array of elements $a[r], a[r+1], \ldots, a[s]$ *(where* $r \leq s$*) and to order it. The output array is denoted* $a[r], a[r+1], \ldots, a[s]$ *also. It has the same values as the input array, but they are in ascending order. The input array is split into two nearly equal-length subarrays, each of which is ordered using merge sort. Then the two subarrays are merged together.]*

Input: r and s *[positive integers with* $r \leq s$*],* $a[r], a[r+1], \ldots, a[s]$ *[an array of data items that can be ordered]*

Merge Sort (13/16)

continued

Algorithm 11.5.2 Merge Sort

Algorithm Body:

$bot := r$, $top := s$

while $(bot < top)$

$mid := \left\lfloor \frac{bot + top}{2} \right\rfloor$

call **merge sort** with input bot, mid, and

$a[bot], a[bot + 1], \ldots, a[mid]$

call **merge sort** with input $mid + 1$, top and

$a[mid + 1], a[mid + 2], \ldots, a[top]$

[After these steps are completed, both arrays $a[bot], a[bot + 1], \ldots, a[mid]$ and $a[mid + 1], a[mid + 2], \ldots, a[top]$ are in order.]

merge $a[bot], a[bot + 1], \ldots, a[mid]$ and $a[mid + 1], a[mid + 2], \ldots, a[top]$

[This step can be done with a call to a merge algorithm. To put the final array in ascending order, the merge algorithm must be written so as to take two arrays in ascending order and merge them into an array in ascending order.]

end while

Output: $a[r], a[r + 1], \ldots, a[s]$ *[an array with the same elements as the input array but in ascending order]*

Merge Sort (14/16)

To derive the efficiency of merge sort, let

m_n = the maximum number of comparisons used when merge sort is applied to an array of length n.

Then $m_1 = 0$ because no comparisons are used when merge sort is applied to an array of length 1. Also for any integer $k > 1$, consider an array $a[bot], a[bot + 1], \ldots, a[top]$ of length k that is split into two subarrays, $a[bot], a[bot + 1], \ldots, a[mid]$ and $a[mid + 1], a[mid + 2], \ldots, a[top]$, where $mid = \lfloor (bot + top)/2 \rfloor$.

Merge Sort (15/16)

In exercise 24 you are asked to show that the right subarray has length $\lfloor k/2 \rfloor$ and the left subarray has length $\lceil k/2 \rceil$.

From the previous discussion of the merge process, it is known that to merge two subarrays into an array of length k, at most $k - 1$ comparisons are needed.

Merge Sort (16/16)

Consequently,

$$\begin{bmatrix}\text{the number of comparisons}\\ \text{when merge sort is applied}\\ \text{to an array of length } k\end{bmatrix} = \begin{bmatrix}\text{the number of comparisons}\\ \text{when merge sort is applied}\\ \text{to an array of length } \lfloor k/2 \rfloor\end{bmatrix}$$

$$+ \begin{bmatrix}\text{the number of comparisons}\\ \text{when merge sort is applied}\\ \text{to an array of length } \lceil k/2 \rceil\end{bmatrix} + \begin{bmatrix}\text{the number of comparisons}\\ \text{used to merge two subarrays}\\ \text{into an array of length } k.\end{bmatrix}$$

In other words,

$$m_k = m_{\lfloor k/2 \rfloor} + m_{\lceil k/2 \rceil} + (k - 1) \qquad \text{for every integer } k > 1.$$

Tractable and Intractable Problems

Tractable and Intractable Problems (1/5)

At an opposite extreme from an algorithm such as binary search, which has logarithmic order, is an algorithm with exponential order.

For example, consider an algorithm to direct the movement of each of the 64 disks in the Tower of Hanoi puzzle as they are transferred one by one from one pole to another. We know that such a transfer requires $2^{64} - 1$ steps.

Tractable and Intractable Problems (2/5)

If a computer took a nanosecond to calculate each transfer step, the total time to calculate all the steps would be

$$(2^{64} - 1) \cdot \left(\frac{1}{10^9}\right) \cdot \left(\frac{1}{60}\right) \cdot \left(\frac{1}{60}\right) \cdot \left(\frac{1}{24}\right) \cdot \left(\frac{1}{365.25}\right) \cong 584.5 \text{ years.}$$

number of moves

moves per second

seconds per minute

minutes per hour

hours per day

days per year

Tractable and Intractable Problems (3/5)

Problems whose solutions can be found with algorithms whose worst-case order with respect to time is a polynomial are said to belong to **class P**. They are called **polynomial-time algorithms** and are said to be **tractable**.

Problems that cannot be solved in polynomial time are called **intractable**. For certain problems, it is possible to check the correctness of a proposed solution with a polynomial-time algorithm, but it may not be possible to find a solution in polynomial time.

Tractable and Intractable Problems (4/5)

Such problems are said to belong to **class NP**. The biggest open question in theoretical computer science is whether every problem in class NP belongs to class P.

This is known as the **P vs. NP** problem. The Clay Institute, in Cambridge, Massachusetts, has offered a prize of $1,000,000 to anyone who can either prove or disprove that P = NP.

Tractable and Intractable Problems (5/5)

In recent years, computer scientists have identified a fairly large set of problems, called **NP-complete**, that all belong to class NP but are widely believed not to belong to class P.

What is known for sure is that if any one of these problems is solvable in polynomial time, then so are all the others. One of the NP-complete problems, commonly known as the *traveling salesman problem*.

A Final Remark on Algorithm Efficiency

A Final Remark on Algorithm Efficiency (1/2)

This section and the previous one on algorithm efficiency have offered only a partial view of what is involved in analyzing a computer algorithm. For one thing, it is assumed that searches and sorts take place in the memory of the computer.

Searches and sorts on disk-based files require different algorithms, though the methods for their analysis are similar. For another thing, time efficiency is not the only factor that matters in the decision about which algorithm to choose.

A Final Remark on Algorithm Efficiency (2/2)

Although computer memory is now very inexpensive, the amount of memory space required may also be a factor. There are mathematical techniques to estimate space efficiency, which are very similar to those used to estimate time efficiency.

Furthermore, as parallel processing of data becomes increasingly prevalent, current methods of algorithm analysis are being modified and extended to apply to algorithms designed for this new technology.

CHAPTER 12

REGULAR EXPRESSIONS AND FINITE-STATE AUTOMATA

Copyright © Cengage Learning. All rights reserved.

12.1 Formal Languages and Regular Expressions

Copyright © Cengage Learning. All rights reserved.

Formal Languages and Regular Expressions (1/1)

Alphabet Σ:	a finite set of characters
String over Σ:	(1) a finite juxtaposition of elements (called **characters**) of Σ or (2) the null string λ
Length of a string over Σ:	the number of characters that made up the string, with the null string having length 0
Formal language over Σ:	a set of strings over the alphabet

Example 12.1.1 – *Examples of Formal Languages*

Let the alphabet $\Sigma = \{a, b\}$.

a. Define a language L_1 over Σ to be the set of all strings that begin with the character *a* and have length at most three characters. Find L_1.

b. A **palindrome** is a string that looks the same if the order of its characters is reversed. For instance, *aba* and *baab* are palindromes. Define a language L_2 over Σ to be the set of all palindromes obtained using the characters of Σ. Write ten elements of L_2.

Example 12.1.1 – *Solution*

a. $L_1 = \{a, aa, ab, aaa, aab, aba, abb\}$

b. L_2 contains the following ten strings (among infinitely many others):

λ, a, b, aa, bb, aaa, bab, $abba$, $babaabab$, $abaabbbbbaaba$

Formal Languages and Regular Expressions (2/2)

Notation

Let Σ be an alphabet. For each nonnegative integer n, let

$$\Sigma^n = \text{the set of all strings over } \Sigma \text{ that have length } n,$$
$$\Sigma^+ = \text{the set of all strings over } \Sigma \text{ that have length at least 1, and}$$
$$\Sigma^* = \text{the set of all strings over } \Sigma.$$

Note that Σ^n is essentially the Cartesian product of *n* copies of Σ. The language Σ^* is called the **Kleene closure of** Σ, in honor of Stephen C. Kleene (pronounced CLAY-nee). Σ^+ is the set of all strings over Σ except for λ and is called the **positive closure of** Σ.

Example 12.1.2 – *The Languages* Σ^n, Σ^+, *and* Σ^*

Let Σ = {*a*, *b*}.

a. Find $\Sigma^0, \Sigma^1, \Sigma^2$, and Σ^3.

b. Let $A = \Sigma^0 \cup \Sigma^1$ and $B = \Sigma^2 \cup \Sigma^3$. Use words to describe *A*, *B*, and *A* ∪ *B.*

c. Describe a systematic way of writing the elements of Σ^+. What change needs to be made to obtain the elements of Σ^*?

Example 12.1.2 – *Solution (1/2)*

a. $\Sigma^0 = \{\lambda\}$, $\Sigma^1 = \{a, b\}$, $\Sigma^2 = \{aa, ab, ba, bb\}$, and $\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

b. A is the set of all strings over Σ of length at most 1.

B is the set of all strings over Σ of length 2 or 3.

$A \cup B$ is the set of all strings over Σ of length at most 3.

c. Elements of Σ^+ can be written systematically by writing all the strings of length 1, then all the strings of length 2, and so forth.

Example 12.1.2 – *Solution (2/2)* continued

Σ^+ : *a*, *b*, *aa*, *ab*, *ba*, *bb*, *aaa*, *aab*, *aba*, *abb*, *baa*, *bab*, *bba*, *bbb*, *aaaa*, … .

Of course, the process of writing the strings in Σ^+ would continue forever, because Σ^+ is an infinite set. The only change that needs to be made to obtain Σ^* is to place the

null string at the beginning of the list.

Example 12.1.3 – *Polish Notation: A Language Consisting of Postfix Expressions (1/3)*

An expression such as $a + b$, in which a binary operator such as + sits between the two quantities on which it acts, is said to be written in **infix notation**.

Alternative notations are called **prefix notation** (in which the binary operator precedes the quantities on which it acts) and **postfix notation** (in which the binary operator follows the quantities on which it acts). In prefix notation, $a + b$ is written $+ ab$. In postfix notation, $a + b$ is written $ab +$.

Example 12.1.3 – *Polish Notation: A Language Consisting of Postfix Expressions (2/3)*

continued

Prefix and postfix notations were introduced in 1920 by the Polish mathematician Jan Łukasiewicz (pronounced Wu-ca-SHAY-vich).

In his honor—and because some people have difficulty pronouncing his name—they are often referred to as **Polish notation** and **reverse Polish notation**, respectively.

A great advantage of these notations is that they eliminate the need for parentheses in writing arithmetic expressions.

Example 12.1.3 – *Polish Notation: A Language Consisting of Postfix Expressions (3/3)*

continued

If the expression $(a + b) \cdot c$ in infix notation is converted to postfix notation, the result is $ab + c \cdot$.

a. If the expression $ab \cdot cd \cdot +$ in postfix notation is converted to infix notation, what is the result?

b. Let $\Sigma = \{4, 1, +, -\}$, and let L = the set of all strings over Σ obtained by writing either a 4 or a 1 first, then either a 4 or a 1, and finally either a + or a −. List all elements of L between braces, and evaluate the resulting expressions.

Example 12.1.3 – *Solution*

a. $a \cdot b + c \cdot d$

b. $L = \{4\ 1\ +, 4\ 1\ -, 1\ 4\ +, 1\ 4\ -, 4\ 4\ +, 4\ 4\ -, 1\ 1\ +, 1\ 1\ -\}$

$4\ 1+ = 4 + 1 = 5$, $4\ 1 - = 4 - 1 = 3$, $1\ 4+ = 1 + 4 = 5$,

$1\ 4- = 1 - 4 = -3$, $4\ 4 + = 4 + 4 = 8$, $4\ 4- = 4 - 4 = 0$,

$1\ 1+ = 1 + 1 = 2$, $1\ 1 - = 1 - 1 = 0$

Formal Languages and Regular Expressions

The following definition describes ways in which languages can be combined to form new languages.

Definition

Let Σ be an alphabet. Given any strings x and y over Σ, the **concatenation of x and y** is the string obtained by writing all the characters of x followed by all the characters of y. For any languages L and L' over Σ, three new languages can be defined as follows:

The **concatenation of L and L'**, denoted LL', is

$$LL' = \{xy \mid x \in L \text{ and } y \in L'\}.$$

The **union of L and L'**, denoted $L \cup L'$, is

$$L \cup L' = \{x \mid x \in L \text{ or } x \in L'\}.$$

The **Kleene closure of L**, denoted L^*, is

$$L^* = \{x \mid x \text{ is a concatenation of any finite number of strings in } L\}.$$

Note that λ is in L^* because it is regarded as a concatenation of zero strings in L.

Example 12.1.4 – *New Languages from Old*

Let L_1 be the set of all strings consisting of an even number of *a*'s (namely, λ, *aa*, *aaaa*, *aaaaaa*, …), and let $L_2 = \{b, bb, bbb\}$. Find L_1L_2, $L_1 \cup L_2$, and $(L_1 \cup L_2)^*$.

Note that the null string λ is in L_1 because 0 is an even number.

Example 12.1.4 – *Solution*

$L_1 L_2$ = the set of all strings that consist of an even number of *a*'s followed by *b* or by *bb* or by *bbb.*

$L_1 \cup L_2$ 5 the set that includes the strings *b*, *bb*, *bbb* and any strings consisting of an even number of *a*'s.

$(L_1 \cup L_2)^*$ = the set of all strings of *a*'s and *b*'s in which every occurrence of *a* is in a block consisting of an even number of *a*'s.

The Language Defined by a Regular Expression

The Language Defined by a Regular Expression (1/2)

One of the most useful ways to define a language is by means of a *regular expression*, a concept first introduced by Kleene. We give a recursive definition for generating the set of all regular expressions over an alphabet.

Definition

Given an alphabet Σ, the following are **regular expressions over Σ**:

I. Base: $\varnothing$, λ, and each individual symbol in Σ are regular expressions over Σ.

II. Recursion: If r and s are regular expressions over Σ, then the following are also regular expressions over Σ:

$$\text{(i) } (rs) \qquad \text{(ii) } (r \mid s) \qquad \text{(iii) } (r^*),$$

where rs denotes the concatenation of r and s, r^* denotes the concatenation of r with itself any finite number (including zero) of times, and $r \mid s$ denotes either one of the strings r or s. The regular expression r^* is called the **Kleene closure** of r.

III. Restriction: Nothing is a regular expression over Σ except for objects defined in (I) and (II) above.

Example 12.1.5 – *Order of Precedence for the Operations in a Regular Expression*

a. Add parentheses to emphasize the order of precedence in the following expression:

$ab^* \mid b^*a.$

b. Use the rules about order of precedence to eliminate the parentheses in the following expression:

$((a \mid ((b^*)c))(a^*)).$

Example 12.1.5 – *Solution*

a. $((a(b^*)) \mid ((b^*)a))$ b. $(a \mid b^*c)a^*$

The Language Defined by a Regular Expression (2/2)

Given a finite alphabet, every regular expression over the alphabet defines a formal language. The function from regular expressions to formal languages is defined recursively.

Definition

For any finite alphabet Σ, the function L that associates a language to each regular expression over Σ is defined by (I)–(III) below. For each such regular expression r, $L(r)$ is called the **language defined by *r***.

I. Base: $L(\varnothing) = \varnothing$, $L(\lambda) = \{\lambda\}$, $L(a) = \{a\}$ for every a in Σ.

II. Recursion: If $L(r)$ and $L(r')$ are the languages defined by the regular expressions r and r' over Σ, then

(i) $L(rr') = L(r)L(r')$ (ii) $L(r \mid r') = L(r) \cup L(r')$ (iii) $L(r^*) = (L(r))^*$.

III. Restriction: The function L is completely determined by I and II above.

Example 12.1.6 – *Using Set Notation to Describe the Language Defined by a Regular Expression*

Let $\Sigma = \{a, b\}$, and consider the language defined by the regular expression $(a \mid b)^*$. Use set notation to find this language, and describe it in words.

Example 12.1.6 – *Solution*

The language defined by $(a \mid b)^*$ is

$$
\begin{aligned}
L((a \mid b)^*) &= (L(a \mid b))^* \\
&= (L(a) \cup L(b))^* \\
&= (\{a\} \cup \{b\})^* \\
&= \{a, b\}^* \qquad \text{by definition of operations on languages} \\
&= \text{the set of all strings of } a\text{'s and } b\text{'s} \\
&= \Sigma^*.
\end{aligned}
$$

Example 12.1.7 – *The Language Defined by a Regular Expression*

Let $\Sigma = \{0, 1\}$. Use words to describe the languages defined by the following regular expressions over Σ.

a. $0^*1^* \mid 1^*0^*$

b. $0(0 \mid 1)^*$

Example 12.1.7 – *Solution*

a. The strings in this language consist either of a string of 0's followed by a string of 1's or of a string of 1's followed by a string of 0's. However, in either case the strings could be empty, which means that λ is also in the language.

b. The strings in this language have to start with a 0. The 0 may be followed by any finite number (including zero) of 0's and 1's in any order. Thus the language is the set of all strings of 0's and 1's that start with a 0.

Example 12.1.10 – *Deciding Whether Regular Expressions Define the Same Language*

In (a) and (b), determine whether the given regular expressions define the same language. If they do, describe the language. If they do not, give an example of a string that is in one of the languages but not the other.

a. $(a \mid \lambda)^*$ and a^*

b. $0^* \mid 1^*$ and $(01)^*$

Example 12.1.10 – *Solution*

a. Note that because the null string λ has no characters, when it is concatenated with any other string *x*, the result is just *x*: for every string *x*, *x*λ = λ*x* = *x*. Now $L((a \mid \lambda)^*)$ is the set of strings formed using *a* and λ in any order, and so, because *a*λ = λ*a* = *a*, this is the same as the set of strings consisting of zero or more *a*'s. Thus $L((a \mid \lambda)^*) = L(a^*)$.

b. The two languages defined by the given regular expressions are not the same: 0101 is in the second language but not the first.

Practical Uses of Regular Expressions

Practical Uses of Regular Expressions (1/5)

Many applications of computers involve performing operations on pieces of text. For instance, word and text processing programs allow us to find certain words or phrases in a document and possibly replace them with others.

A compiler for a computer language analyzes an incoming stream of characters to find groupings that represent aspects of the computer language such as keywords, constants, identifiers, and operators.

Practical Uses of Regular Expressions (2/5)

And in bioinformatics, pattern matching and flexible searching techniques are used extensively to analyze the long sequences of the base pairs A, C, G, and T that occur in DNA.

Regular expressions provide an extremely useful way to describe a pattern in order to identify a string or a collection of strings within a piece of text. Regular expressions make it possible to replace a long, complicated set of if-then-else statements with code that is easy both to produce and to understand.

Practical Uses of Regular Expressions (3/5)

Because of their convenience, regular expressions were introduced into a number of UNIX utilities, such as *grep* (short for ***g***lobally ***s***earch for ***r***egular ***e***xpression and ***p***rint) and *egrep* (***e****xtended grep*), in text editors, such as *QED* (short for ***Q***uick ***ED***itor, the first text editor to use regular expressions), *vi* (short for ***v***isual ***i***nterface), *sed* (short for ***s***tream ***ed***itor and originally developed for UNIX but now used by many systems), and *Emacs* (short for ***E***ditor ***mac***ro***s***), and in the lexical scanner component of a compiler.

Practical Uses of Regular Expressions (4/5)

The computer language Perl has a particularly powerful implementation for regular expressions, which has become a de facto standard. The implementations used in Java, Python, and .NET are similar.

A number of shorthand notations have been developed to facilitate working with regular expressions in text processing.

Practical Uses of Regular Expressions (5/5)

When characters in an alphabet or in a part of an alphabet are understood to occur in a standard order, the notation *[beginning character – ending character]* is commonly used to represent the regular expression that consists of a single character in the range from the beginning to the ending character. It is called a **character class**.

Example 12.1.11 – *A Regular Expression for a Date*

People often write dates in a variety of formats. For instance, in the United States the following all represent the fifth of February of 2050:

2/5/2050 2-5-2050 02/05/2050 02-05-2050

Write a regular expression that would help check whether a given string might be a valid date written in one of these forms.

Example 12.1.11 – *Solution (1/2)*

The language defined by the following regular expression consists of all strings that begin with one or two digits followed by either a hyphen or a slash, followed by either one or two digits, followed by either a hyphen or a slash, followed by four digits.

[0 − 9] {1, 2} [-/] [0 − 9] {1, 2} [0 − 9] {4}

All valid dates of the given format are elements of the language defined by this expression, but the language also includes strings that are not valid dates.

Example 12.1.11 – *Solution (2/2)* continued

For instance, 09/54/1978 is in the language, but it is not a valid date because September does not have 54 days, and 38/12/2184 is not valid because there is no 38th month.

It is possible to write a more complicated regular expression that could be used to check all aspects of the validity of a date (see exercise 40 at the end of the section), but the kind of simpler expression given above is nonetheless useful.

For instance, it provides an easy way to notify a user of an interactive program that a certain kind of mistake was made and that information should be reentered.

CHAPTER 12

REGULAR EXPRESSIONS AND FINITE-STATE AUTOMATA

Copyright © Cengage Learning. All rights reserved.

12.2 Finite-State Automata

Copyright © Cengage Learning. All rights reserved.

Finite-State Automata (1/4)

A **finite-state automaton** is an idealized machine that embodies the essential idea of a sequential circuit. Each piece of input to a finite-state automaton leads to a change in the state of the automaton, which in turn affects how subsequent input is processed.

Imagine, for example, the act of dialing a telephone number. Dialing 1-800 puts the telephone circuit in a state of readiness to receive the final seven digits of a toll-free call, whereas dialing 328 leads to a state of expectation for the four digits of a local call.

Finite-State Automata (2/4)

Vending machines operate similarly. Just knowing that you put a quarter into a vending machine is not enough for you to be able to predict what the behavior of the machine will be.

You also have to know the state the machine was in when the quarter was inserted. If 75¢ had already been deposited, you might get a beverage or some candy, but if the quarter was the first coin deposited, you would probably get nothing at all.

Example 12.2.1 – *A Simple Vending Machine (1/9)*

A simple vending machine dispenses bottles of juice that cost $1 each. The machine accepts quarters and half-dollars only and does not give change.

As soon as the amount deposited equals or exceeds $1 the machine releases a bottle of juice. The next coin deposited starts the process over again.

Example 12.2.1 – *A Simple Vending Machine (2/9)*

The operation of the machine is represented by the diagram of Figure 12.2.1.

A Simple Vending Machine

Figure 12.2.1

Example 12.2.1 – *A Simple Vending Machine (3/9)*

Each circle represents a state of the machine: the state in which 0¢ has been deposited, 25¢, 50¢, 75¢, and $1 or more. The unlabeled arrow pointing to “0¢ deposited” indicates that this is the initial state of the machine.

The double circle around “$1 or more deposited” indicates that a bottle of juice is released when the machine has reached this state. (It is called an *accepting state* of the machine because when the machine is in this state, it has accepted the input sequence of coins as payment for juice.)

Example 12.2.1 – *A Simple Vending Machine (4/9)*

The arrows that link the states indicate what happens when a particular input is made to the machine in each of its various states.

For instance, the arrow labeled “quarter” that goes from “0¢ deposited” to “25¢ deposited” indicates that when the machine is in the state “0¢ deposited” and a quarter is inserted, the machine goes to the state “25¢ deposited.”

Example 12.2.1 – *A Simple Vending Machine (5/9)*

The arrow labeled “half-dollar” that goes from “75¢ deposited” to “$1 or more deposited” indicates that when the machine is in the state “75¢ deposited” and a half-dollar is inserted, the machine goes to the state “$1 or more deposited” and juice is dispensed.

(In this case the purchaser would pay $1.25 for the juice because the machine does not return change.)

Example 12.2.1 – *A Simple Vending Machine (6/9)*

The arrow labeled “quarter” that goes from “$1 or more deposited” to “25¢ deposited” indicates that when the machine is in the state “$1 or more deposited” and a quarter is inserted, the machine goes back to the state “25¢ deposited.”

(This corresponds to the fact that after the machine has dispensed a bottle of juice, it starts operation all over again.)

Example 12.2.1 – *A Simple Vending Machine (7/9)*

Equivalently, the operation of the vending machine can be represented by a *next-state table* as shown in Table 12.2.1.

		Input	
		Quarter	**Half-Dollar**
→	0¢ deposited	25¢ deposited	50¢ deposited
	25¢ deposited	50¢ deposited	75¢ deposited
State	50¢ deposited	75¢ deposited	$1 or more deposited
	75¢ deposited	$1 or more deposited	$1 or more deposited
◎	$1 or more deposited	25¢ deposited	50¢ deposited

Next-State Table

Table 12.2.1

Example 12.2.1 – *A Simple Vending Machine (8/9)*

The entries in the left-most column are all the possible states of the machine, with the arrow pointing to “0¢ deposited” indicating that the machine begins operation in this state and the double circle next to “$1 or more deposited” showing that a bottle of juice is released when the machine has reached this state.

The inputs to the machine are shown in at the tops of the columns labeled “Quarter” and “Half-Dollar.” Entries in the body of the table show the states to which the machine goes when it has been in one state and a given input is applied to it.

Example 12.2.1 – *A Simple Vending Machine (9/9)*

For instance, the entry in the third row of the column labeled “Half-Dollar” shows that when the machine is in state “50¢ deposited” and a half-dollar is deposited, it goes to state “$1 or more deposited.”

Note that Table 12.2.1 conveys exactly the same information as the diagram of Figure 12.2.1. If the diagram is given, the table can be constructed, and if the table is given, the diagram can be drawn.

Finite-State Automata (3/4)

The most important finite-state automata are digital computers. Each computer consists of several subsystems: input devices, a processor, and output devices.

A processor typically consists of a central processing unit and a finite number of memory locations. At any given time, the state of the processor is determined by the locations and values of all the bits stored within its memory.

A computer that has *n* different locations for storing a single bit can therefore exist in 2^n different states.

Finite-State Automata (4/4)

The basic theory of automata was developed to answer very theoretical questions about the foundations of mathematics posed by the great German mathematician David Hillbert in 1900.

The ground-breaking work on automata was done in the mid-1930s by the English mathematician and logician Alan Turing. In the 1940s and 1950s, Turing's work played an important role in the development of real-world automatic computers.

Definition of a Finite-State Automaton

Definition of a Finite-State Automaton (1/5)

A general *finite-state automaton* is completely described by giving a set of states, together with an indication about which is the initial state and which are the accepting states (when something special happens), a list of all input elements, and specification for a *next-state function* that defines which state is produced by each input in each state.

Definition of a Finite-State Automaton (2/5)

This is formalized in the following definition:

Definition

A **finite-state automaton** A consists of five objects:

1. A finite set I, called the **input alphabet**, of input symbols.
2. A finite set S of **states** the automaton can assume.
3. A designated state s_0 called the **initial state**.
4. A designated set of states called the set of **accepting states**.
5. A **next-state function** $N: S \times I \rightarrow S$ that associates a "next-state" to each ordered pair consisting of a "current state" and a "current input." For each state s in S and input symbol m in I, $N(s, m)$ is the state to which A goes if m is input to A when A is in state s.

Definition of a Finite-State Automaton (3/5)

The operation of a finite-state automaton is commonly described by a diagram called a **(state-) transition diagram**, similar to that of Figure 12.2.1.

A Simple Vending Machine

Figure 12.2.1

Definition of a Finite-State Automaton (4/5)

It is called a *transition diagram* because it shows the transitions the machine makes from one state to another in response to various inputs. In a transition diagram, states are represented by circles and accepting states by double circles.

There is one arrow that points to the initial state and there are other arrows that are labeled with input symbols and point from each state to other states to indicate the action of the next-state function. Specifically, an arrow from state s to state t labeled m means that $N(s, m) = t$.

Definition of a Finite-State Automaton (5/5)

The **next-state table** for an automaton shows the values of the next-state function *N* for all possible states *s* and input symbols *i*.

In the **annotated next-state table**, the initial state is indicated by an arrow and the accepting states are marked by double circles.

Example 12.2.2 – *A Finite-State Automaton Given by a Transition Diagram (1/2)*

Consider the finite-state automaton *A* defined by the transition diagram shown in Figure 12.2.2.

Figure 12.2.2

Example 12.2.2 – *A Finite-State Automaton Given by a Transition Diagram (2/2)*

a. What are the states of *A*?

b. What are the input symbols of *A*?

c. What is the initial state of *A*?

d. What are the accepting states of *A*?

e. Determine $N(s_1, 1)$.

f. Construct the annotated next-state table for *A.*

Example 12.2.2 – *Solution (1/2)*

a. The states of A are s_0, s_1, and s_2 *[since these are the labels of the circles]*.

b. The input symbols of A are 0 and 1 *[since these are the labels of the arrows]*.

c. The initial state of A is s_0 *[since the unlabeled arrow points to s_0]*.

d. The only accepting state of A is s_2 *[since this is the only state marked by a double circle]*.

Example 12.2.2 – *Solution (2/2)* continued

e. $N(s_1, 1) = s_2$ *[since there is an arrow from* s_1 *to* s_2 *labeled* 1*]*

f.

			Input	
			0	1
State	→	s_0	s_1	s_0
		s_1	s_1	s_2
	◎	s_2	s_1	s_0

Example 12.2.3 – *A Finite-State Automaton Given by an Annotated Next-State Table (1/2)*

Consider the finite-state automaton *A* defined by the following annotated next-state table:

			Input		
			a	b	c
State	→	U	Z	Y	Y
	◎	V	V	V	V
		Y	Z	V	Y
	◎	Z	Z	Z	Z

Example 12.2.3 – *A Finite-State Automaton Given by an Annotated Next-State Table (2/2)* continued

a. What are the states of A?

b. What are the input symbols of A?

c. What is the initial state of A?

d. What are the accepting states of A?

e. Find $N(U, c)$.

f. Draw the transition diagram for A.

Example 12.2.3 – *Solution (1/2)*

a. The states of *A* are *U*, *V*, *Y*, and *Z*.

b. The input symbols of *A* are *a*, *b*, and *c*.

c. The initial state of *A* is *U [since the arrow points to U]*.

d. The accepting states of *A* are *V* and *Z [since these are marked with double circles]*.

e. *N*(*U*, *c*) = *Y [since the entry in the row labeled U and the column labeled c of the next state table is Y]*.

Example 12.2.3 – *Solution (2/2)* continued

f. The transition diagram for *A* is shown in Figure 12.2.3. It can be drawn more compactly by labeling arrows with multiple-input symbols where appropriate. This is illustrated in Figure 12.2.4.

Figure 12.2.3

Figure 12.2.4

The Language Accepted by an Automaton

The Language Accepted by an Automaton (1/1)

Definition

Let A be a finite-state automaton with set of input symbols I. Let I^* be the set of all strings over I, and let w be a string in I^*. Then **w is accepted by A** if, and only if, A goes to an accepting state when the symbols of w are input to A in sequence from left to right, starting when A is in its initial state. The **language accepted by A**, denoted $\boldsymbol{L(A)}$, is the set of all strings that are accepted by A.

Example 12.2.4 – *Finding the Language Accepted by an Automaton (1/2)*

Consider the finite-state automaton *A* defined in Example 12.2.2 and shown again below.

a. To what states does *A* go if the symbols of the following strings are input to *A* in sequence, starting from the initial state?

(i) 01 (ii) 0011 (iii) 0101100 (iv) 10101

Example 12.2.4 – *Finding the Language Accepted by an Automaton (2/2)*

continued

b. Which of the strings in part (a) send *A* to an accepting state?

c. What is the language accepted by *A*?

d. Is there a regular expression that defines the same language?

Example 12.2.4 – *Solution (1/3)*

a. (i) s_2 (ii) s_0 (iii) s_1 (iv) s_2

b. The strings 01 and 10101 send A to an accepting state.

c. Observe that if w is any string that ends in 01, then w is accepted by A. For if w is any string of length $n \geq 2$, then after the first $n - 2$ symbols of w have been input, A is in one of its three states: s_0, s_1, or s_2. But from any of these three states, input of the symbols 01 in sequence sends A first to s_1 and then to the accepting state s_2. Hence any string that ends in 01 is accepted by A.

Example 12.2.4 – *Solution (2/3)* continued

Also note that the only strings accepted by A are those that end in 01. (That is, no other strings besides those ending in 01 are accepted by A.) The reason is that the only accepting state of A is s_2, and the only arrow pointing to s_2 comes from s_1 and is labeled 1.

Thus in order for an input string w of length n to send A to an accepting state, the last symbol of w must be a 1 and the first $n - 1$ symbols of w must send A to state s_1.

Example 12.2.4 – *Solution (3/3)* continued

Now three arrows point to s_1, one from each of the three states of A, and all are labeled 0. Thus the last of the first $n - 1$ symbols of w must be 0, or, in other words, the next-to-the-last symbol of w must be 0. Hence the last two symbols of w must be 01, and thus

$L(A)$ = the set of all strings of 0's and 1's that end in 01.

d. Yes. One regular expression that defines $L(A)$ is $(0 \mid 1)^*01$.

The Eventual-State Function

The Eventual-State Function (1/3)

Now suppose a finite-state automaton is in one of its states (not necessarily the initial state) and a string of input symbols is fed into it in sequence. To what state will the automaton eventually go?

The function that gives the answer to this question for every possible combination of input strings and states of the automaton is called the *eventual-state function.*

The Eventual-State Function (2/3)

Definition

Let A be a finite-state automaton with set of input symbols I, set of states S, and next-state function $N: S \times I \longrightarrow S$. Let I^* be the set of all strings over I, and define the **eventual-state function** $N^*: S \times I^* \longrightarrow S$ as follows:

For any state s and for any input string w,

$$N^*(s, w) = \begin{bmatrix} \text{the state to which } A \text{ goes if the} \\ \text{symbols of } w \text{ are input to } A \text{ in sequence,} \\ \text{starting when } A \text{ is in state } s \end{bmatrix}.$$

Example 12.2.5 – *Computing Values of the Eventual-State Function*

Consider again the finite-state automaton of Example 12.2.2 shown below for convenience. Find $N^*(s_1, 10110)$.

Example 12.2.5 – *Solution (1/2)*

By definition of the eventual-state function,

$$N^*(s_1, 10110) = \begin{bmatrix} \text{the state to which } A \text{ goes if the} \\ \text{symbols of } 10110 \text{ are input to } A \text{ in} \\ \text{sequence, starting when } A \text{ is in state } s_1 \end{bmatrix}.$$

Example 12.2.5 – *Solution (2/2)*

By referring to the transition diagram for A, you can see that starting from s_1, when a 1 is input, A goes to s_2; then when a 0 is input, A goes back to s_1; after that, when a 1 is input, A goes to s_2; from there, when a 1 is input, A goes to s_0; and finally, when a 0 is input, A goes back to s_1. This sequence of state transitions can be written as follows:

$$s_1 \xrightarrow{1} s_2 \xrightarrow{0} s_1 \xrightarrow{1} s_2 \xrightarrow{1} s_0 \xrightarrow{0} s_1.$$

Thus, after all the symbols of 10110 have been input in sequence, the eventual state of A is s_1, so

$$N^*(s_1, 10110) = s_1.$$

The Eventual-State Function (3/3)

The definitions of string and language accepted by an automaton can be restated symbolically using the eventual-state function. Suppose *A* is a finite-state automaton with set of input symbols *I* and next-state function *N*, and suppose that I^* is the set of all strings over *I* and that *w* is a string in I^*.

$$w \text{ is accepted by } A \quad \Leftrightarrow \quad N^*(s_0, w) \text{ is an accepting state of } A$$

$$L(A) = \{w \in I^* \mid N^*(s_0, w) \text{ is an accepting state of } A\}$$

Designing a Finite-State Automaton

Designing a Finite-State Automaton (1/1)

Now consider the problem of starting with a description of a language and designing an automaton to accept exactly that language.

Example 12.2.6 – *A Finite-State Automaton That Accepts the Set of Strings of 0's and 1's for Which the Number of 1's Is Divisible by 3*

a. Design a finite-state automaton *A* that accepts the set of all strings of 0's and 1's such that the number of 1's in the string is divisible by 3.

b. Is there a regular expression that defines this set?

Example 12.2.6 – *Solution (1/6)*

a. Let s_0 be the initial state of A, s_1 its state after one 1 has been input, and s_2 its state after two 1's have been input. Note that s_0 is the state of A after zero 1's have been input, and since zero is divisible by 3 ($0 = 0 \cdot 3$), s_0 must be an accepting state.

The states s_0, s_1, and s_2 must be different from one another because from state s_0 three 1's are needed to reach a new total divisible by 3, whereas from state s_1 two additional 1's are necessary, and from state s_2 just one more 1 is required.

Example 12.2.6 – *Solution (2/6)* continued

Now the state of A after three 1's have been input can also be taken to be s_0 because after three 1's have been input, three more are needed to reach a new total divisible by 3.

More generally, if $3k$ 1's have been input to A, where k is any nonnegative integer, then three more are needed for the total again to be divisible by 3 (since $3k + 3 = 3(k + 1)$). Thus the state in which $3k$ 1's have been input, for any nonnegative integer k, can be taken to be the initial state s_0.

Example 12.2.6 – *Solution (3/6)* continued

By similar reasoning, the states in which $(3k + 1)$ 1's and $(3k + 2)$ 1's have been input, where k is a nonnegative integer, can be taken to be s_1 and s_2, respectively.

Now every nonnegative integer can be written in one of the three forms $3k$, $3k + 1$, or $3k + 2$, so the three states s_0, s_1, and s_2 are all that is needed to create A. Thus the states of A can be drawn and labeled as shown below.

Example 12.2.6 – *Solution (4/6)* continued

Next consider the possible inputs to A in each of its states. No matter what state A is in, if a 0 is input the total number of 1's in the input string remains unchanged. Thus there is a loop at each state labeled 0.

Now suppose a 1 is input to A when it is in state s_0. Then A goes to state s_1 (since the total number of 1's in the input string has changed from $3k$ to $3k + 1$). Similarly, if a 1 is input to A when it is in state s_1, then A goes to state s_2 (since the total number of 1's in the input string has changed from $3k + 1$ to $3k + 2$).

Example 12.2.6 – *Solution (5/6)* continued

Finally, if a 1 is input to A when it is in state s_2, then it goes to state s_0 (since the total number of 1's in the input string becomes $(3k + 2) + 1 = 3k + 3 = 3(k + 1)$, which is a multiple of 3).

It follows that the transition diagram for A has the appearance shown below.

This automaton accepts the set of strings of 0's and 1's for which the number of 1's is divisible by 3.

Example 12.2.6 – *Solution (6/6)* continued

b. A regular expression that defines the given set is

$$0^* \mid (0^*10^* \; 10^*10^*)^*.$$

Simulating a Finite-State Automaton Using Software

Simulating a Finite-State Automaton Using Software (1/5)

Suppose items have been coded with strings of 0's and 1's. A program is to be written to govern the processing of items coded with strings that end in 011; items coded any other way are to be ignored. This situation can be modeled by the finite-state automaton shown in Figure 12.2.5.

Figure 12.2.5

This automaton recognizes strings that end in 011.

Simulating a Finite-State Automaton Using Software (2/5)

The symbols of the code for the item are fed into this automaton in sequence, and every string of symbols in a given code sends the automaton to one of the four states s_0, s_1, s_2, or s_3. If state s_3 is reached, the item is processed; if not, the item is ignored.

The action of this finite-state automaton can be simulated by a computer algorithm as given in Algorithm 12.2.1.

Simulating a Finite-State Automaton Using Software (3/5)

Algorithm 12.2.1 A Finite-State Automaton

[This algorithm simulates the action of the finite-state automaton of Figure 12.2.5 by mimicking the functioning of the transition diagram. The states are denoted 0, 1, 2, *and* 3.*]*

Input: string *[a string of* 0*'s and* 1*'s plus an end marker e]*

Algorithm Body:

```
state := 0
symbol := first symbol in the input string
while (symbol ≠ e)
    if   state = 0   then if symbol = 0
                          then state := 1
                          else state = 0
    else if state = 1 then if symbol = 0
                          then state := 1
                          else state := 2
```

Simulating a Finite-State Automaton Using Software (4/5)

Algorithm 12.2.1 A Finite-State Automaton

```
        else if state = 2 then if symbol = 0
                                then state := 1
                                else state := 3
        else if state = 3 then if symbol = 0
                                then state := 1
                                else state := 0
        symbol := next symbol in the input string
    end while
    [After execution of the while loop, the value of state is 3 if, and only if, the input
    string ends in 011e.]
Output: state
```

Simulating a Finite-State Automaton Using Software (5/5)

Algorithm 12.2.2 A Finite-State Automaton

[This algorithm simulates the action of the finite-state automaton of Figure 12.2.5 by repeated application of the next-state function. The states are denoted 0, 1, 2, *and* 3.*]*

Input: *string [a string of* 0*'s and* 1*'s plus an end marker e]*

Algorithm Body:

$N(0, 0) := 1$, $N(0, 1) := 0$, $N(1, 0) := 1$, $N(1, 1) := 2$,
$N(2, 0) := 1$, $N(2, 1) := 3$, $N(3, 0) := 1$, $N(3, 1) := 0$
state := 0
symbol := first symbol in the input string

while (*symbol* $\neq$ *e*)
  state := *N* (*state, symbol*)
  symbol := next symbol in the input string
end while

[After execution of the **while** *loop, the value of state is* 3 *if, and only if, the input string ends in* 011*e.]*

Output: *state*

Finite-State Automata and Regular Expressions

Finite-State Automata and Regular Expressions (1/2)

Each time we considered a language accepted by a finite-state automaton, we found a regular expression that defined the same language. Stephen Kleene showed that our ability to do this is not sheer coincidence.

He proved that any language accepted by a finite-state automaton can be defined by a regular expression and that, conversely, any language defined by a regular expression is accepted by a finite-state automaton.

Finite-State Automata and Regular Expressions (2/2)

Kleene's Theorem, Part 1

Given any language that is accepted by a finite-state automaton, there is a regular expression that defines the same language.

Kleene's Theorem, Part 2

Given any language defined by a regular expression, there is a finite-state automaton that accepts the same language.

Regular Languages

Regular Languages (1/1)

According to Kleene's theorem, the set of languages defined by regular expressions is identical to the set of languages accepted by finite-state automata. Any such language is called a **regular language**.

Example 12.2.8 – *Showing That a Language Is Not Regular*

Let the language *L* consist of all strings of the form $a^k b^k$, where *k* is a positive integer. Symbolically, *L* is the language over the alphabet Σ = {*a*, *b*} defined by

$L = \{w \in \Sigma^* \mid w = a^k b^k$, where *k* is a positive integer}.

Use the pigeonhole principle to show that *L* is not regular. In other words, show that there is no finite-state automaton that accepts *L.*

Example 12.2.8 – *Solution (1/6)*

[Use a proof by contradiction.] Suppose not. That is, suppose there is a finite-state automaton *A* that accepts *L*.

[A contradiction will be derived.] Since *A* has only a finite number of states, these states can be denoted s_1, s_2, s_3, . . . , s_n, where *n* is a positive integer. Consider all input strings that consist entirely of a's: $a, a^2, a^3, a^4, \ldots$.
Now there are infinitely many such strings and only finitely many states.

Example 12.2.8 – *Solution (2/6)* continued

Thus, by the pigeonhole principle, there must be a state s_m and two input strings a^p and a^q with $p \neq q$ such that when either a^p or a^q is input to A, A goes to state s_m. (See Figure 12.2.6.)

Since F is not one-to-one, ∃ strings a^p and a^q with $p \neq q$ such that both a^p and a^q send A to the same state s_m.

Figure 12.2.6

Example 12.2.8 – *Solution (3/6)*

[The pigeons are the strings of a's, the pigeonholes are the states, and the correspondence associates each string with the state to which A goes when the string is input.]

This means that after *p a*'s have been input, at which point *A* is in state s_m, inputting *p* additional *b*'s sends *A* into an accepting state, say s_a.

Example 12.2.8 – *Solution (4/6)*

But that implies that

$$a^q b^p$$

also sends *A* to the accepting state s_a, and so $a^q b^p$ is accepted by *A*. The reason is that after *q* *a*'s have been input, *A* is also in state s_m, and from that point, inputting *p* additional *b*'s sends *A* to state s_a, which is an accepting state.

Example 12.2.8 – *Solution (5/6)*

Pictorially, if $p < q$, then

Example 12.2.8 – *Solution (6/6)*

Now, by supposition, *L* is the language accepted by *A.* Thus since $a^q b^p$ is accepted by *A*, $a^q b^p \in L.$

But by definition of *L*, *L* consists only of strings with equal numbers of *a*'s and *b*'s. So since $p \neq q$, $a^q b^p \notin L.$ Hence $a^q b^p \in L \text{ and } a^q b^p \notin L,$ which is a contradiction.

It follows that the supposition is false, and so there is no finite-state automaton that accepts *L*.

CHAPTER 12

REGULAR EXPRESSIONS AND FINITE-STATE AUTOMATA

Copyright © Cengage Learning. All rights reserved.

12.3 Simplifying Finite-State Automata

Copyright © Cengage Learning. All rights reserved.

Example 12.3.1 – *An Overview (1/4)*

Consider the finite-state automata *A* and *A′* in Figure 12.3.1.

Two Equivalent Automata

Figure 12.3.1

A moment's thought should convince you that *A′* accepts all those strings, and only those strings, that contain an even number of 1's.

Example 12.3.1 – *An Overview (2/4)* continued

But *A*, although it appears more complicated, accepts exactly those strings also. Thus the two automata are "equivalent" in the sense that they accept the same language, even though *A′* has fewer states than *A*.

Roughly speaking, the reason for the equivalence of these automata is that some of the states of *A* can be combined without affecting the acceptance or nonacceptance of any input string.

Example 12.3.1 – *An Overview (3/4)* continued

It turns out that s_2 can be combined with state s_0 and that s_3 can be combined with state s_1. (How to figure out which states can be combined is explained later in this section.)

The automaton with the two combined states $\{s_0, s_2\}$ and $\{s_1, s_3\}$ is called the *quotient automaton* of A and is denoted $\underline{A}$.

Its transition diagram is obtained by combining the circles for s_0 and s_2 and for s_1 and s_3 and by replacing any arrow from a state s to a state t by an arrow from the combined state containing s to the combined state containing t.

Example 12.3.1 – *An Overview (4/4)* continued

For instance, since there is an arrow labeled 1 from s_1 to s_2 in A, there is an arrow labeled 1 from $\{s_1, s_3\}$ to $\{s_0, s_2\}$ in $\overline{A}$.

The complete transition diagram for $\overline{A}$ is shown in Figure 12.3.2.

Transition Diagram for $\overline{A}$

Figure 12.3.2

As you can see, except for labeling the names of the states, it is identical to the diagram for A'.

*-Equivalence of States

*-Equivalence of States (1/4)

Two states of a finite-state automaton are said to be *-*equivalent* (this is read “star equivalent”) if, and only if, any string accepted by the automaton when it starts from one of the states is accepted by the automaton when it starts from the other state.

Recall that the value of the eventual-state function, *N**, for a state *s* and input string *w* is the state to which the automaton goes if the characters of *w* are input in sequence when the automaton is in state *s.*

*-Equivalence of States (2/4)

Definition

Let A be a finite-state automaton with next-state function N and eventual-state function N^*. Define a binary relation on the set of states of A as follows: Given any states s and t of A, we say that ***s* and *t* are *-equivalent** and write $s \; R_* \; t$ if, and only if, for each input string w,

either both $N^*(s, w)$ and $N^*(t, w)$ are accepting states or both are nonaccepting states.

*-Equivalence of States (3/4)

In other words, states *s* and *t* are ***-equivalent** if, and only if, for each input string *w,*

$N^*(s, w)$ is an accepting state $\Leftrightarrow$ $N^*(t, w)$ is an accepting state.

Or, more simply, for each input string *w*,

$$\begin{bmatrix} A \text{ goes to an accepting state if} \\ w \text{ is input when } A \text{ is in state } s \end{bmatrix} \Leftrightarrow \begin{bmatrix} A \text{ goes to an accepting state if} \\ w \text{ is input when } A \text{ is in state } t \end{bmatrix}.$$

*-Equivalence of States (4/4)

It follows immediately, by substitution into the definition, that

R_* is an equivalence relation on S, the set of states of A. 12.3.1

k-Equivalence of States

k-Equivalence of States (1/6)

From a procedural point of view, it is difficult to determine the *-equivalence of two states using the definition directly. According to the definition, you must know the action of the automaton starting in states *s* and *t* on *all* input strings in order to tell whether *s* and *t* are equivalent.

But since most languages have infinitely many input strings, you cannot check individually the effect of every string that is input to an automaton.

k-Equivalence of States (2/6)

As a practical matter, you can tell whether or not two states *s* and *t* are *-equivalent by using an iterative procedure based on a simpler kind of equivalence of states called *k-equivalence.*

Two states are *k-equivalent* if, and only if, any string *of length less than or equal to k* that is accepted by the automaton when it starts from one of the states is accepted by the automaton when it starts from the other state.

k-Equivalence of States (3/6)

Definition

Let A be a finite-state automaton with next-state function N and eventual-state function N^*. Define a relation on the set of states of A as follows: Given any states s and t of A and an integer $k \geq 0$, we say that s is ***k*-equivalent** to t and write $\boldsymbol{s\ R_k\ t}$ if, and only if, for every input string *w of length less than or equal to k*, either $N^*(s, w)$ and $N^*(t, w)$ are both accepting states or they are both nonaccepting states.

k-Equivalence of States (4/6)

Certain useful facts follow quickly from the definition of k-equivalence:

For each integer $k \geq 0$, k-equivalence is an equivalence relation.	12.3.2
For each integer $k \geq 0$, the k-equivalence classes partition the set of all states of the automaton into a union of mutually disjoint subsets.	12.3.3
For each integer $k \geq 1$, if two states are k-equivalent, then they are also $(k-1)$ equivalent.	12.3.4
For each integer $k \geq 1$, each k-equivalence class is a subset of a $(k-1)$-equivalence class.	12.3.5
Any two states that are k-equivalent for every integer $k \geq 0$ are $*$-equivalent.	12.3.6

k-Equivalence of States (5/6)

The following theorem gives a recursive description of k-equivalence of states.

Theorem 12.3.1

Let A be a finite-state automaton with next-state function N. Given any states s and t in A,

1. s is 0-equivalent to t $\Leftrightarrow$ $\left[\begin{array}{l}\text{either } s \text{ and } t \text{ are both accepting states} \\ \text{or they are both nonaccepting states}\end{array}\right]$

2. for every integer $k \geq 1$, s is k-equivalent to t $\Leftrightarrow$ $\left[\begin{array}{l}s \text{ and } t \text{ are } (k-1)\text{-equivalent, and} \\ \text{for any input symbol } m, N(s, m) \text{ and} \\ N(t, m) \text{ are also } (k-1)\text{-equivalent}\end{array}\right]$.

k-Equivalence of States (6/6)

It says, first, that any two states are 0-equivalent if, and only if, either both are accepting states or both are nonaccepting states and, second, that any two states are k-equivalent (for $k \geq 1$) if, and only if, (1) they are $(k - 1)$-equivalent and (2) for any input symbols their next-states are also $(k - 1)$-equivalent.

Example 12.3.2 – *Finding k-Equivalence Classes*

Find the 0-equivalence classes, the 1-equivalence classes, and the 2-equivalence classes for the states of the automaton shown below.

Example 12.3.2 – *Solution (1/7)*

1. **0-*equivalence classes:*** By Theorem 12.3.1 two states are 0-equivalent if, and only if, both are accepting states or both are nonaccepting states. Thus there are two sets of 0-equivalent states:

   $\{s_0, s_1, s_4\}$ (the nonaccepting states) and $\{s_2, s_3\}$ (the accepting states),

   and so

   the 0-equivalence classes are $\{s_0, s_1, s_4\}$ and $\{s_2, s_3\}$.

Example 12.3.2 – *Solution (2/7)* continued

2. **1-*equivalence classes:*** By Theorem 12.3.1, two states are 1-equivalent if, and only if, they are 0-equivalent and, after input of any input symbol, their next-states are 0-equivalent.

Theorem 12.3.1

Let A be a finite-state automaton with next-state function N. Given any states s and t in A,

1. s is 0-equivalent to t $\Leftrightarrow$ $\left[\begin{array}{l}\text{either } s \text{ and } t \text{ are both accepting states} \\ \text{or they are both nonaccepting states}\end{array}\right]$

2. for every integer $k \geq 1$, s is k-equivalent to t $\Leftrightarrow$ $\left[\begin{array}{l}s \text{ and } t \text{ are } (k-1)\text{-equivalent, and} \\ \text{for any input symbol } m, N(s, m) \text{ and} \\ N(t, m) \text{ are also } (k-1)\text{-equivalent}\end{array}\right]$.

Example 12.3.2 – *Solution (3/7)* continued

Thus s_1 is not 1-equivalent to s_0 because when a 0 is input to the automaton in state s_1 it goes to state s_2, whereas when a 0 is input to the automaton in state s_0 it goes to state s_0, and s_2 and s_0 are not 0-equivalent.

On the other hand, s_1 *is* 1-equivalent to s_4 because when a 0 is input to the automaton in state s_1 or s_4 the next-states are s_2 and s_3, which are 0-equivalent; and when a 1 is input to the automaton in state s_1 or s_4 the next-states are s_4 and s_1, which are 0-equivalent.

Example 12.3.2 – *Solution (4/7)* continued

By a similar argument, s_2 is 1-equivalent to s_3. Since 1-equivalent states must also be 0-equivalent *[by property* (12.3.4)-*]*, no other pairs of states can be 1-equivalent.

Hence the 1-equivalence classes are $\{s_0\}$, $\{s_1, s_4\}$, and $\{s_2, s_3\}$.

Example 12.3.2 – *Solution (5/7)* continued

3. ***2-equivalence classes:*** By Theorem 12.3.1, two states are 2-equivalent if, and only if, they are 1-equivalent and, after input of any input symbol, their next-states are 1-equivalent.

Theorem 12.3.1

Let A be a finite-state automaton with next-state function N. Given any states s and t in A,

1. s is 0-equivalent to t $\Leftrightarrow$ $\left[\begin{array}{l}\text{either } s \text{ and } t \text{ are both accepting states} \\ \text{or they are both nonaccepting states}\end{array}\right]$

2. for every integer $k \geq 1$, s is k-equivalent to t $\Leftrightarrow$ $\left[\begin{array}{l}s \text{ and } t \text{ are } (k-1)\text{-equivalent, and} \\ \text{for any input symbol } m, N(s, m) \text{ and} \\ N(t, m) \text{ are also } (k-1)\text{-equivalent}\end{array}\right]$.

Example 12.3.2 – *Solution (6/7)* continued

Now s_1 is 2-equivalent to s_4 because they are 1-equivalent; and when a 1 is input to the automaton in state s_1 or s_4 the next-states are s_4 and s_1, which are 1-equivalent; and when a 0 is input to the automaton in state s_1 or s_4 the next-states are s_2 and s_3, which are 1-equivalent.

Similarly, s_2 is 2-equivalent to s_3. Since 2-equivalent states must also be 1-equivalent *[by property* (12.3.4)*]*, no other pairs of states can be 2-equivalent.

Example 12.3.2 – *Solution (7/7)* continued

Hence

the 2-equivalence classes are $\{s_0\}$, $\{s_1, s_4\}$, and $\{s_2, s_3\}$.

Note that the set of 2-equivalence classes equals the set of 1-equivalence classes.

Finding the *-Equivalence Classes

Finding the *-Equivalence Classes (1/1)

Example 12.3.2 illustrates the relative ease with which the sets of *k*-equivalence classes of states can be found. But to simplify a finite-state automaton, you need to find the set of *-equivalence classes of states. The next theorem says that for some integer *K*, the set of *-equivalence classes equals the set of *K*-equivalence classes.

Theorem 12.3.2

If A is a finite-state automaton, then for some integer, $K \geq 0$, the set of K-equivalence classes of states of A equals the set of $(K+1)$-equivalence classes of states of A, and for all such K these are both equal to the set of $*$-equivalence classes of states of A.

Example 12.3.3 – *Finding *-Equivalence Classes of R*

Let *A* be the finite-state automaton defined in Example 12.3.2. Find the *-equivalence classes of states of *A*.

Example 12.3.3 – *Solution*

According to Example 12.3.2, the set of 1-equivalence classes for A equals the set of 2-equivalence classes. By Theorem 12.3.2, then, the set of *-equivalence classes also equals the set of 1-equivalence classes.

Hence

the *-equivalence classes are $\{s_0\}$, $\{s_1, s_4\}$, and $\{s_2, s_3\}$.

The equivalence classes are denoted

$[s_0] = \{s_0\}$ $\quad$ $[s_1] = \{s_1, s_4\} = [s_4]$ $\quad$ $[s_2] = \{s_2, s_3\} = [s_3]$.

The Quotient Automaton

The Quotient Automaton (1/5)

We next define the *quotient automaton* $\overline{A}$ of an automaton *A*.

However, in order for all parts of the definition to make sense, we must make two observations.

No $*$-equivalence class of states of A can contain both accepting and nonaccepting states.	12.3.7

The Quotient Automaton (2/5)

The reason this is true is that the 0-equivalence classes divide the set of states of *A* into accepting and nonaccepting states, and the *-equivalence classes are subsets of 0-equivalence classes.

If two states are $*$-equivalent, then their next-states are also $*$-equivalent for each input symbol m. 12.3.8

The Quotient Automaton (3/5)

Now we can define the quotient automaton *A* of *A*. It is the finite-state automaton whose states are the *-equivalence classes of states of *A*, whose initial state is the *-equivalence class containing the initial state of *A*, whose accepting states are of the form [*s*] where *s* is an accepting state of *A*, whose input symbols are the same as the input symbols of *A*, and whose next-state function is derived from the next-state function for *A* in the following way: To find the next-state of $\overline{A}$ for a state *s* and an input symbol *m*, pick any state *t* in [*s*] and look to see what next-state *A* goes to if *m* is input when *A* is in state *t*; the equivalence class of this state is the next-state of $\overline{A}$.

The Quotient Automaton (4/5)

Definition

Let A be a finite-state automaton with set of states S, set of input symbols I, and next-state function N. The **quotient automaton** $\overline{A}$ is defined as follows:

1. The set of states, $\overline{S}$, of $\overline{A}$ is the set of $*$-equivalence classes of states of A.
2. The set of input symbols, $\overline{I}$, of $\overline{A}$ equals I.
3. The initial state of $\overline{A}$ is $[s_0]$, where s_0 is the initial state of A.
4. The accepting states of $\overline{A}$ are the states of the form $[s]$, where s is an accepting state of A.
5. The next-state function $\overline{N}: \overline{S} \times I \longrightarrow \overline{S}$ is defined as follows:

$$\text{For all states } [s] \text{ in } \overline{S} \text{ and input symbols } m \text{ in } I, \quad \overline{N}([s], m) = [N(s, m)].$$

(That is, if m is input to $\overline{A}$ when $\overline{A}$ is in state $[s]$, then $\overline{A}$ goes to the state that is the $*$-equivalence class of $N(s, m)$.)

The Quotient Automaton (5/5)

Theorem 12.3.3

If A is a finite-state automaton, then the quotient automaton $\overline{A}$ accepts exactly the same languages as A. In other words, if $L(A)$ denotes the language accepted by A and $L(\overline{A})$ denotes the language accepted by $\overline{A}$, then

$$L(A) = L(\overline{A}).$$

Constructing the Quotient Automaton

Constructing the Quotient Automaton (1/2)

Let A be a finite-state automaton with set of states S, next-state function N, relation R^* of *-equivalence of states, and relation R_k of k-equivalence of states. It follows from Theorems 12.3.2 and 12.3.3 and from the definition of quotient automaton that to find the quotient automaton $\overline{A}$ of A, you can proceed as follows:

1. Find the set of 0-equivalence classes of S.

2. For each integer $k \geq 1$, subdivide the $(k - 1)$-equivalence classes of S (as described earlier) to find the k-equivalence classes of S.

Constructing the Quotient Automaton (2/2)

Stop subdividing when you observe that for some integer *K* the set of (*K* + 1)-equivalence classes equals the set of *K*-equivalence classes. At this point, conclude that the set of *K*-equivalence classes equals the set of *-equivalence classes.

3. Construct the quotient automaton $\overline{A}$ whose states are the *-equivalence classes of states of *A* and whose next-state function $\overline{N}$ is given by

$$\overline{N}([s], m) = [N(s, m)]$$

for each state *s* of $\overline{A}$ and each input symbol *m*.

Example 12.3.4 – *Constructing a Quotient Automaton*

Consider the automaton *A* of Examples 12.3.2 and 12.3.3. This automaton is shown again below for reference. Find the quotient automaton of *A.*

Example 12.3.4 – *Solution (1/4)*

According to Example 12.3.3, the *-equivalence classes of the states of A are

$\{s_0\}$, $\{s_1, s_4\}$, and $\{s_2, s_3\}$.

Hence the states of the quotient automaton $\overline{A}$ are

$[s_0] = \{s_0\}$, $[s_1] = \{s_1, s_4\} = [s_4]$, $[s_2] = \{s_2, s_3\} = [s_3]$.

The accepting states of A are s_2 and s_3, so the accepting state of $\overline{A}$ is $[s_2] = [s_3]$.

Example 12.3.4 – *Solution (2/4)* continued

The next-state function $\overline{N}$ of $\overline{A}$ is defined as follows: for all states [*s*] and input symbols *m* of $\overline{A}$,

$$\overline{N}([s], m) = [N(s, m)] = \text{the *-equivalence class of } N(s, m).$$

Thus

$$\overline{N}([s_0], 0) = [N(s_0, 0)] = \text{the *-equivalence class of } N(s_0, 0).$$

Now $N(s_0, 0) = s_0$, and so

$$\overline{N}([s_0], 0) = \text{the *-equivalence class of } s_0 = [s_0].$$

Example 12.3.4 – *Solution (3/4)* continued

Similarly,

$$\overline{N}([s_0], 1) = [N(s_0, 1)] = [s_1]$$

$$\overline{N}([s_1], 0) = [N(s_1, 0)] = [s_2]$$

$$\overline{N}([s_1], 1) = [N(s_1, 1)] = [s_4] = [s_1]$$

$$\overline{N}([s_2], 0) = [N(s_2, 0)] = [s_3] = [s_2]$$

$$\overline{N}([s_2], 1) = [N(s_2, 1)] = [s_4] = [s_1].$$

Example 12.3.4 – *Solution (4/4)* continued

The transition diagram for $\overline{A}$ is, therefore, as shown below.

By Theorem 12.3.3, this automaton accepts the same language as the original automaton.

Theorem 12.3.3

If A is a finite-state automaton, then the quotient automaton $\overline{A}$ accepts exactly the same languages as A. In other words, if $L(A)$ denotes the language accepted by A and $L(\overline{A})$ denotes the language accepted by $\overline{A}$, then

$$L(A) = L(\overline{A}).$$

Equivalent Automata

Equivalent Automata (1/3)

When a finite-state automaton is implemented by a circuit, output indicators may be attached to its states to indicate whether they are accepting or nonaccepting. For example, accepting states might produce an output of 1 and nonaccepting states an output of 0.

Then a finite-state automaton can be thought of as an input/output device whose input consists of strings and whose output consists of 0’s and 1’s.

Equivalent Automata (2/3)

We know that a circuit can be thought of as a black box that transforms combinations of input signals into output signals. Two circuits that produce identical output signals for each combination of input signals are called *equivalent.*

Similarly, a finite-state automaton can be regarded as a black box that processes input strings and produces output signals (indicating whether or not the strings are accepted).

Two finite-state automata are called *equivalent* if they produce identical output signals for each input string.

Equivalent Automata (3/3)

This implies that two finite-state automata are equivalent if, and only if, they accept the same language.

Definition

Let A and A' be finite-state automata with the same set of input symbols I. Let $L(A)$ denote the language accepted by A and $L(A')$ the language accepted by A'. Then A is said to be **equivalent** to A' if, and only if, $L(A) = L(A')$.

Example 12.3.5 – *Showing That Two Automata Are Equivalent*

Show that the automata *A* and *A′* that follow are equivalent.

The label 0, 1 on an arrow of a transition diagram means that for either input 0 or 1, the next-state of the automaton is the state to which the arrow points.

Example 12.3.5 – *Solution (1/7)*

For the automaton A: The 0-equivalence classes are

$\{s_0, s_1\}$ and $\{s_2, s_3\}$

since s_0 and s_1 are accepting states and s_2 and s_3 are nonaccepting states.

The 1-equivalence classes are

$\{s_0\}$, $\{s_1\}$, and $\{s_2, s_3\}$

since s_0 and s_1 are not 1-equivalent (because $N(s_0, 1) = s_1$, and $N(s_1, 1) = s_3$, and s_1 is not 0-equivalent to s_3) but s_2 and s_3 are 1-equivalent.

The 2-equivalence classes are

$\{s_0\}$, $\{s_1\}$, and $\{s_2, s_3\}$

since s_2 and s_3 are 1-equivalent.

Example 12.3.5 – *Solution (2/7)* continued

This discussion shows that the set of 1-equivalence classes equals the set of 2-equivalence classes, so by Theorem 12.3.2 this is equal to the set of *-equivalence classes.

Theorem 12.3.2

If A is a finite-state automaton, then for some integer, $K \geq 0$, the set of K-equivalence classes of states of A equals the set of $(K+1)$-equivalence classes of states of A, and for all such K these are both equal to the set of $*$-equivalence classes of states of A.

Example 12.3.5 – *Solution (3/7)* continued

Hence the *-equivalence classes are

$\{s_0\}$, $\{s_1\}$, and $\{s_2, s_3\}$.

For the automaton A′: By reasoning similar to that done previously, the 0-equivalence classes are

$\{s'_0, s'_2, s'_3\}$, and $\{s'_1\}$.

The 1-equivalence classes are

$\{s'_0, s'_3\}$, $\{s'_2\}$, and $\{s'_1\}$.

Example 12.3.5 – *Solution (4/7)* continued

The 2-equivalence classes are the same as the 1-equivalence classes, which are therefore equal to the *-equivalence classes. Thus the *-equivalence classes are

$$\{s'_0, s'_3\}, \{s'_2\}, \text{ and } \{s'_1\}.$$

To calculate the next-state functions for $\overline{A}$ and $\overline{A}'$, you repeatedly use the fact that in the quotient automaton, the next-state of [*s*] and *m* is the class of the next-state of *s* and *m.*

Example 12.3.5 – *Solution (5/7)* continued

For instance,

$$\overline{N}([s_1], 1) = [N(s_1, 1)] = [s_3] = [s_2]$$

$$\overline{N'}([s'_0], 0) = [N'(s'_0, 0)] = [s'_3] = [s'_0],$$

and

where N is the next-state function for A and N' is the next-state function for A'.

Example 12.3.5 – *Solution (6/7)* continued

The complete transition diagrams for the quotient automata $\overline{A}$ and $\overline{A'}$ are shown below.

As you can see, except for the labeling of the names of the states, $\overline{A}$ and $\overline{A'}$ are identical and hence accept the same language.

Example 12.3.5 – *Solution (7/7)* continued

Now by Theorem 12.3.3, each original automaton accepts the same language as its quotient automaton.

Theorem 12.3.3

If A is a finite-state automaton, then the quotient automaton $\overline{A}$ accepts exactly the same languages as A. In other words, if $L(A)$ denotes the language accepted by A and $L(\overline{A})$ denotes the language accepted by $\overline{A}$, then

$$L(A) = L(\overline{A}).$$

Thus A and A' accept the same language, and so they are equivalent.