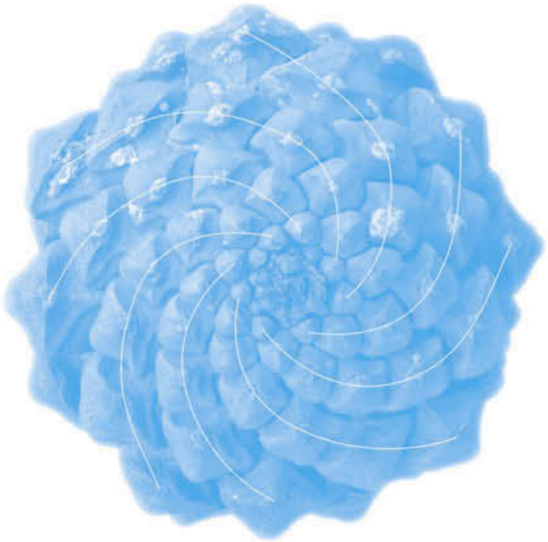
ESSENTIALS *of* DISCRETE MATHEMATICS

ESSENTIALS *of* DISCRETE MATHEMATICS



DAVID J. HUNTER
WESTMONT COLLEGE

JONES AND BARTLETT PUBLISHERS
Sudbury, Massachusetts
BOSTON TORONTO LONDON SINGAPORE

World Headquarters
Jones and Bartlett Publishers
40 Tall Pine Drive
Sudbury, MA 01776
978-443-5000
info@jbpub.com
www.jbpub.com

Jones and Bartlett Publishers
Canada
6339 Ormindale Way
Mississauga, Ontario L5V 1J2
Canada

Jones and Bartlett Publishers
International
Barb House, Barb Mews
London W6 7PA
United Kingdom

Production Credits
Acquisitions Editor: Timothy Anderson
Editorial Assistant: Melissa Potter
Production Director: Amy Rose
Production Manager: Jennifer Bagdigian
Production Editor: Katherine Macdonald
Senior Marketing Manager: Andrea DeFronzo
V.P., Manufacturing and Inventory Control: Therese Connell
Composition: Northeast Compositors, Inc.
Cover Design: Brian Moore/Kristin E. Ohlin
Interior Images: All photographs, excepting Figure 3.2, courtesy of David J. Hunter. Figure 3.2 courtesy of Pau Atela and Christophe Golé (www.math.smith.edu/phyllo). Technical illustrations by George Nichols.
Cover Image: Image courtesy of David J. Hunter.
Printing and Binding: Malloy, Inc.
Cover Printing: Malloy, Inc.

Library of Congress Cataloging-in-Publication Data
Hunter, David J. (David James), 1968–
Essentials of discrete mathematics / David J. Hunter.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-7637-4892-0 (cloth)
ISBN-10: 0-7637-4892-7 (cloth)
1. Computer science—Mathematics. I. Title.

QA76.9.M35H867 2008
511'.1—dc22

2008040794

6048
Printed in the United States of America
12 11 10 09 08 10 9 8 7 6 5 4 3 2 1

Contents

Preface

Introduction

Essentials of Discrete Mathematics is designed for first- or second-year Computer Science and Math majors in discrete mathematics courses. This text is also an excellent resource for students in other disciplines.

Unlike other textbooks on the market, *Essentials* presents the material in a manner that is suitable for a comprehensive and cohesive one-semester course. Students will learn to think mathematically and to find mathematical structures in almost everything.

Although most discrete mathematics texts are organized around mathematical objects, *Essentials* is organized around five types of mathematical thinking: logical, relational, recursive, quantitative, and analytical. To reinforce this approach, graphs are introduced early and referred to throughout the text, providing a richer context for examples and applications.

Applications are provided throughout the text, and the final chapter explores several uses of discrete mathematical thinking in a variety of disciplines. Case studies from biology, sociology, linguistics, economics, and music can be used as the basis for independent study or undergraduate research projects. Every chapter has its own set of exercises, which are designed to develop the skills of reading and writing proofs.

Synopsis of the Chapters

Chapter 1 introduces the reader to and emphasizes the importance of **Logical Thinking**. The chapter explores logic formally (symbolically), and then teaches the student to consider how logic is used in mathematical statements and arguments. The chapter begins with an introduction to formal logic, focusing on the importance of notation and symbols in mathematics, and then explains how formal logic can be applied. The chapter closes with a look at the different ways that proofs are constructed in mathematics.

As most mathematical problems contain different objects that are related to each other, Chapter 2 considers **Relational Thinking**. Finding the relationships among objects often is the first step in solving a mathematical problem. The mathematical structures of sets, relations, functions, and graphs describe these relationships, and thus this chapter focuses on exploring ways to use these structures to model mathematical relationships. Graph theory is introduced early and used throughout the chapter.

Chapter 3 describes **Recursive Thinking**. Many objects in nature have recursive structures: a branch of a tree looks like a smaller tree; an ocean swell has the same shape as the ripples formed by its splashes; an onion holds a smaller onion under its outer layer. Finding similar traits in mathematical objects unleashes a powerful tool. Chapter 3 begins by studying simple recurrence relations and then considers other recursive structures in a variety of contexts. Students also will cover recursive definitions, including how to write their own, and will extend the technique of induction to prove facts about recursively defined objects.

Chapter 4 engages the reader in **Quantitative Thinking**, as many problems in mathematics, computer science, and other disciplines involve counting the elements of a set of objects. The chapter examines the different tools used to count certain types of sets and teaches students to think about problems from a quantitative point of view. After exploring the different enumeration techniques, students will consider applications, including a first look at how to count operations in an algorithm. Chapter 4 also will practice the art of estimation, a valuable skill when precise enumeration is difficult.

Chapter 5 explores **Analytical Thinking**. Many applications of discrete mathematics are algorithms, and thus it is essential to be able to understand and analyze them. This chapter builds on the four foundations of thinking covered in the first four chapters, applying quantitative and relational thinking to the study of algorithm complexity, and then applying logical and recursive thinking to the study of program correctness. Finally, students will study mathematical ways to determine the accuracy and efficiency of algorithms.

The final chapter, **Thinking Through Applications**, examines different ways that discrete mathematical thinking can be applied: patterns in DNA, social networks, the structure of language, population models, and twelve-tone music.

Supplements

Instructor's Solutions Manual
PowerPoint Slide Presentation

Acknowledgments

I would like to thank the following reviewers for their valuable input and suggestions:

Peter B. Henderson; Butler University
Uwe Kaiser; Boise State University
Miklos Bona; University of Florida
Brian Hopkins; St. Peter's College
Frank Ritter; The Pennsylvania State University
Bill Marion; Valparaiso University

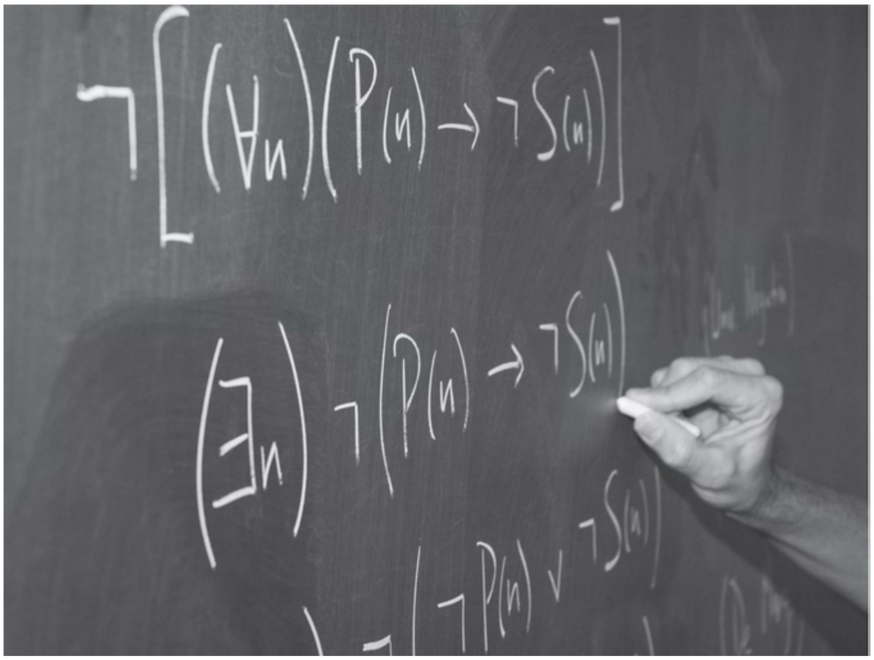
Also, I would like to express my deep appreciation to the team at Jones and Bartlett Publishers. In particular I would like to thank: my Acquisitions Editor, Tim Anderson; Amy Rose, Production Director; Jennifer Bagdigian, Production Manager; Katherine Macdonald, Production Editor; Melissa Elmore, Associate Production Editor; and Melissa Potter, Editorial Assistant.

David Hunter
Westmont College

Chapter 1

Logical Thinking

Mathematicians are in the business of stating things precisely. When you read a mathematical statement, you are meant to take every word seriously; good mathematical language conveys a clear, unambiguous message. In order to read



$\neg[(\forall n)(P(n) \to \neg S(n))]$
$(\exists n)\neg(P(n) \to \neg S(n))$
$\neg(\neg P(n) \vee \neg S(n))$

Figure 1.1 Symbols are an important part of the language of mathematics.

and write mathematics, you must practice the art of logical thinking. The goal of this chapter is to help you communicate mathematically by understanding the basics of logic.

A word of warning: mathematical logic can be difficult—especially the first time you see it. This chapter begins with the study of formal, or symbolic, logic, and then applies this study to the language of mathematics. Expect things to be a bit foggy at first, but eventually (we hope) the fog will clear. When it does, mathematical statements will start making more sense to you.

1.1 Formal Logic

Notation is an important part of mathematical language. Mathematicians' chalkboards are often filled with an assortment of strange characters and symbols; such a display can be intimidating to the novice, but there's a good reason for communicating this way. Often the act of reducing a problem to symbolic language helps us see what is really going on. Instead of operating in the fuzzy world of prose, we translate a problem to notation and then perform well-defined symbolic manipulations on that notation. This is the essence of the powerful tool called *formalism*. In this section, we explore how a formal approach to logic can help us avoid errors in reasoning.

A note on terminology: we'll use the word *formal* to describe a process that relies on manipulating notation. Often people use this word to mean "rigorous," but that's not our intention. A formal argument can be rigorous, but so can an argument that does not rely on symbols.

One nice feature of formalism is that it allows you to work without having to think about what all the symbols mean. In this sense, formal logic is really "logical *not*-thinking." Why is this an advantage? Formal calculations are less prone to error. You are already familiar with this phenomenon: much of the arithmetic you learned in school was formal. You have a well-defined symbolic algorithm for multiplying numbers using pencil and paper, and you can quite effectively multiply three-digit numbers without thinking much about what you are really doing. Of course, formalism is pointless if you don't know what you are doing; at the end of any formal calculation, it is important to be able to interpret the results.

1.1.1 Connectives and Propositions

In order to formalize logic, we need a system for translating statements into symbols. We'll start with a precise definition of *statement*.

Definition 1.1 A *statement* (also known as a *proposition*) is a declarative sentence that is either true or false, but not both.

The following are examples of statements:

- 7 is odd.
- $1 + 1 = 4$
- If it is raining, then the ground is wet.
- Our professor is from Mars.

Note that we don't need to be able to decide whether a statement is true or false in order for it to be a statement. Either our professor is from Mars, or our professor is not from Mars, though we may not be sure which is the case.

How can a declarative sentence fail to be a statement? There are two main ways. A declarative sentence may contain an unspecified term:

> x is even.

In this case, x is called a *free variable*. The truth of the sentence depends on the value of x, so if that value is not specified, we can't regard this sentence as a statement. A second type of declarative non-statement can happen when a sentence is *self-referential*:

> This sentence is false.

We can't decide whether or not the above sentence is true. If we say it is true, then it claims to be false; if we say it is false, then it appears to be true.

Often, a complicated statement consists of several simple statements joined together by words such as "and," "or," "if . . . then," etc. These connecting words are represented by the five *logical connectives* shown in Table 1.1. Logical connectives are useful for decomposing compound statements into simpler ones, because they highlight important logical properties of a statement.

In order to use a formal system for logic, we must be able to *translate* between a statement in English and its formal counterpart. We do this by assigning letters for simple statements, and then building expressions with connectives.

Name	Symbol
and	$\wedge$
or	$\vee$
not	$\neg$
implies (if . . . then)	$\rightarrow$
if and only if	$\leftrightarrow$

Table 1.1 The five logical connectives.

Example 1.1 If p is the statement "you are wearing shoes" and q is the statement "you can't cut your toenails," then

$$p \to q$$

represents the statement, "If you are wearing shoes, then you can't cut your toenails." We may choose to express this statement differently in English: "You can't cut your toenails if you are wearing shoes," or "Wearing shoes makes it impossible to cut your toenails." The statement $\neg q$ translates literally as "It is not the case that you can't cut your toenails." Of course, in English, we would prefer to say simply, "You can cut your toenails," but this involves using logic, as we will see in the next section.

1.1.2 Truth Tables

We haven't finished setting up our formal system for logic because we haven't been specific about the meaning of the logical connectives. Of course, the names of each connective suggest how they should be used, but in order to make statements with mathematical precision, we need to know exactly what each connective means.

Defining the meaning of a mathematical symbol is harder than it might seem. Even the + symbol from ordinary arithmetic is problematic. Although we all have an intuitive understanding of addition—it describes how to combine two quantities—it is hard to express this concept in words without appealing to our intuition. What does "combine" mean, exactly? What are "quantities," really?

One simple, but obviously impractical, way to define the + sign would be to list all possible addition problems, as in Table 1.2. Of course, such a table could never end, but it would, in theory, give us a precise definition of the + sign.

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	2
2	1	3
1	2	3
$\vdots$	$\vdots$	$\vdots$

Table 1.2 Defining the + sign by listing all possible addition problems would require an infinite table.

The situation in logic is easier to handle. Any statement has two possible values: true (T) or false (F). So when we use variables such as p or q for statements in logic, we can think of them as unknowns that can take one of only two values: T or F. This makes it possible to define the meaning of each connective using tables; instead of infinitely many values for numbers x and y, we have only two choices for each variable p and q.

We will now stipulate the meaning of each logical connective by listing the T/F values for every possible case. The simplest example is the "not" connective, $\neg$. If p is true, then $\neg p$ should be false, and *vice versa.*

p	$\neg p$
T	F
F	T

This table of values is called a *truth table*; it defines the T/F values for the connective.

The "and" and "or" connectives are defined by the following truth tables. Since we have two variables, and each can be either T or F, we need four cases.

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

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

The definition of the "and" connective $\wedge$ is what you would expect: in order for $p \wedge q$ to be true, p must be true *and* q must be true. The "or" connective $\vee$ is a little less obvious. Notice that our definition stipulates that $p \vee q$ is true whenever p is true, or q is true, or both are true. This can be different from the way "or" is used in everyday speech. When you are offered "soup or salad" in a restaurant, your server isn't expecting you to say "both."

The "if and only if" connective says that two statements have exactly the same truth values. Thus, its truth table is as follows.

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

Sometimes authors will write "iff" as an abbreviation for "if and only if."

The "implies" connective has the least intuitive definition.

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

To understand the motivation for this definition, recall Example 1.1. In order to demonstrate that the statement

> $p \to q =$ "If you are wearing shoes, then you can't cut your toenails."

is false, you would have to be able to cut your toenails while wearing shoes. In any other situation, you would have to concede that the statement is not false (and if a statement is not false, it must be true). If you are not wearing shoes, then maybe you can cut your toenails or maybe you can't, for some other reason. This doesn't contradict the statement $p \to q$.

Put another way, if you live in a world without shoes, then the statement is *vacuously* true; since you can never actually wear shoes, it isn't false to say that"*If* you are wearing shoes," then anything is possible. This explains the last two lines of the truth table; if p is false, then $p \to q$ is true, no matter what q is.

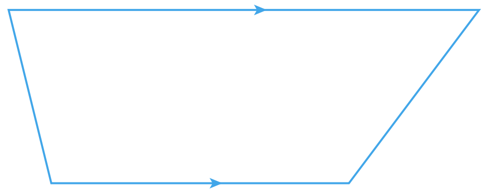
1.1.3 Logical Equivalences

Definition 1.2 Two statements are *logically equivalent* if they have the same T/F values for all cases, that is, if they have the same truth tables.

There are some logical equivalences that come up often in mathematics, and also in life in general.

Example 1.2 Consider the following theorem from high school geometry.

> If a quadrilateral has a pair of parallel sides, then it has a pair of supplementary angles.[1]



1. Recall that two angles are supplementary if their angle measures sum to 180°.

This theorem is of the form $p \to q$, where p is the statement that the quadrilateral has a pair of parallel sides, and q is the statement that the quadrilateral has a pair of supplementary angles.

We can state a different theorem, represented by $\neg q \to \neg p$.

> If a quadrilateral does not have a pair of supplementary angles, then it does not have a pair of parallel sides.

We know that this second theorem is logically equivalent to the first because the formal statement $p \to q$ is logically equivalent to the formal statement $\neg q \to \neg p$, as the following truth table shows.

p	q	$p \to q$	$\neg q$	$\neg p$	$\neg q \to \neg p$
T	T	T	F	F	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Notice that the column for $p \to q$ matches the column for $\neg q \to \neg p$. Since the first theorem is a true theorem from geometry, so is the second.

Now consider a different variation on this theorem.

> If a quadrilateral has a pair of supplementary angles, then it has a pair of parallel sides.

This statement is of the form $q \to p$. But the following truth table shows that $q \to p$ is *not* logically equivalent to $p \to q$, because the T/F values are different in the second and third rows.

p	q	$q \to p$
T	T	T
T	F	T
F	T	F
F	F	T

In fact, this last statement is not true, in general, in geometry. (Can you draw an example of a quadrilateral for which it fails to be true?)

The statement $\neg q \to \neg p$ is called the *contrapositive* of $p \to q$, and the statement $q \to p$ is called the *converse*. The truth tables above prove that, for any statement s, the contrapositive of s is logically equivalent to s, while the converse of s may not be.

There are lots of situations where assuming the converse can cause trouble. For example, suppose that the following statement is true.

> If a company is not participating in illegal accounting practices, then an audit will turn up no evidence of wrongdoing.

It is certainly reasonable to assume this, since there couldn't be evidence of wrongdoing if no such wrongdoing exists. However, the converse is probably not true:

> If an audit turns up no evidence of wrongdoing, then the company is not participating in illegal accounting practices.

After all, it is possible that the auditors missed something.

At this point you might object that formal logic seems like a lot of trouble to go through just to verify deductions like this last example. This sort of thing is just common sense, right? Well, maybe. But something that appears obvious to you may not be obvious to someone else. Furthermore, our system of formal logic can deal with more complicated situations, where our common sense might fail us. The solution to the next example uses formal logic. Before you look at this solution, try to solve the problem using "common sense." Although the formal approach takes a little time, it resolves any doubt you might have about your own reasoning process.

Example 1.3 If Aaron is late then Bill is late, and, if both Aaron and Bill are late, then class is boring. Suppose that class is not boring. What can you conclude about Aaron?

Solution: Let's begin by translating the first sentence into the symbols of logic, using the following statements.

$p =$ Aaron is late.
$q =$ Bill is late.
$r =$ Class is boring.

Let S be the statement "If Aaron is late then Bill is late, and, if both Aaron and Bill are late, then class is boring." In symbols, S translates to the following.

$$S = (p \to q) \land [(p \land q) \to r]$$

Now let's construct a truth table for S. We do this by constructing truth tables for the different parts of S, starting inside the parentheses and working our way out.

Row #	p	q	r	$p \to q$	$p \wedge q$	$(p \wedge q) \to r$	S
1.	T	T	T	T	T	T	T
2.	T	T	F	T	T	F	F
3.	T	F	T	F	F	T	F
4.	T	F	F	F	F	T	F
5.	F	T	T	T	F	T	T
6.	F	T	F	T	F	T	T
7.	F	F	T	T	F	T	T
8.	F	F	F	T	F	T	T

You should check that the last column is the result of "and-ing" the column for $p \to q$ with the column for $(p \wedge q) \to r$.

We are interested in the possible values of p. It is given that S is true, so we can eliminate rows 2, 3, and 4, the rows where S is false. If we further assume that class is not boring, we can also eliminate the rows where r is true, namely the odd-numbered rows. The rows that remain are the only possible T/F values for p, q, and r: rows 6 and 8. In both of these rows, p is false. In other words, Aaron is not late. ♢

Exercises 1.1

1. Let the following statements be given.

$$p = \text{"There is water in the cylinders."}$$
$$q = \text{"The head gasket is blown."}$$
$$r = \text{"The car will start."}$$

   (a) Translate the following statement into symbols of formal logic.

   If the head gasket is blown and there's water in the cylinders, then the car won't start.

   (b) Translate the following formal statement into everyday English.

$$r \to \neg(q \vee p)$$

2. Let the following statements be given.

$$p = \text{"You are in Seoul."}$$
$$q = \text{"You are in Kwangju."}$$
$$r = \text{"You are in South Korea."}$$

(a) Translate the following statement into symbols of formal logic.

If you are not in South Korea, then you are not in Seoul or Kwangju.

(b) Translate the following formal statement into everyday English.

$$q \to (r \wedge \neg p)$$

3. Let the following statements be given.

$$\begin{aligned} p &= \text{“You can vote.”} \\ q &= \text{“You are under 18 years old.”} \\ r &= \text{“You are from Mars.”} \end{aligned}$$

(a) Translate the following statement into symbols of formal logic.

You can't vote if you are under 18 years old or you are from Mars.

(b) Give the converse of this statement in the symbols of formal logic.

(c) Give the converse in English.

4. Let s be the following statement.

If you are studying hard, then you are staying up late at night.

(a) Give the converse of s.

(b) Give the contrapositive of s.

5. Let s be the following statement.

If it is raining, then the ground is wet.

(a) Give the converse of s.

(b) Give the contrapositive of s.

6. Give an example of a quadrilateral that shows that the *converse* of the following statement is false.

If a quadrilateral has a pair of parallel sides, then it has a pair of supplementary angles.

7. We say that two ordered pairs (a, b) and (c, d) are *equal* when $a = c$ and $b = d$. Let s be the following statement.

If $(a, b) = (c, d)$, then $a = c$.

(a) Is this statement true?

(b) Write down the converse of s.

(c) Is the converse of s true? Explain.

8. Give an example of a true if–then statement whose converse is also true.

9. Show that $p \leftrightarrow q$ is logically equivalent to $(p \to q) \wedge (q \to p)$ using truth tables.

10. Use truth tables to establish the following equivalences.

    (a) Show that $\neg(p \vee q)$ is logically equivalent to $\neg p \wedge \neg q$.

    (b) Show that $\neg(p \wedge q)$ is logically equivalent to $\neg p \vee \neg q$.

    These equivalences are known as *De Morgan's laws*, after the 19th century logician Augustus De Morgan.

11. Use truth tables to show that $(a \vee b) \wedge (\neg(a \wedge b))$ is logically equivalent to $a \leftrightarrow \neg b$. (This arrangement of T/F values is sometimes called the *exclusive or* of a and b.)

12. Use a truth table to prove that the statement

$$[(p \vee q) \wedge (\neg p)] \to q$$

    is always true, no matter what p and q are.

13. Let the following statements be given.

    $p =$ Andy is hungry.
    $q =$ The refrigerator is empty.
    $r =$ Andy is mad.

    (a) Use connectives to translate the following statement into formal logic.

        If Andy is hungry and the refrigerator is empty, then Andy is mad.

    (b) Construct a truth table for the statement in part (a)

    (c) Suppose that the statement given in part (a) is true, and suppose also that Andy is not mad and the refrigerator is empty. Is Andy hungry? Explain how to justify your answer using the truth table.

14. Use truth tables to prove the following *distributive properties* for propositional logic.

    (a) $p \wedge (q \vee r)$ is logically equivalent to $(p \wedge q) \vee (p \wedge r)$.

    (b) $p \vee (q \wedge r)$ is logically equivalent to $(p \vee q) \wedge (p \vee r)$.

15. Use truth tables to prove the *associative properties* for propositional logic.

    (a) $p \vee (q \vee r)$ is logically equivalent to $(p \vee q) \vee r$.

    (b) $p \wedge (q \wedge r)$ is logically equivalent to $(p \wedge q) \wedge r$.

16. Mathematicians say that "statement P is *stronger* than statement Q" if Q is true whenever P is true, but not conversely. (In other words, "P is stronger than Q" means that $P \to Q$ is always true, but $Q \to P$ is not true, in general.) Use truth tables to show the following.

    (a) $a \wedge b$ is stronger than a.

    (b) a is stronger than $a \vee b$.

    (c) $a \wedge b$ is stronger than $a \vee b$.

    (d) b is stronger than $a \to b$.

17. Suppose $\mathcal{Q}$ is a quadrilateral. Which statement is stronger?

    - $\mathcal{Q}$ is a square.
    - $\mathcal{Q}$ is a rectangle.

    Explain.

18. Which statement is stronger?

    - Manchester United is the best football team in England.
    - Manchester United is the best football team in Europe.

    Explain.

19. Which statement is stronger?

    - n is divisible by 3.
    - n is divisible by 12.

    Explain.

20. Mathematicians say that "Statement P is a *sufficient condition* for statement Q" if $P \to Q$ is true. In other words, in order to know that Q is true, it is sufficient to know that P is true. Let x be an integer. Give a sufficient condition on x for $x/2$ to be an even integer.

21. Mathematicians say that "Statement P is a *necessary condition* for statement Q" if $Q \to P$ is true. In other words, in order for Q to be true, it is necessary that P must be true. Let $n \geq 1$ be a natural number. Give a necessary but not sufficient condition on n for $n + 2$ to be prime.

22. Write the statement "P is necessary and sufficient for Q" in the symbols of formal logic, using as few connectives as possible.

23. Often a complicated expression in formal logic can be simplified. For example, consider the statement $S = (p \wedge q) \vee (p \wedge \neg q)$.

    (a) Construct a truth table for S.

    (b) Find a simpler expression that is logically equivalent to S.

24. The NAND connective $\uparrow$ is defined by the following truth table.

p	q	$p \uparrow q$
T	T	F
T	F	T
F	T	T
F	F	T

    Use truth tables to show that $p \uparrow q$ is logically equivalent to $\neg(p \wedge q)$. (This explains the name NAND: Not AND.)

25. The NAND connective is important because it is easy to build an electronic circuit that computes the NAND of two signals (see Figure 1.2). Such a circuit is called a *logic gate*. Moreover, it is possible to build logic gates for the other logical connectives entirely out of NAND gates. Prove this fact by proving the following equivalences, using truth tables.

    (a) $(p \uparrow q) \uparrow (p \uparrow q)$ is logically equivalent to $p \wedge q$.

    (b) $(p \uparrow p) \uparrow (q \uparrow q)$ is logically equivalent to $p \vee q$.

    (c) $p \uparrow (q \uparrow q)$ is logically equivalent to $p \to q$.

26. Write $\neg p$ in terms of p and $\uparrow$.

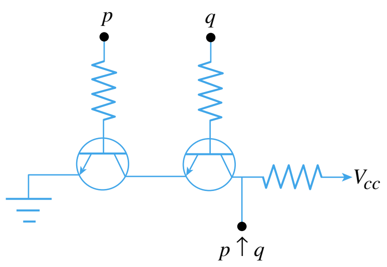


Figure 1.2 A NAND gate can be built with just two transistors.

27. A technician suspects that one or more of the processors in a distributed system is not working properly. The processors, A, B, and C, are all capable of reporting information about the status (working or not working) of the processors in the system. The technician is unsure whether a processor is really not working, or whether the problem is in the status reporting routines in one or more of the processors. After polling each processor, the technician receives the following status reports.

   - Processor A reports that Processor B is not working and processor C is working.
   - Processor B reports that A is working if and only if B is working.
   - Processor C reports that at least one of the other two processors is not working.

   Help the technician by answering the following questions.

   (a) Let a = "A is working," b = "B is working," and c = "C is working." Write the three status reports in terms of a, b, and c, using the symbols of formal logic.

   (b) Complete the following truth table.

a	b	c	A's report	B's report	C's report
T	T	T			
T	T	F			
T	F	T			
T	F	F			
F	T	T			
F	T	F			
F	F	T			
F	F	F			

   (c) Assuming that all of the status reports are true, which processor(s) is/are working?

   (d) Assuming that all of the processors are working, which status report(s) is/are false?

   (e) Assuming that a processor's status report is true if and only if the processor is working, what is the status of each processor?

1.2 Propositional Logic

After working through the exercises of the previous section, you may have noticed a serious limitation of the truth table approach. Each time you add a new statement to a truth table, you must double the number of rows. This makes truth table analysis unwieldy for all but the simplest examples.

In this section we will develop a system of rules for manipulating formulas in symbolic logic. This system, called the *propositional calculus*, will allow us to make logical deductions formally. There are at least three reasons for doing this.

1. These formal methods are useful for analyzing complex logical problems, especially where truth tables are impractical.

2. The derivation rules we will study are commonly used in mathematical discourse.

3. The system of derivation rules and proof sequences is a simple example of mathematical proof.

Of these three, the last is the most important. The mechanical process of writing proof sequences in propositional calculus will prepare us for writing more complicated proofs in other areas of mathematics.

1.2.1 Tautologies and Contradictions

There are some statements in formal logic that are always true, no matter what the T/F values of the component statements are. For example, the truth table for $(p \wedge q) \to p$ is as follows.

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

Such a statement is called a *tautology*, and we write

$$(p \wedge q) \Rightarrow p$$

to indicate this fact. The notation $A \Rightarrow B$ means that the statement $A \to B$ is true in all cases; in other words, the truth table for $A \to B$ is all T's. Similarly, the $\Leftrightarrow$ symbol denotes a tautology containing the $\leftrightarrow$ connective.

Example 1.4 In Exercise 1.1.10 you proved the following tautologies.

(a) $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$.

(b) $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$.

When a tautology is of the form $(C \wedge D) \Rightarrow E$, we often prefer to write

$$\left.\begin{array}{r} C \\ D \end{array}\right\} \Rightarrow E$$

instead. This notation highlights the fact that if you know both C and D, then you can conclude E. The use of the $\wedge$ connective is implicit.

Example 1.5 Use a truth table to prove the following.

$$\left.\begin{array}{r} p \\ p \to q \end{array}\right\} \Rightarrow q$$

Solution: Let S be the statement $[p \wedge (p \to q)] \to q$. We construct our truth table by building up the parts of S, working from inside the parentheses outward.

p	q	$p \to q$	$p \wedge (p \to q)$	S
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Since the column for S is all T's, this proves that S is a tautology. ♢

The tautology in Example 1.5 is known as *modus ponens*, which is Latin for "affirmative mode." This concept goes back at least as far as the Stoic philosophers of ancient Greece, who stated it as follows.

> If the first, then the second;
> but the first;
> therefore the second.

In the exercises, you will have the opportunity to prove a related result called *modus tollens* ("denial mode"). In the symbols of logic, this tautology is as follows.

$$\left.\begin{array}{r} \neg q \\ p \to q \end{array}\right\} \Rightarrow \neg p$$

There are also statements in formal logic that are never true. A statement whose truth table contains all F's is called a *contradiction*.

Example 1.6 Use a truth table to show that $p \wedge \neg p$ is a contradiction.

Solution:

p	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

In other words, a statement and its negation can never both be true. ◇

A statement in propositional logic that is neither a tautology nor a contradiction is called a *contingency*. A contingency has both T's and F's in its truth table, so its truth is "contingent" on the T/F values of its component statements. For example, $p \wedge q$, $p \vee q$, and $p \to q$ are all contingencies.

1.2.2 Derivation Rules

Tautologies are important because they show how one statement may be logically deduced from another. For example, suppose we know that the following statements are true.

> Our professor does not own a spaceship.
> If our professor is from Mars, then our professor owns a spaceship.

We can apply the *modus tollens* tautology to deduce that "Our professor is not from Mars." This is a valid argument, or *derivation*, that allows us to conclude this last statement given the first two.

Every tautology can be used as a rule to justify deriving a new statement from an old one. There are two types of derivation rules: equivalence rules and inference rules. Equivalence rules describe logical equivalences, while inference rules describe when a weaker statement can be deduced from a stronger statement. The equivalence rules given in Table 1.3 could all be checked using truth tables. If A and B are statements (possibly comprised of many other statements joined by connectives), then the tautology $A \Leftrightarrow B$ is another way of saying that A and B are logically equivalent.[2]

An equivalence rule of the form $A \Leftrightarrow B$ can do three things:

1. Given A, deduce B.

2. Given B, deduce A.

2. A word on notation: We typically use $p, q, r, \ldots$ to stand for simple statements, and we use $A, B, C, \ldots$ to denote statements that are (possibly) made up of simple statements and logical connectives. This convention, however, is purely expository, and doesn't signify any difference in meaning.

3. Given a statement containing statement A, deduce the same statement, but with statement A replaced by statement B.

The third option is a form of *substitution*. For example, given the following statement,

> If Micah is not sick and Micah is not tired, then Micah can play.

we can deduce the following using De Morgan's laws.

> If it is not the case that Micah is sick or tired, then Micah can play.

In addition to equivalence rules, there are also inference rules for propositional logic. Unlike equivalence rules, inference rules only work in one direction. An inference rule of the form $A \Rightarrow B$ allows you to do only one thing:

1. Given A, deduce B.

In other words, you can conclude a weaker statement, B, if you have already established a stronger statement, A. For example, *modus tollens* is an inference rule: the weaker statement

> $B =$ "Our professor is not from Mars."

follows from the stronger statement

> $A =$ "Our professor does not own a spaceship, and if our professor is from Mars, then our professor owns a spaceship."

If A is true, then B must be true, but not *vice versa*. (Our professor might own a spaceship and be from Jupiter, for instance.) Table 1.4 lists some useful inference rules, all of which can be verified using truth tables.

Equivalence	Name
$p \Leftrightarrow \neg\neg p$	double negation
$p \to q \Leftrightarrow \neg p \vee q$	implication
$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$ $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$	De Morgan's laws
$p \vee q \Leftrightarrow q \vee p$ $p \wedge q \Leftrightarrow q \wedge p$	commutativity
$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$ $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$	associativity

Table 1.3 Equivalence Rules

Inference	Name
$\left.\begin{array}{r} p \\ q \end{array}\right\} \Rightarrow p \wedge q$	conjunction
$\left.\begin{array}{r} p \\ p \rightarrow q \end{array}\right\} \Rightarrow q$	*modus ponens*
$\left.\begin{array}{r} \neg q \\ p \rightarrow q \end{array}\right\} \Rightarrow \neg p$	*modus tollens*
$p \wedge q \Rightarrow p$	simplification
$p \Rightarrow p \vee q$	addition

Table 1.4 Inference Rules

1.2.3 Proof Sequences

We now have enough tools to derive some new tautologies from old ones. A *proof sequence* is a sequence of statements and reasons to justify an assertion of the form $A \Rightarrow C$. The first statement, A, is given.[3] The proof sequence can then list statements $B_1, B_2, B_3, \ldots,$ etc., as long as each new statement can be derived from a previous statement (or statements) using some derivation rule. Of course, this sequence of statements must culminate in C, the statement we are trying to prove, given A.

Example 1.7 Write a proof sequence for the assertion

$$\left.\begin{array}{r} p \\ p \rightarrow q \\ q \rightarrow r \end{array}\right\} \Rightarrow r.$$

Solution:

Statements	Reasons
1. p	given
2. $p \rightarrow q$	given
3. $q \rightarrow r$	given
4. q	*modus ponens*, 1, 2
5. r	*modus ponens*, 4, 3

◇

3. Often there are several given statements.

Every time we prove something, we get a new inference rule. The rules in Table 1.4 are enough to get us started, but we should feel free to use proven assertions in future proofs. For example, the assertion proved in Example 1.7 illustrates the *transitive* property of the $\to$ connective.

Another thing to notice about Example 1.7 is that it was pretty easy—we just had to apply *modus ponens* twice. Compare this to the truth table approach: the truth table for

$$[p \wedge (p \to q) \wedge (q \to r)] \to r$$

would consist of eight rows and several columns. Truth tables are easier to do, but they can be much more tedious.

Proof sequences should remind you of the types of proofs you did in high school geometry. The rules are simple: start with the given, see what you can deduce, end with what you are trying to prove. Here's a harder example.

Example 1.8 Prove:

$$\left.\begin{array}{c} p \vee q \\ \neg p \end{array}\right\} \Rightarrow q$$

Solution:

Statements	Reasons
1. $p \vee q$	given
2. $\neg p$	given
3. $\neg(\neg p) \vee q$	double negation, 1
4. $\neg p \to q$	implication, 3
5. q	*modus ponens*, 4, 2

◊

Notice that in step 3 of this proof, we used one of the equivalence rules (double negation) to make a substitution in the formula. This is allowed: since $\neg(\neg p)$ is logically equivalent to p, it can take the place of p in any formula.

1.2.4 Forward–Backward

If you are having trouble coming up with a proof sequence, try the "forward–backward" approach: consider statements that are one step forward from the given, and also statements that are one step backward from the statement you are trying to prove. Repeat this process, forging a path of deductions forward from the given and backward from the final statement. If all goes well, you will discover a way to make these paths meet in the middle. The next example illustrates this technique.

Example 1.9 In Section 1.1, we used truth tables to show that a statement is logically equivalent to its contrapositive. In this example we will construct a proof sequence for one direction of this logical equivalence:

$$p \to q \quad \Rightarrow \quad \neg q \to \neg p$$

Solution: We apply the forward–backward approach. The only given statement is $p \to q$, so we search our derivation rules for something that follows from this statement. The only candidate is $\neg p \vee q$, by the implication rule, so we tentatively use this as the second step of the proof sequence. Now we consider the statement we are trying to prove, $\neg q \to \neg p$, and we look backward for a statement from which this statement follows. Since implication is an equivalence rule, we can also use it to move backward to the statement $\neg(\neg q) \vee \neg p$, which we propose as the second-to-last statement of our proof. By moving forward one step from the given and backward one step from the goal, we have reduced the task of proving

$$p \to q \quad \Rightarrow \quad \neg q \to \neg p$$

to the (hopefully) simpler task of proving

$$\neg p \vee q \quad \Rightarrow \quad \neg(\neg q) \vee \neg p.$$

Now it is fairly easy to see how to finish the proof: we can switch the $\vee$ statement around using commutativity and simplify using double negation. We can now write down the proof sequence.

Statements	Reasons
1. $p \to q$	given
2. $\neg p \vee q$	implication
3. $q \vee \neg p$	commutativity
4. $\neg(\neg q) \vee \neg p$	double negation
5. $\neg q \to \neg p$	implication

We used the forward–backward approach to move forward from step 1 to step 2, and again to move backward from step 5 to step 4. Then we connected step 2 to step 4 with a simple proof sequence. ♢

You may have noticed that in Section 1.1, we proved the stronger statement

$$p \to q \quad \Leftrightarrow \quad \neg q \to \neg p$$

using truth tables; the above example only proves the "$\Rightarrow$" direction of this equivalence. To prove the other direction, we need another proof sequence. However, in this case, this other proof sequence is easy to write down, because all of the derivation rules we used were reversible. Implication, commutativity, and double negation are all equivalence rules, so we could write down a new proof sequence with the order of the steps reversed, and we would have a valid proof of the "$\Leftarrow$" direction.

Exercises 1.2

1. Use truth tables to establish the *modus tollens* tautology:

$$\left.\begin{array}{r} \neg q \\ p \to q \end{array}\right\} \Rightarrow \neg p$$

2. Fill in the reasons in the following proof sequence. Make sure you indicate which step(s) each derivation rule refers to.

Statements	Reasons
1. $p \wedge (q \vee r)$	given
2. $\neg(p \wedge q)$	given
3. $\neg p \vee \neg q$	
4. $\neg q \vee \neg p$	
5. $q \to \neg p$	
6. p	
7. $\neg(\neg p)$	
8. $\neg q$	
9. $q \vee r$	
10. $r \vee q$	
11. $\neg(\neg r) \vee q$	
12. $\neg r \to q$	
13. $\neg(\neg r)$	
14. r	
15. $p \wedge r$	

3. Justify each conclusion with a derivation rule.

   (a) Someone who is artistic must also be creative. Joe is not creative. Therefore, Joe is not artistic.

   (b) Lingli is both athletic and intelligent. Therefore, Lingli is athletic.

(c) If you are 18 years old, then you may vote. Monique is 18 years old. Therefore, Monique may vote.

(d) Marianne has never been north of Saskatoon or south of Santo Domingo. In other words, she has never been north of Saskatoon and she has never been south of Santo Domingo.

4. Let x and y be integers. Given the statement

   $x > y$ or x is odd.

   what statement follows by the implication rule?

5. Let $\mathcal{Q}$ be a quadrilateral. Given the statements

   If $\mathcal{Q}$ is a rhombus, then $\mathcal{Q}$ is a parallelogram.
   $\mathcal{Q}$ is not a parallelogram.

   what statement follows by *modus tollens*?

6. Let x and y be numbers. Simplify the following statement using De Morgan's laws and double negation.

   It is not the case that x is not greater than 3 and y is not found.

7. Write a statement that follows from the statement

   It is sunny and warm today.

   by the simplification rule.

8. Write a statement that follows from the statement

   This soup tastes funny.

   by the addition rule.

9. Recall Exercise 1.1.27. Suppose that all of the following status reports are correct:

   - Processor B is not working and processor C is working.
   - Processor A is working if and only if processor B is working.
   - At least one of the two processors A, B is not working.

   Let $a =$ "A is working," $b =$ "B is working," and $c =$ "C is working."

   (a) If you haven't already done so, write each status report in terms of a, b, and c, using the symbols of formal logic.

   (b) How would you justify the conclusion that B is not working? (In other words, given the statements in part (a), which derivation rule allows you to conclude $\neg b$?)

(c) How would you justify the conclusion that C is working?

(d) Write a proof sequence to conclude that A is not working. (In other words, given the statements in part (a), write a proof sequence to conclude $\neg a$.)

10. Write a proof sequence for the following assertion. Justify each step.

$$\left.\begin{array}{r} p \to \neg q \\ r \to (p \wedge q) \end{array}\right\} \Rightarrow \neg r$$

11. Write a proof sequence for the following assertion. Justify each step.

$$\left.\begin{array}{c} p \\ p \to r \\ q \to \neg r \end{array}\right\} \Rightarrow \neg q$$

12. Write a proof sequence for the following assertion. Justify each step.

$$\left.\begin{array}{c} p \to q \\ p \wedge r \end{array}\right\} \Rightarrow q \wedge r$$

13. Write a proof sequence for the following assertion. Justify one of the steps in your proof using the result of Example 1.8.

$$\left.\begin{array}{c} \neg(a \wedge \neg b) \\ \neg b \end{array}\right\} \Rightarrow \neg a$$

14. Write a proof sequence to establish that $p \Leftrightarrow p \wedge p$ is a tautology.

15. Write a proof sequence to establish that $p \Leftrightarrow p \vee p$ is a tautology. (Hint: Use De Morgan's laws and Exercise 14.)

16. Write a proof sequence for the following assertion. Justify each step.

$$\neg(\neg p \to q) \vee (\neg p \wedge \neg q) \quad \Rightarrow \quad \neg p \wedge \neg q$$

17. Write a proof sequence for the following assertion. Justify each step.

$$(p \vee q) \vee (p \vee r) \quad \Rightarrow \quad \neg r \to (p \vee q)$$

18. Consider the following assertion.

$$\neg(\neg p \vee q) \quad \Rightarrow \quad p \vee q$$

(a) Find a statement that is one step forward from the given.

(b) Find a statement that is one step backward from the goal. (Use the addition rule—in reverse—to find a statement from which the goal will follow.)

(c) Give a proof sequence for the assertion.

(d) Is your proof reversible? Why or why not?

19. Use a truth table to show that

$$\left.\begin{array}{c} p \to q \\ \neg p \end{array}\right\} \overset{?}{\Rightarrow} \neg q$$

is not a tautology. (This example shows that substitution isn't valid for inference rules, in general. Substituting the weaker statement, q, for the stronger statement, p, in the expression "$\neg p$" doesn't work.)

20. (a) Fill in the reasons in the following proof sequence. Make sure you indicate which step(s) each derivation rule refers to.

Statements	Reasons
1. $p \to (q \to r)$	given
2. $\neg p \vee (q \to r)$	
3. $\neg p \vee (\neg q \vee r)$	
4. $(\neg p \vee \neg q) \vee r$	
5. $\neg(p \wedge q) \vee r$	
6. $(p \wedge q) \to r$	

(b) Explain why the proof in part (a) is reversible.

(c) The proof in part (a) (along with its reverse) establishes the following tautology:

$$p \to (q \to r) \;\Leftrightarrow\; (p \wedge q) \to r$$

Therefore, to prove an assertion of the form $A \Rightarrow B \to C$, it is sufficient to prove

$$\left.\begin{array}{c} A \\ B \end{array}\right\} \Rightarrow C$$

instead. Use this fact to rewrite the tautology

$$p \wedge (q \to r) \;\Rightarrow\; q \to (p \wedge r)$$

as a tautology of the form

$$\left.\begin{array}{c} A \\ B \end{array}\right\} \Rightarrow C,$$

where C does not contain the $\rightarrow$ connective. (The process of rewriting a tautology this way is called the *deduction method*.)

(d) Give a proof sequence for the rewritten tautology in part (c).

21. This exercise will lead you through a proof of the *distributive property* of $\wedge$ over $\vee$. We will prove:

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

(a) The above assertion is the same as the following:

$$p \wedge (q \vee r) \Rightarrow \neg(p \wedge q) \rightarrow (p \wedge r).$$

Why?

(b) Use the deduction method to rewrite the tautology from part (b).

(c) Prove your rewritten tautology.

22. Use a truth table to show that $(a \rightarrow b) \wedge (a \wedge \neg b)$ is a contradiction.

23. Is $a \rightarrow \neg a$ a contradiction? Why or why not?

1.3 Predicate Logic

When we defined statements, we said that a sentence of the form

$$x \text{ is even}$$

is not a statement, because its T/F value depends on x. Mathematical writing, however, almost always deals with sentences of this type; we often express mathematical ideas in terms of some unknown variable. This section explains how to extend our formal system of logic to deal with this situation.

1.3.1 Predicates

Definition 1.3 A *predicate* is a declarative sentence whose T/F value depends on one or more variables. In other words, a predicate is a declarative sentence with variables, and after those variables have been given specific values, the sentence becomes a statement.

We use function notation to denote predicates. For example,

$$\begin{aligned} P(x) &= \text{“}x \text{ is even,” and} \\ Q(x, y) &= \text{“}x \text{ is heavier than } y\text{”} \end{aligned}$$

are predicates. The statement $P(8)$ is true, while the statement

$$Q(\text{feather}, \text{brick})$$

is false.

Implicit in a predicate is the *domain* (or *universe*) of values that the variable(s) can take. For $P(x)$, the domain could be the integers; for $Q(x, y)$, the domain could be some collection of physical objects. We will usually state the domain along with the predicate, unless it is clear from the context.

Equations are predicates. For example, if $E(x)$ stands for the equation

$$x^2 - x - 6 = 0,$$

then $E(3)$ is true and $E(4)$ is false. We regard equations as declarative sentences, where the $=$ sign plays the role of a verb.

1.3.2 Quantifiers

By themselves, predicates aren't statements because they contain free variables. We can make them into statements by plugging in specific values of the domain, but often we would like to describe a range of values for the variables in a predicate. A *quantifier* modifies a predicate by describing whether some or all elements of the domain satisfy the predicate.

We will need only two quantifiers: universal and existential. The *universal quantifier* "for all" is denoted by $\forall$. So the statement

$$(\forall x)P(x)$$

says that $P(x)$ is true *for all* x in the domain. The *existential quantifier* "there exists" is denoted by $\exists$. The statement

$$(\exists x)P(x)$$

says that *there exists* an element x of the domain such that $P(x)$ is true; in other words, $P(x)$ is true *for some* x in the domain.

For example, if $E(x)$ is the real number equation $x^2 - x - 6 = 0$, then the expression

$$(\exists x)E(x)$$

says, "There is some real number x such that $x^2 - x - 6 = 0$," or more simply, "The equation $x^2 - x - 6 = 0$ has a solution." The variable x is no longer a free variable, since the $\exists$ quantifier changes the role it plays in the sentence.

If $Z(x)$ represents the real number equation $x \cdot 0 = 0$, the expression

$$(\forall x)Z(x)$$

means "For all real numbers x, $x \cdot 0 = 0$." Again, this is a sentence without free variables, since the range of possible values for x is clearly specified.

When we put a quantifier in front of a predicate, we form a *quantified statement*. Since the quantifier restricts the range of values for the variables in the predicate, the quantified statement is either true or false (but not both). In the above examples, $(\exists x)E(x)$ and $(\forall x)Z(x)$ are both true, while the statement

$$(\forall x)E(x)$$

is false, since there are some real numbers that do not satisfy the equation $x^2 - x - 6 = 0$.

The real power of predicate logic comes from combining quantifiers, predicates, and the symbols of propositional logic. For example, if we would like to claim that there is a negative number that satisfies the equation $x^2 - x - 6 = 0$, we could define a new predicate

$$N(x) = \text{“}x \text{ is negative.”}$$

Then the statement

$$(\exists x)(N(x) \wedge E(x))$$

translates as "There exists some real number x such that x is negative and $x^2 - x - 6 = 0$."

The *scope* of a quantifier is the part of the formula to which the quantifier refers. In a complicated formula in predicate logic, it is important to use parentheses to indicate the scope of each quantifier. In general, the scope is what lies inside the set of parentheses right after the quantifier:

$$(\forall x)(\ldots \text{scope of } \forall \ldots), \qquad (\exists x)(\ldots \text{scope of } \exists \ldots).$$

In the statement $(\exists x)(N(x) \wedge E(x))$, the scope of the $\exists$ quantifier is the expression $N(x) \wedge E(x)$.

1.3.3 Translation

There are lots of different ways to write quantified statements in English. Translating back and forth between English statements and predicate logic is a skill that takes practice.

Example 1.10 Using all cars as a domain, if

$$P(x) = \text{“}x \text{ gets good mileage.”}$$
$$Q(x) = \text{“}x \text{ is large.”}$$

then the statement $(\forall x)(Q(x) \to \neg P(x))$ could be translated very literally as

> "For all cars x, if x is large, then x does not get good mileage."

However, a more natural translation of the same statement is

> "All large cars get bad mileage."

or

> "There aren't any large cars that get good mileage."

If we wanted to say the opposite, that is, that there are some large cars that get good mileage, we could write the following.

$$(\exists x)(P(x) \wedge Q(x))$$

We'll give a formal proof that this negation is correct in Example 1.13.

The next example shows how a seemingly simple mathematical statement yields a rather complicated formula in predicate logic. The careful use of predicates can help reveal the logical structure of a mathematical claim.

Example 1.11 In the domain of all integers, let $P(x) =$ "x is even." We can express the fact that the sum of an even number with an odd number is odd as follows.

$$(\forall x)(\forall y)[(P(x) \wedge \neg P(y)) \to (\neg P(x+y))]$$

Of course, the literal translation of this quantified statement is "For all integers x and for all integers y, if x is even and y is not even, then $x + y$ is not even," but we normally say something informal like "An even plus an odd is odd."

This last example used two universal quantifiers to express a fact about an arbitrary pair x, y of integers. The next example shows what can happen when you combine universal and existential quantifiers in the same statement.

Example 1.12 In the domain of all real numbers, let $G(x, y)$ be the predicate "$x > y$." The statement

$$(\forall y)(\exists x)G(x, y)$$

says literally that "For all numbers y, there exists some number x such that $x > y$," or more simply, "Given any number y, there is some number that is greater than y." This statement is clearly true: the number $y + 1$ is always greater than y, for example. However, the statement

$$(\exists x)(\forall y)G(x, y)$$

translates literally as "There exists a number x such that, for all numbers y, $x > y$." In simpler language, this statement says, "There is some number that is greater than any other number." This statement is clearly false, because there is no largest number.

The order of the quantifiers matters. In both of these statements, a claim is made that x is greater than y. In the first statement, you are first given an arbitrary number y, then the claim is that it is possible to find some x that is greater than it. However, the second statement claims there is some number x, such that, given any other y, x will be the greater number. In the second statement, you must decide on what x is before you pick y. In the first statement, you pick y first, then you can decide on x.

1.3.4 Negation

The most important thing you need to be able to do with predicate logic is to write down the negation of a quantified statement. As with propositional logic, there are some formal equivalences that describe how negation works. Table 1.5 lists two important rules for forming the opposite of a quantified statement. It is easy to see the formal pattern of these two rules: to negate a quantified statement, bring the negation inside the quantifier, and switch the quantifier.

Let's interpret the negation rules in the context of an example. In the domain of all people, let $L(x)$ stand for "x is a liar." The universal negation rule says that the negation of "All people are liars" is "There exists a person who is not a liar." In symbols,

$$\neg[(\forall x)L(x)] \Leftrightarrow (\exists x)(\neg L(x)).$$

Similarly, the existential negation rule says that the negation of "There exists a liar" is "There are no liars."

Example 1.13 In Example 1.10, we discussed what the negation of the statement

"All large cars get bad mileage."

should be. We can answer this question by negating the formal statement

$$(\forall x)(Q(x) \to \neg P(x)) \tag{1.3.1}$$

Equivalence	Name
$\neg[(\forall x)P(x)] \Leftrightarrow (\exists x)(\neg P(x))$	universal negation
$\neg[(\exists x)P(x)] \Leftrightarrow (\forall x)(\neg P(x))$	existential negation

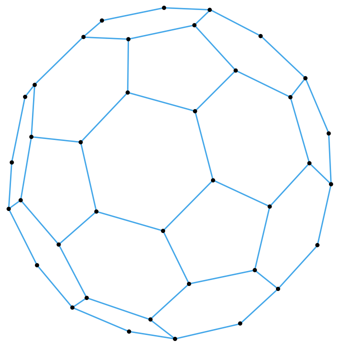
Table 1.5 Negation rules for predicate logic.

using a proof sequence. We'll suppose as given the negation of statement 1.3.1, and deduce an equivalent statement.

Statements	Reasons
1. $\neg[(\forall x)(Q(x) \to \neg P(x))]$	given
2. $(\exists x)\neg(Q(x) \to \neg P(x))$	universal negation
3. $(\exists x)\neg(\neg Q(x) \vee \neg P(x))$	implication
4. $(\exists x)(\neg(\neg Q(x)) \wedge \neg(\neg P(x)))$	De Morgan's law
5. $(\exists x)(Q(x) \wedge P(x))$	double negation
6. $(\exists x)(P(x) \wedge Q(x))$	commutativity

Notice that the result of our formal argument agrees with the intuitive negation we did in Example 1.10: There exists some car that is both large and gets good mileage.

Example 1.14 Let the domain be all faces of the following truncated icosahedron (also known as a soccer ball).



Consider the following predicates.

$$\begin{aligned} P(x) &= \text{“}x\text{ is a pentagon.”} \\ H(x) &= \text{“}x\text{ is a hexagon.”} \\ B(x, y) &= \text{“}x\text{ borders }y\text{.”} \end{aligned}$$

Here we say that two polygons border each other if they share an edge. Confirm that the following observations are true for any truncated icosahedron.

1. No two pentagons border each other.

2. Every pentagon borders some hexagon.

3. Every hexagon borders another hexagon.

Write these statements in predicate logic, and negate them. Simplify the negated statements so that no predicate lies withing the scope of a negation. Translate your negated statement back into English.

Solution: The formalizations of these statements are as follows.

1. $(\forall x)(\forall y)((P(x) \wedge P(y)) \to \neg B(x,y))$
2. $(\forall x)(P(x) \to (\exists y)(H(y) \wedge B(x,y)))$
3. $(\forall x)(H(x) \to (\exists y)(H(y) \wedge B(x,y))$

We'll negate (2), and leave the others as exercises. See if you can figure out the reasons for each equivalence.

$$\begin{aligned}
&\neg[(\forall x)(P(x) \to (\exists y)(H(y) \wedge B(x,y)))] \\
\Leftrightarrow\ &(\exists x)[\neg(P(x) \to (\exists y)(H(y) \wedge B(x,y)))] \\
\Leftrightarrow\ &(\exists x)[\neg(\neg P(x) \vee (\exists y)(H(y) \wedge B(x,y)))] \\
\Leftrightarrow\ &(\exists x)[\neg\neg(P(x) \wedge \neg(\exists y)(H(y) \wedge B(x,y)))] \\
\Leftrightarrow\ &(\exists x)[\neg\neg(P(x) \wedge (\forall y)\neg(H(y) \wedge B(x,y)))] \\
\Leftrightarrow\ &(\exists x)(P(x) \wedge (\forall y)\neg(H(y) \wedge B(x,y))) \\
\Leftrightarrow\ &(\exists x)(P(x) \wedge (\forall y)(\neg H(y) \vee \neg B(x,y))) \\
\Leftrightarrow\ &(\exists x)(P(x) \wedge (\forall y)(H(y) \to \neg B(x,y)))
\end{aligned}$$

This last statement says that there exists an x such that x is a pentagon and, for any y, if y is a hexagon, then x does not border y. In other words, there is some pentagon that borders no hexagon. If you found a solid with this property, it couldn't be a truncated icosahedron. ♢

1.3.5 Two Common Constructions

There are two expressions that come up often, and knowing the predicate logic for these expressions makes translation much easier. The first is the statement

All ⟨blanks⟩ are ⟨something⟩.

For example, "All baseball players are rich," or "All oysters taste funny." In general, if $P(x)$ and $Q(x)$ are the predicates "x is ⟨blank⟩" and "x is ⟨something⟩" respectively, then the predicate logic expression

$$(\forall x)(P(x) \to Q(x))$$

translates as "For all x, if x is ⟨blank⟩, then x is ⟨something⟩." Put more simply, "All x's with property ⟨blank⟩ must have property ⟨something⟩," or even simpler, "All ⟨blanks⟩ are ⟨something⟩." In the domain of all people, if $R(x)$ stands for "x is rich" and $B(x)$ stands for "x is a baseball player," then

$$(\forall x)(B(x) \to R(x))$$

is the statement "All baseball players are rich."

The second construction is of the form

There is a ⟨blank⟩ that is ⟨something⟩.

For example, "There is a rich baseball player," or "There is a funny-tasting oyster." This expression has the following form in predicate logic.

$$(\exists x)(P(x) \wedge Q(x))$$

Note that this translates literally as "There is some x such that x is ⟨blank⟩ and x is ⟨something⟩," which is what we want. In the domain of shellfish, if $O(x)$ is the predicate "x is an oyster" and $F(x)$ is the predicate "x tastes funny," then

$$(\exists x)(F(x) \wedge O(x))$$

would translate as "There is a funny-tasting oyster." Note that you could also say "There is an oyster that tastes funny," "Some oysters taste funny," or, more awkwardly, "There is a funny-tasting shellfish that is an oyster." These statements all mean the same thing.

Exercises 1.3

1. In the domain of integers, let $P(x, y)$ be the predicate "$x \cdot y = 12$." Tell whether each of the following statements is true or false.
   (a) $P(3, 4)$
   (b) $P(3, 5)$
   (c) $P(2, 6) \vee P(3, 7)$
   (d) $(\forall x)(\forall y)(P(x, y) \to P(y, x))$
   (e) $(\forall x)(\exists y)P(x, y)$
2. In the domain of all penguins, let $D(x)$ be the predicate "x is dangerous." Translate the following quantified statements into simple, everyday English.

(a) $(\forall x)D(x)$

(b) $(\exists x)D(x)$

(c) $\neg(\exists x)D(x)$

(d) $(\exists x)\neg D(x)$

3. In the domain of all movies, let $V(x)$ be the predicate "x is violent." Write the following statements in the symbols of predicate logic.

   (a) Some movies are violent.

   (b) Some movies are not violent.

   (c) No movies are violent.

   (d) All movies are violent.

4. In the domain of all books, consider the following predicates.

$$\begin{aligned} H(x) &= \text{“}x\text{ is heavy.”} \\ C(x) &= \text{“}x\text{ is confusing.”} \end{aligned}$$

   Translate the following statements in predicate logic into ordinary English.

   (a) $(\forall x)(H(x) \to C(x))$

   (b) $(\exists x)(C(x) \wedge H(x))$

   (c) $(\forall x)(C(x) \vee H(x))$

   (d) $(\exists x)(H(x) \wedge \neg C(x))$

5. The domain of the following predicates is the set of all plants.

$$\begin{aligned} P(x) &= \text{“}x\text{ is poisonous.”} \\ Q(x) &= \text{“Jeff has eaten }x\text{.”} \end{aligned}$$

   Translate the following statements into predicate logic.

   (a) Some plants are poisonous.

   (b) Jeff has never eaten a poisonous plant.

   (c) There are some nonpoisonous plants that Jeff has never eaten.

6. In the domain of nonzero integers, let $I(x, y)$ be the predicate "x/y is an integer." Determine whether the following statements are true or false, and explain why.

   (a) $(\forall y)(\exists x)I(x, y)$

   (b) $(\exists x)(\forall y)I(x, y)$

7. The domain of the following predicates is the set of all traders who work at the Tokyo Stock Exchange.

$$\begin{aligned} P(x, y) &= \text{“}x \text{ makes more money than } y\text{.”} \\ Q(x, y) &= \text{“}x \neq y\text{”} \end{aligned}$$

   Translate the following predicate logic statements into *ordinary, everyday* English. (Don't simply give a word-for-word translation; try to write sentences that make sense.)

   (a) $(\forall x)(\exists y)P(x, y)$
   (b) $(\exists y)(\forall x)(Q(x, y) \to P(x, y))$
   (c) Which statement is impossible in this context? Why?

8. Translate the following statements into predicate logic using the two common constructions in Section 1.3.5. State what your predicates are, along with the domain of each.

   (a) All natural numbers are integers.
   (b) Some integers are natural numbers.
   (c) All the streets in Cozumel, Mexico are one-way.
   (d) Some streets in London don't have modern curb cuts.

9. Write the following statements in predicate logic. Define what your predicates are. Use the domain of all quadrilaterals.

   (a) All rhombuses are parallelograms.
   (b) Some parallelograms are not rhombuses.

10. Let the following predicates be given. The domain is all people.

$$\begin{aligned} R(x) &= \text{“}x \text{ is rude.”} \\ \neg R(x) &= \text{“}x \text{ is pleasant.”} \\ C(x) &= \text{“}x \text{ is a child.”} \end{aligned}$$

   (a) Write the following statement in predicate logic.

   There is at least one rude child.

   (b) Formally negate your statement from part (a).
   (c) Write the English translation of your negated statement.

11. In the domain of all people, consider the following predicate.

$$P(x, y) = \text{“}x \text{ needs to love } y\text{.”}$$

(a) Write the statement "Everybody needs somebody to love" in predicate logic.
(b) Formally negate your statement from part (a).
(c) Write the English translation of your negated statement.

12. The domain for this problem is some unspecified collection of numbers. Consider the predicate

$$P(x, y) = \text{"}x \text{ is greater than } y.\text{"}$$

(a) Translate the following statement into predicate logic.

Every number has a number that is greater than it.

(b) Negate your expression from part (a), and simplify it so that no quantifier lies within the scope of a negation.
(c) Translate your expression from part (b) into understandable English. Don't use variables in your English translation.

13. Any equation or inequality with variables in it is a predicate in the domain of real numbers. For each of the following statements, tell whether the statement is true or false.

(a) $(\forall x)(x^2 > x)$
(b) $(\exists x)(x^2 - 2 = 1)$
(c) $(\exists x)(x^2 + 2 = 1)$
(d) $(\forall x)(\exists y)(x^2 + y = 4)$
(e) $(\exists y)(\forall x)(x^2 + y = 4)$

14. The domain of the following predicates is all integers greater than 1.

$$P(x) = \text{"}x \text{ is prime.\text{"}}$$
$$Q(x, y) = \text{"}x \text{ divides } y.\text{"}$$

Consider the following statement.

For every x that is not prime, there is some prime y that divides it.

(a) Write the statement in predicate logic.
(b) Formally negate the statement.
(c) Write the English translation of your negated statement.

15. Write the following statement in predicate logic, and negate it. Say what your predicates are, along with the domains.

Let x and y be real numbers. If x is rational and y is irrational, then $x + y$ is irrational.

16. Refer to Example 1.14.

    (a) Give the reasons for each $\Leftrightarrow$ step in the simplification of the formal negation of statement (2).

    (b) Give the formal negation of statement (1). Simplify your answer so that no quantifier lies within the scope of a negation. Translate your negated statement back into English.

    (c) Give the formal negation of statement (3). Simplify your answer. Translate your negated statement back into English.

17. Let the following predicates be given in the domain of all triangles.

$$\begin{aligned} R(x) &= \text{“}x \text{ is a right triangle.”} \\ B(x) &= \text{“}x \text{ has an obtuse angle.”} \end{aligned}$$

    Consider the following statements.

$$\begin{aligned} S_1 &= \neg(\exists x)(R(x) \wedge B(x)) \\ S_2 &= (\forall x)(R(x) \rightarrow \neg B(x)) \end{aligned}$$

    (a) Write a proof sequence to show that $S_1 \Leftrightarrow S_2$.

    (b) Write S_1 in ordinary English.

    (c) Write S_2 in ordinary English.

18. Let the following predicates be given. The domain is all computer science classes.

$$\begin{aligned} I(x) &= \text{“}x \text{ is interesting.”} \\ U(x) &= \text{“}x \text{ is useful.”} \\ H(x,y) &= \text{“}x \text{ is harder than } y\text{.”} \\ M(x,y) &= \text{“}x \text{ has more students than } y\text{.”} \end{aligned}$$

    (a) Write the following statements in predicate logic.

        i. All interesting CS classes are useful.

        ii. There are some useful CS classes that are not interesting.

        iii. Every interesting CS class has more students than any non-interesting CS class.

    (b) Write the following predicate logic statement in everyday English. Don't just give a word-for-word translation; your sentence should make sense.

$$(\exists x)[I(x) \wedge (\forall y)(H(x,y) \rightarrow M(y,x))]$$

(c) Formally negate the statement from part (b). Simplify your negation so that no quantifier lies within the scope of a negation. State which derivation rules you are using.

(d) Give a translation of your negated statement in everyday English.

19. Let the following predicates be given. The domain is all cars.

$$\begin{aligned} F(x) &= \text{“}x\text{ is fast.”} \\ S(x) &= \text{“}x\text{ is a sports car.”} \\ E(x) &= \text{“}x\text{ is expensive.”} \\ A(x,y) &= \text{“}x\text{ is safer than }y\text{.”} \end{aligned}$$

(a) Write the following statements in predicate logic.

i. All sports cars are fast.

ii. There are fast cars that aren't sports cars.

iii. Every fast sports car is expensive.

(b) Write the following predicate logic statement in everyday English. Don't just give a word-for-word translation; your sentence should make sense.

$$(\forall x)[S(x) \to (\exists y)(E(y) \wedge A(y,x))]$$

(c) Formally negate the statement from part (b). Simplify your negation so that no quantifier lies within the scope of a negation. State which derivation rules you are using.

(d) Give a translation of your negated statement in everyday English.

20. (a) Give an example of a pair of predicates $P(x)$ and $Q(x)$ in some domain to show that the $\exists$ quantifier does not distribute over the $\wedge$ connective. That is, give an example to show that the statements

$$(\exists x)(P(x) \wedge Q(x)) \quad \text{and} \quad (\exists x)P(x) \wedge (\exists x)Q(x)$$

are not logically equivalent.

(b) It is true, however, that $\exists$ distributes over $\vee$. That is,

$$(\exists x)(P(x) \vee Q(x)) \Leftrightarrow (\exists x)P(x) \vee (\exists x)Q(x)$$

is an equivalence rule for predicate logic. Verify that your example from part (a) satisfies this equivalence.

21. (a) Give an example to show that $\forall$ does not distribute over $\vee$.

(b) It is a fact that $\forall$ distributes over $\wedge$. Check that your example from part (a) satisfies this equivalence rule.

1.4 Logic in Mathematics

There is much more that we could say about symbolic logic; we have only scratched the surface. But we have developed enough tools to help us think carefully about the types of language mathematicians use. This section provides an overview of the basic mathematical "parts of speech."

Most mathematics textbooks (including this one) label important statements with a heading, such as "Theorem," "Definition," or "Proof." The name of each statement describes the role it plays in the logical development of the subject. Therefore it is important to understand the meanings of these different statement labels.

1.4.1 The Role of Definitions in Mathematics

When we call a statement a "definition" in mathematics, we mean something different from the usual everyday notion. Everyday definitions are *descriptive.* The thing being defined already exists, and the purpose of the definition is to describe the thing. When a dictionary defines some term, it is characterizing the way the term is commonly used. For example, if we looked up the definition of "mortadella" in the *Oxford English Dictionary* (OED), we would read the following.

> Any of several types of Italian (esp. Bolognese) sausage; (now) spec. a thick smooth-textured pork sausage containing pieces of fat and typically served in slices.

The authors of the OED have done their best to describe what is meant by the term "mortadella." A good dictionary definition is one that does a good job describing something.

In mathematics, by contrast, a *definition* is a statement that stipulates the meaning of a new term, symbol, or object. For example, a plane geometry textbook may define parallel lines as follows.

Definition 1.4 Two lines are *parallel* if they have no points in common.

The job of this definition is not to describe parallel lines, but rather to specify exactly what we mean when we use the word "parallel." Once parallel lines have been defined in this way, the statement "l and m are parallel" means "l and m have no points in common." We may have some intuitive idea of what l and m might look like (e.g., they must run in the same direction), but for the purposes of any future arguments, the only thing we really *know* about l and m is that they don't intersect each other.

In a sense, a mathematical definition calls into being the object it defines. Without a definition of parallel lines, there can be no mathematical discussion of parallel lines. A dictionary definition, by contrast, doesn't create anything new; we don't imagine that the writers of the OED are in the business of making sausage (and everything else, for that matter).

Why is this distinction important? It means that the whole meaning of a mathematical statement depends on the definitions of the terms involved. If you don't understand a mathematical statement, start looking at the definitions of all the terms. These definitions stipulate the meanings of the terms. The statement won't make sense without them.

For example, suppose we want to state and prove some facts about even and odd numbers. We already know what even and odd numbers are; we all come to this task with a previously learned *concept image* of "even" and "odd." Our concept image is what we think of when we hear the term: an even number ends in an even digit, an odd number can't be divided in half evenly, "2, 4, 6, 8; who do we appreciate," etc. When writing mathematically, however, it is important not to rely too heavily on these concept images. Any mathematical statement about even and odd numbers derives its meaning from from definitions. We specify these as follows.

Definition 1.5 An integer n is *even* if $n = 2k$ for some integer k.

Definition 1.6 An integer n is *odd* if $n = 2k + 1$ for some integer k.

Given these definitions, we can justify the statement "17 is odd" by noting that $17 = 2 \cdot 8 + 1$. In fact, this equation is precisely the meaning of the statement that "17 is odd;" there is some integer k (in this case, $k = 8$) such that $17 = 2k + 1$. You already "knew" that 17 is odd, but in order to mathematically *prove* that 17 is odd, you need to use the definition.

Mathematical definitions must be extremely precise, and this can make them somewhat limited. Often our concept image contains much more information than the definition supplies. For example, we probably all agree that it is impossible for a number to be both even and odd, but this fact doesn't follow immediately from Definitions 1.5 and 1.6. To say that some given number n is even means that $n = 2k_1$ for some integer k_1, and to say that it is odd is to say that $n = 2k_2 + 1$ for some integer k_2. (Note that k_1 and k_2 may be different.) Now, is this possible? It would imply that $2k_1 = 2k_2 + 1$, which says that $1 = 2(k_1 - k_2)$, showing that 1 is even, by Definition 1.5. At this point we might object that 1 is odd, so it can't be even, but this reasoning is circular: we were trying to show that a number cannot be both even and odd. We haven't yet shown this fact, so we can't use this fact in our argument. It turns out that Definitions 1.5 and 1.6 alone are not enough to show that a number can't be both even and odd; to do so requires more facts about integers, as we will see in Section 1.5.

One reasonable objection to the above discussion is that our definition of odd integers was too limiting; why not define an odd integer to be an integer that isn't even? This is certainly permissible, but then it would be hard[4] to show that an odd integer n can be written as $2k+1$ for some integer k. And we can't have two definitions for the same term. Stipulating a definition usually involves a choice on the part of the author, but once this choice is made, we are stuck with it. We have chosen to define odd integers as in Definition 1.6, so this is what we mean when we say "odd."

Since definitions are stipulative, they are logically "if and only if" statements. However, it is common to write definitions in the form

[Object] x is [defined term] if [defining property about x].

The foregoing examples all take this form. In predicate logic, if

$$\begin{aligned} D(x) &= x \text{ is [defined term]} \\ P(x) &= \text{[defining property about } x\text{]} \end{aligned}$$

then the above definition really means the following.

$$(\forall x)(P(x) \leftrightarrow D(x))$$

However, this is not what the definition says at face value. Definitions look like "if ... then" statements, but we interpret them as "if and only if" statements because they are definitions. For example, Definition 1.4 is stipulating the property that defines all parallel lines, not just a property some parallel lines might have. Strictly speaking, we really should use "if and only if" instead of "if" in our definitions. But the use of "if" is so widespread that most mathematicians would find a definition like

Two lines are *parallel* if and only if they have no points in common.

awkward to read. Since this statement is a definition, it is redundant to say "if and only if."

1.4.2 Other Types of Mathematical Statements

Definitions are a crucial part of mathematics, but there are other kinds of statements that occur frequently in mathematical writing. Any mathematical system needs to start with some assumptions. Without any statements to build on, we would never be able to prove any new statements. Statements that are assumed without proof are called *postulates* or *axioms*. For example, the following is a standard axiom about the natural numbers.

If n is a natural number, so is $n+1$.

4. Actually, it would be impossible, without further information.

Axioms are typically very basic, fundamental statements about the objects they describe. Any theorem in mathematics is based on the assumption of some set of underlying axioms. So to say theorems are "true" is not to say they are true in any absolute sense, only that they are true, given that some specified set of axioms is true.

A *theorem* is a statement that follows logically from statements we have already established or taken as given. Before a statement can be called a theorem, we must be able to prove it. A *proof* is a valid argument, based on axioms, definitions, and proven theorems, that demonstrates the truth of a statement. The derivation sequences that we did in Section 1.2 were very basic mathematical proofs. We will see more interesting examples of proofs in the next section.

We also use the terms *lemma*, *proposition*, and *corollary* to refer to specific kinds of theorems. Usually authors will label a result a lemma if they are using it to prove another result. Some authors make no distinction between a theorem and a proposition, but the latter often refers to a result that is perhaps not as significant as a full-fledged theorem. A corollary is a theorem that follows immediately from another result via a short argument.

One last word on terminology: A statement that we intend to prove is called a *claim*. A statement that we can't yet prove but that we suspect is true is called a *conjecture*.

1.4.3 Counterexamples

Often mathematical statements are of the form

$$(\forall x)P(x). \tag{1.4.1}$$

We saw in the previous section that the negation of statement 1.4.1 is

$$(\exists x)\neg P(x). \tag{1.4.2}$$

So either statement 1.4.1 is true, or statement 1.4.2 is true, but not both. If we can find a single value for x that makes $\neg P(x)$ true, then we know that statement 1.4.2 is true, and therefore we also know that statement 1.4.1 is false.

For example, we might be tempted to make the following statement.

$$\text{Every prime number is odd.} \tag{1.4.3}$$

But 2 is an example of a prime number that is not odd, so statement 1.4.3 is false. A particular value that shows a statement to be false is called a *counterexample* to the statement.

Another common logical form in mathematics is the universal if–then statement.

$$(\forall x)(P(x) \to Q(x))$$

To find a counterexample to a statement of this form, we need to find some x that satisfies the negation

$$(\exists x)\neg(P(x) \rightarrow Q(x)).$$

This last statement is equivalent (using implication and De Morgan's law) to

$$(\exists x)(P(x) \wedge \neg Q(x)).$$

So a counterexample is something that satisfies P and violates Q.

Example 1.15 Find a counterexample to the following statement.

> For all sequences of numbers $a_1, a_2, a_3, \ldots$, if $a_1 < a_2 < a_3 < \cdots$, then some a_i must be positive.

Solution: By the above discussion, we need an example of a sequence that satisfies the "if" part of the statement and violates the "then" part. In other words, we need to find an increasing sequence that is always negative. Something with a horizontal asymptote will work: $a_n = -1/n$ is one example. Note that $-1 < -1/2 < -1/3 < \cdots$, but all the terms are less than zero. $\diamondsuit$

1.4.4 Axiomatic Systems

In rigorous, modern treatments of mathematics, any system (e.g., plane geometry, the real numbers) must be clearly and unambiguously defined from the start. The definitions should leave nothing to intuition; they mean what they say and nothing more. It is important to be clear about the assumptions, or axioms, for the system. Every theorem in the system must be proved with a valid argument, using only the definitions, axioms, and previously proved theorems of the system.

This sounds good, but it is actually impossible. It is impossible because we can't define everything; before we write the first definition we have to have some words in our vocabulary. These starting words are called *undefined terms*. An undefined term has no meaning—it is an abstraction—its meaning comes from the role it plays in the axioms of the system. A collection of undefined terms and axioms is called an *axiomatic system*.

Axiomatic systems for familiar mathematics such as plane geometry and the real number system are actually quite complicated and beyond the scope of an introductory course. Here we will look at some very simple axiomatic systems to get a feel for how they work. This will also give us some experience with logic in mathematics.

The first example defines a "finite geometry," that is, a system for geometry with a finite number of points. Although this system speaks of "points" and "lines," these terms don't mean the same thing they meant in high school geometry. In fact, these terms don't mean anything at all, to begin with at least. The only thing we know about points and lines is that they satisfy the given axioms.

Example 1.16 Axiomatic system for a four-point geometry.

Undefined terms: point, line, is on

Axioms:

1. For every pair of distinct points x and y, there is a unique line such that x is on l and y is on l.
2. Given a line l and a point x that is not on l, there is a unique line m such that x is on m and no point on l is also on m.
3. There are exactly four points.
4. It is impossible for three points to be on the same line.

Notice that these axioms use terms from logic in addition to the undefined terms. We are also using numbers ("four" and "three"), even though we haven't defined an axiomatic system for the natural numbers. In this case, our use of numbers is more a convenient shorthand than anything; we aren't relying on any properties of the natural numbers such as addition, ordering, divisibility, etc.

It is common to use an existing system to define a new axiomatic system. For example, some modern treatments of plane geometry use axioms that rely on the real number system. The axioms in Example 1.16 use constructions from predicate logic. In any event, these prerequisite systems can also be defined axiomatically, so systems that use them are still fundamentally axiomatic.

Definitions can help make an axiomatic system more user-friendly. In the four-point geometry of Example 1.16, we could make the following definitions. In these (and other) definitions, the word being defined is in *italics*.

Definition 1.7 A line l *passes through* a point x if x is on l.

Definition 1.7 gives us a convenient alternative to using the undefined term "is on." For example, in the first axiom, it is a bit awkward to say "x is on l and y is on l," but Definition 1.7 allows us to rephrase this as "l passes through x and y." The definition doesn't add any new features to the system; it just helps us describe things more easily. This is basically what any definition in mathematics does. The following definition is a slight restatement of Definition 1.4, modified to fit the terminology of this system.

Definition 1.8 Two lines, l and m, are *parallel* if there is no point x, such that x is on l and x is on m.

Now we could rephrase the second axiom of Example 1.16 as follows.

2. Given a line l and a point x that is not on l, there is a unique line m passing through x such that m is parallel to l.

A simple theorem and proof would look like this.

Theorem 1.1 *In the axiomatic system of Example 1.16, there are at least two distinct lines.*

Proof By Axiom 3, there are distinct points x, y, and z. By Axiom 1, there is a line l_1 through x and y, and a line l_2 through y and z. By Axiom 4, x, y, and z are not on the same line, so l_1 and l_2 must be distinct lines. $\square$

A *model* of an axiomatic system is an interpretation in some context in which all the undefined terms have meanings and all the axioms hold. Models are important because they show that it is possible for all the axioms to be true, at least in some context. And any theorem that follows from the axioms must also be true for any valid model.

Let's make a model for the system in Example 1.16. Let a "point" be a dot, and let a "line" be a simple closed loop. A point "is on" a line if the dot is inside the loop. Figure 1.3 shows this model. Check that all the axioms hold.

This model doesn't really match our concept image of points and lines in ordinary geometry, but it's still a valid model. In an axiomatic system, the properties of the objects are determined by the axioms and described by the theorems and definitions. We may think we know what points and lines should look like, but mathematically speaking we only know whatever we can prove about them. In the exercises you will construct a more intuitive model for this system.

The mathematician David Hilbert (1862–1943) was largely responsible for developing the modern approach to axiomatics. Hilbert, reflecting on the

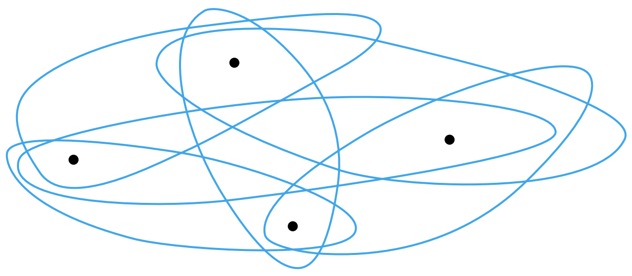


Figure 1.3 A model for the axiomatic system in Example 1.16 using dots and loops.

abstract nature of axiomatic systems, remarked, “Instead of points, lines, and planes, one must be able to say at all times tables, chairs, and beer mugs” [24]. If we used a word processor to replace every occurrence of “point” with “table” and every occurrence of “line” with “chair” in the above axioms, definitions, theorem, and proof, the theorem would still hold, and the proof would still be valid.

The next example is referred to in the exercises. The choice of undefined terms emphasizes that these terms, by themselves, carry no meaning.

Example 1.17 Badda-Bing axiomatic system.

Undefined terms: badda, bing, hit

Axioms:

1. Every badda hits exactly four bings.
2. Every bing is hit by exactly two baddas.
3. If x and y are distinct baddas, each hitting bing q, then there are no other bings hit by both x and y.
4. There is at least one bing.

One possible model for the Badda-Bing system is shown in Figure 1.4. The picture shows an infinite collection of squares; the central square connects to

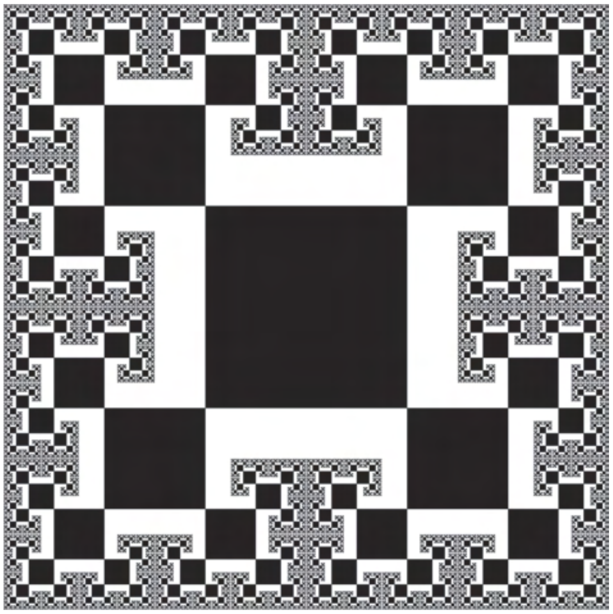


Figure 1.4 A fractal model for the Badda-Bing geometry.

four other squares whose sides are half as long. Each of these squares connects to three other smaller squares, and each of those connects to three others, and so on. This is an example of a *fractal*—a shape with some sort of infinitely repetitive geometric structure. (We'll say more about fractals in Chapter 3.)

In this model, a "badda" is a square, and a "bing" is a corner, or *vertex,* of a square. A square "hits" a vertex if the vertex belongs to the square. Since every square has four vertices, Axiom 1 is satisfied. Axiom 2 holds because every vertex in the model belongs to exactly two squares. Axiom 3 is a little harder to see: if squares x and y share a vertex q, there is no way they can share another vertex. And Axiom 4 is obviously true—there are lots of bings.

Exercises 1.4

1. Look up the word "root" in a dictionary. It should have several different definitions. Find a definition that is (a) descriptive and another definition that is (b) stipulative.
2. Find another word in the English language that has both descriptive and stipulative definitions.
3. Use Definition 1.5 to explain why 104 is an even integer.
4. Let n be an integer. Use Definition 1.6 to explain why $2n + 7$ is an odd integer.
5. Let n_1 and n_2 be even integers.
   (a) Use Definition 1.5 to write n_1 and n_2 in terms of integers k_1 and k_2, respectively.
   (b) Write the product $n_1 n_2$ in terms of k_1 and k_2. Simplify your answer.
   (c) Write the sum $n_1 + n_2$ in terms of k_1 and k_2. Simplify your answer.
6. There are several different models for geometries in which the points are ordered pairs (x, y) of real numbers; we plot these points in the usual way in the xy-plane. In such a geometry, there can be a formula for the *distance* between two points (x_1, y_1) and (x_2, y_2). For example, in Euclidean geometry, distance is given by the usual Euclidean distance formula:
$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$
   In any geometry with a distance formula, we can define a *circle* as follows.

Definition 1.9 A *circle* centered at (a, b) with radius r is the collection of all points (x, y) whose distance from (a, b) is r.

(a) Use Definition 1.9 to give an equation for the circle with radius 5 centered at $(0, 0)$ in the Euclidean plane.

(b) Plot the circle from part (a) in the xy-plane.

(c) In the *Taxicab geometry*, the distance between two points (x_1, y_1) and (x_2, y_2) is given by the following formula.

$$\text{Distance} = |x_2 - x_1| + |y_2 - y_1|$$

(This is called "taxicab" distance because it models the distance you would have to travel if you were restricted to driving on a rectangular city grid.) In this model, use Definition 1.9 to plot the "circle" with radius 5 centered at $(0, 0)$.

(d) Which type of circle (Euclidean or taxicab) agrees with your concept image of circle?

7. Consider the domain of all quadrilaterals. Let

$$A(x) = \text{“}x \text{ has four right angles.”}$$
$$R(x) = \text{“}x \text{ is a rectangle.”}$$

Write the meaning of each mathematical statement in predicate logic, keeping in mind the logical distinction between definitions and theorems.

(a) **Definition.** A quadrilateral is a *rectangle* if it has four right angles.

(b) **Theorem.** A quadrilateral is a rectangle if it has four right angles.

8. Write Definition 1.5 in predicate logic. Use the predicate $E(x)$ = "x is even" in the domain of integers.

9. Let the following statements be given.

Definition. A triangle is *scalene* if all of its sides have different lengths.

Theorem. A triangle is scalene if it is a right triangle that is not isosceles.

Suppose $\triangle ABC$ is a scalene triangle. Which of the following conclusions are valid? Why or why not?

(a) All of the sides of $\triangle ABC$ have different lengths.

(b) $\triangle ABC$ is a right triangle that is not isosceles.

10. What is the difference between an axiom and a theorem?

11. Let $P(n, x, y, z)$ be the predicate "$x^n + y^n = z^n$."

    (a) Write the following statement in predicate logic, using positive integers as the domain.

    > For every positive integer n, there exist positive integers x, y, and z such that $x^n + y^n = z^n$.

    (b) Formally negate your predicate logic statement from part (a). Simplify so that no quantifier lies within the scope of a negation.

    (c) In order to produce a counterexample to the statement in part (a), what, specifically, would you have to find?

12. Find a counterexample for each statement.

    (a) If n is prime, then $2^n - 1$ is prime.

    (b) Every triangle has at least one obtuse angle.[5]

    (c) For all real numbers x, $x^2 \geq x$.

    (d) For every positive nonprime integer n, if some prime p divides n, then some other prime q (with $q \neq p$) also divides n.

    (e) If all the sides of a quadrilateral are equal, then the diagonals of the quadrilateral are equal.

    (f) For every real number $N > 0$, there is some real number x such that $Nx > x$.

    (g) Let l, m, and n be lines in the plane. If $l \perp m$ and n intersect l, then n intersects m.

    (h) If p is prime, then $p^2 + 4$ is prime.

13. Which of the statements in the previous problem can be proved as theorems?

14. Consider the following theorem.

    > **Theorem.** Let x be a wamel. If x has been schlumpfed, then x is a borfin.

    Answer the following questions.

    (a) Give the converse of this theorem.

    (b) Give the contrapositive of this theorem.

    (c) Which statement, (a) or (b), is logically equivalent to the Theorem?

5. An angle is *obtuse* if it has measure greater than 90°.

15. Draw a model for the axiomatic system of four-point geometry (Example 1.16), where a "line" is a line segment, a "point" is an endpoint of a line segment, and a point "is on" a line if it is one of its endpoints.

16. In four-point geometry, use the axioms to explain why every point is on three different lines.

17. In four-point geometry, is it possible for two different lines to both pass through two given distinct points? Explain why or why not using the axioms.

18. In four-point geometry, do triangles exist? In other words, is it possible to have three distinct points, not on the same line, such that a line passes through each pair of points? Why or why not?

19. In four-point geometry, state a good definition to stipulate what it means for two lines to *intersect.*

20. Consider the following model for four-point geometry.

    Points: 1, 2, 3, 4

    Lines: $\boxed{1\ 2}$, $\boxed{1\ 3}$, $\boxed{1\ 4}$, $\boxed{2\ 3}$, $\boxed{2\ 4}$, $\boxed{3\ 4}$

    A point "is on" a line if the line's box contains the point.

    (a) Give a pair of parallel lines in this model. (Refer to Definition 1.8.)

    (b) Give a pair of intersecting lines in this model. (Use your definition from Exercise 19.)

21. Explain why, in the axiomatic system of Example 1.17, there must be at least seven distinct bings.

22. Consider the following definition in the system of Example 1.17.

    Definition. Let x and y be distinct baddas. We say that a bing q is a *boom* of x and y, if x hits q and y hits q.

    Rewrite Axiom 3 using this definition.

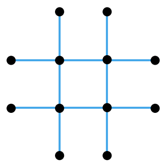
23. In the context of Example 1.17, consider the following predicates.

$$\begin{aligned} N(x,y) &= \text{“}x \neq y.\text{”} \\ D(x) &= \text{“}x \text{ is a badda.”} \\ G(x) &= \text{“}x \text{ is a bing.”} \\ H(x,y) &= \text{“}x \text{ hits } y.\text{”} \end{aligned}$$

    Use these predicates to write Axiom 3 in predicate logic.

24. Refer to Example 1.17 and Figure 1.4. Describe a different model, using squares and vertices, where all the squares are the same size.

25. In the axiomatic system of Example 1.17, let a "badda" be a line segment, let a "bing" be a point, and say that a line segment "hits" a point if it passes through it. In the diagram below, there are 4 baddas and 12 bings. Is this a model for the system? Which of the axioms does this model satisfy? Explain.



26. Describe a model for Example 1.17 with 10 bings, where a "badda" is a line segment and a "bing" is a point.

1.5 Methods of Proof

The types of proofs we did in Section 1.2 were fairly mechanical. We started with the given and constructed a sequence of conclusions, each justified by a deduction rule. We were able to write proofs this way because our mathematical system, propositional logic, was fairly small. Most mathematical contexts are much more complicated; there are more definitions, more axioms, and more complex statements to analyze. These more complicated situations do not easily lend themselves to the kind of structured proof sequences of Section 1.2. In this section we will look at some of the ways proofs are done in mathematics.

1.5.1 Direct Proofs

The structure of a proof sequence in propositional logic is straightforward: in order to prove $A \Rightarrow C$, we prove a sequence of results.

$$A \Rightarrow B_1 \Rightarrow B_2 \Rightarrow \cdots \Rightarrow B_n \Rightarrow C$$

A *direct proof* in mathematics has the same logic, but we don't usually write such proofs as lists of statement and reasons. Instead, this linear chain of implications is couched in mathematical prose and written in paragraph form.

Example 1.18 The proof of Theorem 1.1 on page 45 is a direct proof. Although this proof takes the form of a paragraph, the logical sequence of implications is easy to see.

There are distinct points x, y, and z.
$\Rightarrow$ There is a line l_1 through x and y, and a line l_2 through y and z.
$\Rightarrow$ x, y, and z are not on the same line, so $l_1 \neq l_2$.

These three statements are justified by Axioms 3, 1, and 4, respectively.

Example 1.19 Prove the following statement.

For all real numbers x, if $x > 1$, then $x^2 > 1$.

Proof Let x be a real number, and suppose $x > 1$. Multiplying both sides of this inequality by a positive number preserves the inequality, so we can multiply both sides by x to obtain $x^2 > x$. Since $x > 1$, we have $x^2 > x > 1$, or $x^2 > 1$, as required. □

It is worth looking back at this proof. The chain of implications is as follows.

$$x > 1 \;\Rightarrow\; x^2 > x \;\Rightarrow\; x^2 > 1 \tag{1.5.1}$$

Each conclusion is justified by an elementary fact from high school algebra, and the results are packaged in paragraph form. More precisely, the statement we were proving was actually a quantified statement of the form

$$(\forall x)(P(x) \to Q(x))$$

where $P(x)$ means "$x > 1$" and $Q(x)$ means "$x^2 > 1$." We see that the sequence of implications in Equation (1.5.1) is true no matter what value we initially choose for x. This is the meaning of the introductory phrase "Let x be a real number." We assume nothing about x other than that it is a real number; it is arbitrary in every other respect. We then treat $P(x)$ as given, and try to conclude $Q(x)$. Since x could have been any real number to start with, we have proved the implication for *all* x.

We state this type of proof as our first "Rule of Thumb" for proving theorems.

Rule of Thumb 1.1 To prove a statement of the form $(\forall x)(P(x) \to Q(x))$, begin your proof with a sentence of the form

Let x be [an element of the domain], and suppose $P(x)$.

A direct proof is then a sequence of justified conclusions culminating in $Q(x)$.

Before we look at another example of direct proof, we will need some tools for dealing with integers. We'll start with a definition for what it means for an integer x to *divide* another integer y.

Definition 1.10 An integer x *divides* an integer y if there is some integer k such that $y = kx$.

We write $x \mid y$ to denote that x divides y. An identical definition holds for natural numbers (i.e., positive integers). Just replace the three occurrences of "integer" in Definition 1.10 with "natural number."

We are not going to develop a rigorous axiomatic approach to the integers; such a treatment is beyond the scope of this course. When you deal with integer equations, feel free to use elementary facts from high school algebra. You can add something to both sides of an equation, use the distributive property, combine terms, and so on. However, there are certain facts about the integers that we will state as axioms, because they justify important steps in the proofs that follow.

Axiom 1.1 If a and b are integers, so are $a + b$ and $a \cdot b$.

Axiom 1.1 describes the *closure* property of the integers under addition and multiplication. Most number systems are closed under these two operations; you can't get a new kind of number by adding or multiplying. On the other hand, the integers are not closed under division: 2/3 is not an integer, even though 2 and 3 are.

Example 1.20 Prove the following.

For all integers a, b, and c, if $a \mid b$ and $a \mid c$, then $a \mid (b + c)$.

Proof Let integers a, b, and c be given, and suppose $a \mid b$ and $a \mid c$. Then, by Definition 1.10, there is some integer k_1 such that $b = k_1 a$ and there is some integer k_2 such that $c = k_2 a$. Therefore,

$$b + c = k_1 a + k_2 a = (k_1 + k_2)a.$$

By Axiom 1.1, $k_1 + k_2$ is an integer, so $a \mid (b + c)$, again by Definition 1.10. □

Notice that this proof illustrates how definitions are used in mathematics. We used the definition of "divides" in order to translate the given statement into an equation, we did some simple algebra on this equation to obtain a new equation, and we used the definition again to translate the new equation into the statement we were trying to prove. Figure 1.5 shows a "flow chart" for this proof technique.

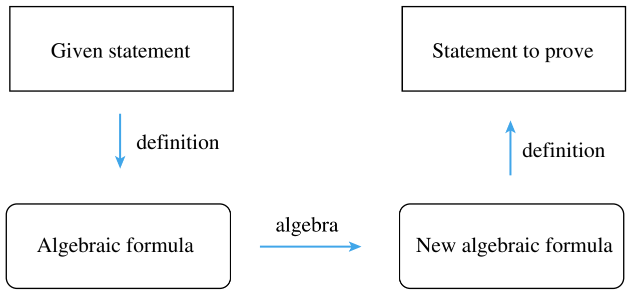


Figure 1.5 The structure of an algebraic proof.

1.5.2 Proof by Contraposition

Sometimes it is hard to see how to get a direct proof started. If you get stuck (and you will), try proving the contrapositive. This is certainly permitted, since the contrapositive of a statement is its logical equivalent. We can state this as another rule of thumb.

Rule of Thumb 1.2 To prove a statement of the form $(\forall x)(P(x) \to Q(x))$, begin your proof with a sentence of the form

Let x be [an element of the domain], and suppose $\neg Q(x)$.

A proof by contraposition is then a sequence of justified conclusions culminating in $\neg P(x)$.

Example 1.21 Suppose x and y are positive real numbers such that the geometric mean $\sqrt{xy}$ is different from the arithmetic mean $\frac{x+y}{2}$. Then $x \neq y$.

Proof (By contraposition.) Let x and y be positive real numbers, and suppose $x = y$. Then

$$\begin{aligned} \sqrt{xy} &= \sqrt{x^2} && \text{since } x = y \\ &= x && \text{since } x \text{ is positive} \\ &= \frac{x+x}{2} && \text{using arithmetic} \\ &= \frac{x+y}{2} && \text{since } x = y \end{aligned}$$

□

Contraposition isn't a radically new proof technique; a proof of a statement by contraposition is just a direct proof of the statement's contrapositive. In Example 1.21, the form of the statement to prove gave a clue that a proof by contraposition would work. If A is the statement "$\sqrt{xy} = \frac{x+y}{2}$" and B is the statement "$x = y$," then the statement to prove has the form $\neg A \to \neg B$. The contrapositive of this statement is $B \to A$, so our proof started with the assumption that $x = y$ and concluded that $\sqrt{xy} = \frac{x+y}{2}$.

For the next example we need some facts from the system of plane geometry that you studied in high school. Henceforth, we'll refer to this type of geometry as *Euclidean* geometry. The following theorem, which we will not prove, is true in Euclidean geometry.

Theorem 1.2 *The sum of the measures of the angles of any triangle equals* 180°.

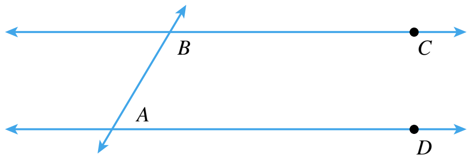
The definition of *parallel* that we used in four-point geometry also works in Euclidean geometry. Although the wording of the following definition is a little different, the content is fundamentally the same.

Definition 1.11 Two lines are parallel if they do not intersect.

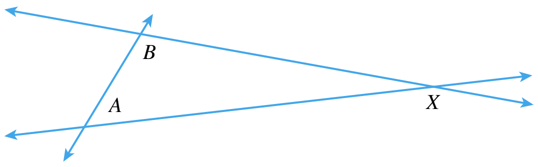
We'll use these two statements in the next example.

Example 1.22 Prove:

> If two lines are cut by a transversal such that a pair of interior angles are supplementary, then the lines are parallel.



Proof (By contraposition.) Suppose we are given two lines cut by a transversal as shown above, and suppose the lines are not parallel. Then, by the definition of parallel lines, the lines intersect. Without loss of generality, suppose they intersect on the right at point X. (If they intersect on the left, the same argument will work.)



By Theorem 1.2, the sum of the angles of $\triangle XAB$ is 180°. Since $\angle X$ has measure greater than 0, the sum of the measures of $\angle A$ and $\angle B$ must be less than 180°, so $\angle A$ and $\angle B$ can't be supplementary. □

1.5.3 Proof by Contradiction

Sometimes even a simple-looking statement can be hard to prove directly, with or without contraposition. In this case, it sometimes helps to try a *proof by contradiction*. The idea is a little counterintuitive. To prove statement A, suppose its negation $\neg A$ is true. Then argue, as in a direct proof, until you reach a statement that you know to be false. You will have established the sequence

$$\neg A \Rightarrow B_1 \Rightarrow B_2 \Rightarrow \cdots \Rightarrow B_n \Rightarrow \mathbf{F}$$

where $\mathbf{F}$ represents a statement that is always false, that is, a contradiction. Taking contrapositives of this chain gives us a sequence

$$A \Leftarrow \neg B_1 \Leftarrow \neg B_2 \Leftarrow \cdots \Leftarrow \neg B_n \Leftarrow \mathbf{T}$$

and since $\mathbf{T}$ is always true (i.e., a tautology) it follows that A is true also. To sum up:

Rule of Thumb 1.3 To prove a statement A by contradiction, begin your proof with the following sentence:

> Suppose, to the contrary, that $\neg A$.

Then argue, as in a direct proof, until you reach a contradiction.

This next example is similar to Example 1.22. In fact, it is a weaker statement, so the proof given in Example 1.22 could also be used to prove it. But it makes a nice example of the contradiction method.

Example 1.23 In Euclidean geometry, prove:

> If two lines share a common perpendicular, then the lines are parallel.

Before stating the proof, notice that this theorem is of the following form.

$$(\forall x)(\forall y)(C(x,y) \rightarrow P(x,y))$$

Here $C(x,y)$ means "x and y share a common perpendicular," and $P(x,y)$ means "$x \parallel y$." You can check that the formal negation of this statement is the following.

$$(\exists x)(\exists y)(C(x,y) \wedge \neg P(x,y))$$

The translation of this last statement is "There exist lines who share a common perpendicular but are not parallel." So we use this statement to start our proof by contradiction.

Proof (By contradiction.) Suppose, to the contrary, that line AB is a common perpendicular to lines AC and BD, and also that AC and BD are not parallel. Then, by Definition 1.11, AC and BD intersect in some point X. But then $\triangle ABX$ has two right angles (and a third angle of nonzero measure), contradicting Theorem 1.2. □

The next results rely on properties of even and odd numbers, so we need to use these definitions in our arguments. Recall:

> **Definition 1.5.** An integer n is even if $n = 2k$ for some integer k.

> **Definition 1.6.** An integer n is odd if $n = 2k+1$ for some integer k.

As we discussed in Section 1.4.1, these definitions alone don't imply that every integer is either even or odd. We'll state this fact as an axiom.[6]

Axiom 1.2 For all integers n, $\neg(n \text{ is even}) \Leftrightarrow (n \text{ is odd})$.

In other words, any integer is either even or odd, but never both. This axiom is the key to proving the following lemma.

Lemma 1.1 Let n be an integer. If n^2 is even, then n is even.

Proof (By contraposition.) Let n be an integer, and suppose n is not even. Then n is odd, by Axiom 1.2. So there is some integer k such that $n = 2k + 1$. Then

$$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$$

and since $(2k^2 + 2k)$ is an integer (by Axiom 1.1), we have shown that n^2 is odd. By Axiom 1.2, n^2 is not even, as required. □

Our final example is a classic proof by contradiction. Recall that a *rational* number is a number that can be written as a/b, where a and b are integers with $b \neq 0$.

Example 1.24 Prove that $\sqrt{2}$ is irrational.

Proof (By contradiction.) Suppose, to the contrary, that $\sqrt{2}$ is rational, so there are integers a and b such that $a/b = \sqrt{2}$, and a and b can be chosen so that

6. In a more rigorous treatment of number theory, this fact could be proved using the division algorithm, which would follow from the well-ordering principle.

the fraction a/b is in lowest terms. Then $a^2/b^2 = 2$, so $a^2 = 2b^2$, that is, a^2 is even. By Lemma 1.1, a is even. Therefore $a = 2k$ for some integer k, so $a^2 = 4k^2$. But now we have $b^2 = a^2/2 = 2k^2$, so b^2 is even, and therefore, by the lemma again, b is even as well. We have shown that a and b are both even, which contradicts the assumption that a/b is in lowest terms. □

Exercises 1.5

1. Consider the following statement.

   For all integers x, if $4 \mid x$, then x is even.

   (a) Write this statement in predicate logic in the domain of integers. Say what your predicates are.
   (b) Apply Rule of Thumb 1.1 to write down the first sentence of a direct proof of this statement.
   (c) Use Definition 1.10 to translate your supposition in part (b) into algebra.
   (d) Finish the proof of the statement.

2. Give a direct proof:

   Let a, b, and c be integers. If $a \mid b$ and $a \mid c$, then $a \mid (b \cdot c)$.

   Remember that you must use the definition of $\mid$ in your proof.

3. Prove: Let a, b, and c be integers. If $(a \cdot b) \mid c$, then $a \mid c$.

4. Give a direct proof.

   Let a, b, and c be integers. If $a \mid b$ and $b \mid c$, then $a \mid c$.

5. Give a direct proof of the following statement in Euclidean geometry. Cite any theorems you use.

   The sum of the measures of the angles of a parallelogram is 360°.

6. Prove:

   For all integers n, if n^2 is odd, then n is odd.

   Use a proof by contraposition, as in Lemma 1.1.

7. Prove the following statement by contraposition.

   Let x be an integer. If $x^2 + x + 1$ is even, then x is odd.

   Make sure that your proof makes appropriate use of Definitions 1.5 and 1.6.

8. Prove that the sum of two even integers is even.

9. Prove that the sum of an even integer and an odd integer is odd.

10. Prove that the sum of two odd integers is even.

11. Write a proof by contradiction of the following.

    Let x and y be integers. If x and y satisfy the equation

    $$3x + 5y = 153$$

    then at least one of x and y is odd.

12. Prove the following statement in Euclidean geometry. Use a proof by contradiction.

    A triangle cannot have more than one obtuse angle.

13. Let "$x \nmid y$" denote "x does not divide y." Prove by any method.

    Let a and b be integers. If $5 \nmid ab$, then $5 \nmid a$ and $5 \nmid b$.

14. Consider the following definition.

    Definition. An integer n is *sane* if $3 \mid (n^2 + 2n)$.

    (a) Give a counterexample to the following: All odd integers are sane.
    (b) Give a direct proof of the following: If $3 \mid n$, then n is sane.
    (c) Prove by contradiction: If $n = 3j + 2$ for some integer j, then n is not sane.

15. Prove that the rational numbers are closed under multiplication. That is, prove that, if a and b are rational numbers, then $a \cdot b$ is a rational number.

16. Prove that the rational numbers are closed under addition.

17. Prove: Let x and y be real numbers. If x is rational and y is irrational, then $x \cdot y$ is irrational.

18. Prove: Let x and y be real numbers. If x is rational and y is irrational, then $x + y$ is irrational.

19. Consider the following definition.

    Definition. A integer n is *frumpable* if $n^2 + 2n$ is odd.

    Prove: All frumpable numbers are odd.

20. Recall the Badda-Bing axiomatic system of Example 1.17. Prove:

    If q and r are distinct bings, both of which are hit by baddas x and y, then $x = y$.

21. Two common axioms for geometry are as follows. The undefined terms are "point," "line," and "is on."

    1. For every pair of points x and y, there is a unique line such that x is on l and y is on l.
    2. Given a line l and a point x that is not on l, there is a unique line m such that x is on m and no point on l is also on m.

    Recall that two lines l and m are *parallel* if there is no point on both l and m. In this case we write $l \parallel m$. Use this definition along with the above two axioms to prove the following.

    Let l, m, and n be distinct lines. If $l \parallel m$ and $m \parallel n$, then $l \parallel n$.

22. The following axioms characterize *projective geometry*. The undefined terms are "point," "line," and "is on."

    1. For every pair of points x and y, there is a unique line such that x is on l and y is on l.
    2. For every pair of lines l and m, there is a point x on both l and m.
    3. There are (at least) four distinct points, no three of which are on the same line.

    Prove the following statements in projective geometry.

    (a) There are no parallel lines.

    (b) For every pair of lines l and m, there is exactly one point x on both l and m.

    (c) There are (at least) four distinct lines such that no point is on three of them.

Chapter 2

Relational Thinking

Most quantitative problems involve several different interrelated objects: market forces determine the price of a commodity, steps in a manufacturing process depend on other steps, virus-infected computers can slow network traffic. In order to analyze these relationships, it helps to think mathematically about them.

In this chapter we will explore different ways that the elements of a set can be related to each other or to the elements of another set. These relationships can be described by mathematical objects such as functions, relations, and graphs. Our goal is to develop the ability to see mathematical relationships between objects, which in turn will enable us to apply tools from discrete mathematics.

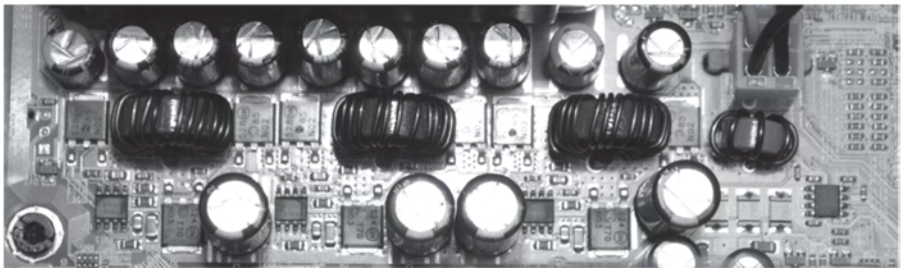


Figure 2.1 A computer's circuit board contains an intricate system of mathematical relationships. Concepts such as connectivity, interdependence, and modularity can be expressed in the language of mathematics.

2.1 Graphs

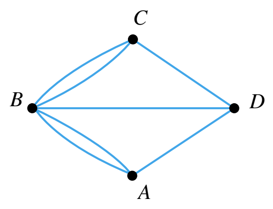
When faced with a difficult problem in mathematics, it often helps to draw a picture. If the problem involves a discrete collection of interrelated objects, it is natural to sketch the objects and draw lines between them to indicate the relationships. A *graph* is the mathematical version of such a sketch. In this section we will study a few basic definitions about graphs, and then explore some ways to use graphs to model mathematical relationships. Our approach here will be informal; later in the chapter we will look at graphs from a more rigorous point of view.

2.1.1 Edges and Vertices

In Section 2.6, we will give a mathematical definition of a graph, and we will prove several theorems about graphs. For now, however, just think of a graph informally as a diagram of dots, called *vertices*, connected by lines or curves, called *edges*. The edges of a graph may have arrows on them; in this case, the graph is called a *directed* graph. A graph without arrows on the edges is called an *undirected* graph.

When we draw a graph, it doesn't really matter where we put the vertices or whether we draw the edges as curved or straight—the thing that matters is whether or not two given vertices are connected by an edge (or edges).

Example 2.1 The Pregel River divided the Prussian city of Königsberg (now Kaliningrad, Russia) into four sections, as shown in Figure 2.2. These sections were connected by seven bridges. If we draw a vertex for each land mass and an edge for each bridge, we can represent the city as the following graph.



Notice that there are some double edges; these reflect the presence of two bridges connecting the same pair of land masses.

The bridges of Königsberg inspired the great eighteenth-century mathematician Leonhard Euler to think about the kinds of relationships that graphs express. In March of 1736, Euler wrote the following to a colleague:

> A problem was posed to me about an island in the city of Königsberg, surrounded by a river spanned by seven bridges, and I was asked whether someone could traverse the separate bridges in a connected

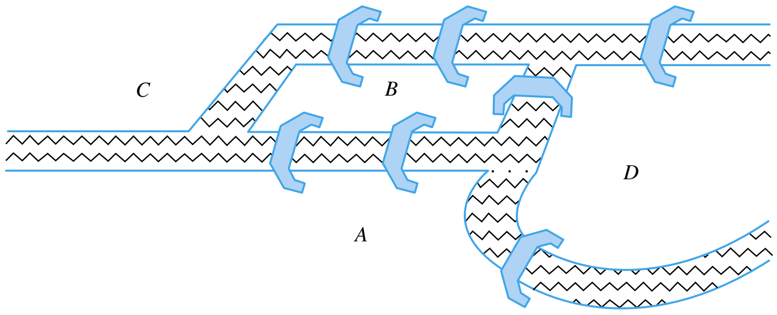


Figure 2.2 The bridges of Königsberg.

walk in such a way that each bridge is crossed only once. I was informed that hitherto no one had demonstrated the possibility of doing this, or shown that it is impossible. This question is so banal, but seemed to me worthy of attention in that geometry, nor algebra, nor even the art of counting was sufficient to solve it. [10]

Although Euler did not use modern notation and terminology, his paper on the Königsberg bridges is widely regarded as the start of modern graph theory. [16]

2.1.2 Terminology

In order to work with graphs, it helps to define some terms. The *degree* of a vertex is the number of times an edge touches it. This is different than the number of edges touching it, because an edge may form a *loop*, as in Figure 2.3. In graph H, vertex x has degree 5. In a directed graph, we can speak of the *indegree* (the number of edges coming in to the vertex) and the *outdegree* (the number of edges going out). In Figure 2.3, vertex a of graph G has indegree 1 and outdegree 2.

A *path* in a graph is a sequence

$$v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$$

of vertices v_i and edges e_j such that edge e_i connects vertices v_{i-1} and v_i. Here $n \geq 1$. A *circuit* is a path that ends where it begins, so $v_0 = v_n$. An undirected graph is *connected* if there is a path connecting any two pairs of vertices. A directed graph is connected if the underlying undirected graph is.

In Figure 2.3, graph H is connected, and therefore so is graph G. There is a circuit in graph H going around the large quadrilateral: start at vertex v, follow the edges clockwise through vertices w, x, and z, and return to vertex v. However, the corresponding sequence is not a circuit in graph G because the edge from a to e goes the wrong direction.

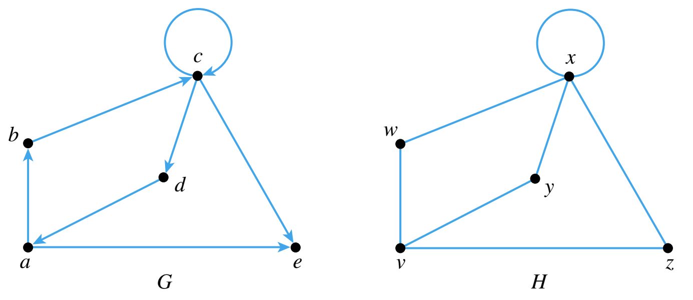


Figure 2.3 Graph H is the underlying undirected graph of the directed graph G.

Why do we need all these terms? One reason is that terminology makes it easier to make *precise* descriptions of the relationships that graphs define. Let's take another look at the bridges of Königsberg. In his 1736 paper, Euler made the following observations (quoted in [16]).

1. The number of bridges written next to the letters A, B, C, etc. together add up to twice the total number of bridges. The reason for this is that, in the calculation where every bridge leading to a given area is counted, each bridge is counted twice, once for each of the two areas which it joins.

2. If there are more than two areas to which an odd number of bridges lead, then such a journey is impossible.

3. If, however, the number of bridges is odd for exactly two areas, then the journey is possible if it starts in either of these two areas.

4. If, finally, there are no areas to which an odd number of bridges lead, then the required journey can be accomplished starting from any area.

We can use modern terminology to restate these observations. Let's define an *Euler path* (resp. *circuit*) as a path (resp. circuit) that uses every edge of the graph exactly once.

1. In any graph, the sum of the degrees of the vertices equals twice the number of edges.

2. If a graph has more than two vertices of odd degree, it does not have an Euler path.

3. If a connected graph has exactly two vertices, v and w, of odd degree, then there is an Euler path from v to w.

4. If all the vertices of a connected graph have even degree, then the graph has an Euler circuit.

The vertices A, B, C, and D of the graph of Example 2.1 have degrees 3, 5, 3, and 3, respectively. Therefore, (if we trust Euler), this graph does not have an Euler path. Note that we haven't yet given rigorous proofs for any of Euler's observations, but we have managed to state them a little more clearly and concisely.

2.1.3 Modeling Relationships with Graphs

In addition to the "banal" problem of walking over bridges, there are many other applications of graphs for problems involving relationships.

Example 2.2 A college registrar would like to schedule the following courses in as few time slots as possible: Physics, Computer Science, Chemistry, Calculus, Discrete Math, Biology, and Psychology. However, from previous experience, the following pairs of classes always have students in common, so they can't be scheduled in the same time slot:

Physics and Computer Science
Physics and Chemistry
Calculus and Chemistry
Calculus and Physics
Calculus and Computer Science
Calculus and Discrete Math
Calculus and Biology
Discrete Math and Computer Science
Discrete Math and Biology
Psychology and Biology
Psychology and Chemistry

What is the fewest number of time slots needed to schedule all these classes without conflicts?

Solution: The natural way to model this problem with a graph is to make each course into a vertex and connect any two vertices that represent courses that cannot be scheduled in the same time slot. Figure 2.4 shows this graph.

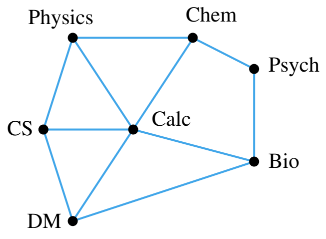


Figure 2.4 Graph for Example 2.2.

Let's start with Calculus: call its time slot A. None of the courses that share an edge with Calculus can be in time slot A, so pick one, say, Computer Science, and call its time slot B. Now Physics shares edges with both Calculus and Computer Science, so this forces us to use a third time slot, C, for Physics.

We should note at this point that all of our choices have been forced so far—we are going to need at least three time slots. The question remains whether we need more than three slots. Notice that we can assign Chemistry to slot B (since it doesn't share an edge with Computer Science) and we can assign Discrete Math to slot C (since it doesn't conflict with Physics). This allows us to use B for Biology and either A or C for Psychology, showing that three time slots are sufficient. $\diamondsuit$

This scheduling problem is an example of graph coloring. A *coloring* for a graph is an assignment of different values (colors) to each vertex such that no two vertices of the same color share an edge. In Example 2.2, the "colors" were the time slots A, B, and C.

We call a graph *planar* if it can be written down (on a "planar" sheet of paper) without any edges crossing each other. The graph in Figure 2.4 is an example of a planar graph. The famous Four Color Theorem states that any planar graph can be colored with, at most, four colors.[1] A consequence of the Four Color Theorem is that cartographers need only four colors of ink: it is always possible to color the regions of a planar map (e.g., the states in a map of the United States) with, at most, four colors, so that no two adjacent regions have the same color.

Example 2.3 Several departments around campus have wireless access points (WAPs), but problems arise if two WAPs within 200 feet of each other are operating on the same frequency. Suppose the departments with WAPs are situated as follows.

Department:	Is within 200 feet of departments:
Math	Physics, Psychology, Chemistry, Sociology
Sociology	History, English, Economics, Math, Chemistry, Psychology
Physics	Math, Chemistry
Psychology	Math, Chemistry, Sociology, Economics
History	Sociology, English
English	Economics, Sociology, History
Economics	English, Sociology, Psychology
Chemistry	Math, Psychology, Sociology, Physics

1. Although this result is very simple to state, the only known proofs to date are extremely complicated.

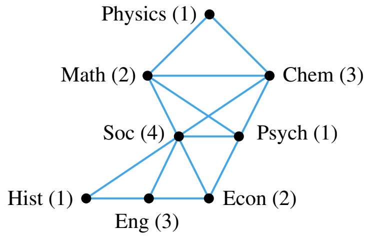


Figure 2.5 Graph for Example 2.3. Colors/frequencies: 1, 2, 3, 4.

What is the fewest number of frequencies needed? How could these frequencies be assigned to departments?

Solution: Figure 2.5 shows a graph model for this situation, along with a possible coloring. We leave it as an exercise to check that four colors (frequencies) are needed. ◇

In Example 2.3, we modeled the positions of the departments by drawing an edge between any two departments that were "close," i.e., within 200 yards of each other. If we had more specific data, we could make our model say *how* close each department is to the others by putting numbers on each edge indicating the distance between two departments. A graph with numerical values, or *weights*, on its edges is called a *network*. Networks can be directed or undirected, depending on what they are intended to model.

Example 2.4 The following table lists some pairwise driving distances (in miles) on selected routes between some California cities.

	B	E	F	L	N	S
Barstow						
Eureka						
Fresno	245	450				
Los Angeles	115	645	220			
Needles	145		385	260		
San Diego	175			125	320	

Figure 2.6 shows the graph for this network. Writing the distance data in this format makes it easier to answer certain questions. How far is it from San Diego to Fresno (using these routes)? If we travel from Needles to Fresno via Barstow, how much longer will it take?

The above examples show how versatile graph models are; often it helps to add extra structure to a graph (for example, colors or weights) to suit a partic-

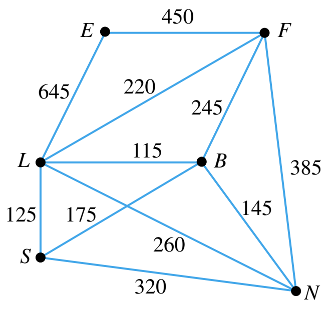


Figure 2.6 A network showing mileage between cities.

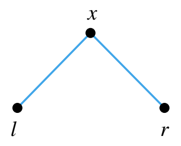
ular application. The next example illustrates another way to add information to a graph model by specifying how the graph is drawn.

Example 2.5 The custom dictionary in a spell-checker keeps track of words that you don't want flagged as misspellings but aren't in the standard dictionary. These words get added one at a time, in no particular order, but it is necessary to keep them organized so that searching the list is easy. Suppose your custom dictionary contains the following words:

macchiato, poset, phat, complexify, jazzed, sheafify, clueless

What is an efficient way to organize this data?

One possible organizational structure is a graph model called a *binary search tree*. The rules for constructing a binary search tree are simple. Start with an item (chosen arbitrarily) at the top of the tree. This is the *root* node of the tree. The root has (at most) two edges touching it, one going down to the right, and one going down to the left. These are the *children* of the root node. In fact, every node in a binary search tree can have up to two children, one on the right, and one on the left. The only condition is that the right child (along with its "descendants") must come after its parent in alphabetical (or numerical) order, and the left child and its descendants must come before its parent. For example, in the following tree, x has right child r and left child l, so this data must have the order l, x, r.



Let's place the data from Example 2.5 into a binary search tree. Start with "macchiato" at the root of the tree. The next word, "poset," comes after "macchiato" in alphabetical order, so "poset" becomes the right child of "macchiato." We would then place the next word, "phat," to the right of "macchiato," but since that spot is taken, and since "phat" precedes "poset" in alphabetical order, we make "phat" the left child of "poset." We continue in this manner, finding the new position for each new word by moving down the tree. At each branch we move right or left, depending on where the new word falls in the alphabet. Figure 2.7 shows the first few steps of this process, along with the final result.

The reason this graph is called a "search" tree is that it makes finding an element in the tree easy. If you are looking for a word in a binary search tree, you go through the same procedure as if you were adding the word to the tree; if you don't come across the word, it isn't there.

For large data sets, this process goes very quickly, since you don't need to look at every element. For example, to find "poset" in the tree in Figure 2.7, we only have to look at two words. To establish that "iPod" is not in the tree, we only need to check three. Each comparison moves the search one level down the tree, and each level contains twice the number of elements as the level before.

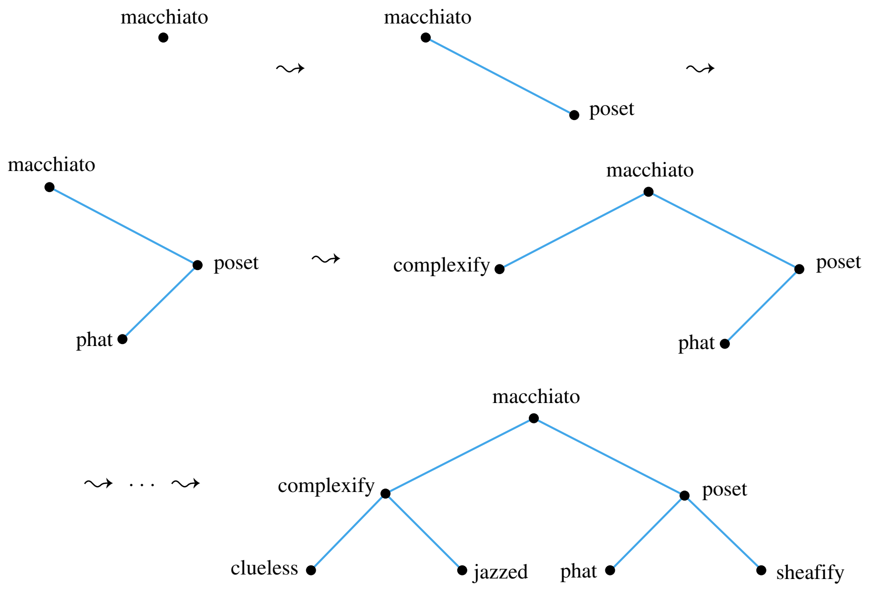


Figure 2.7 Constructing the binary search tree for Example 2.5.

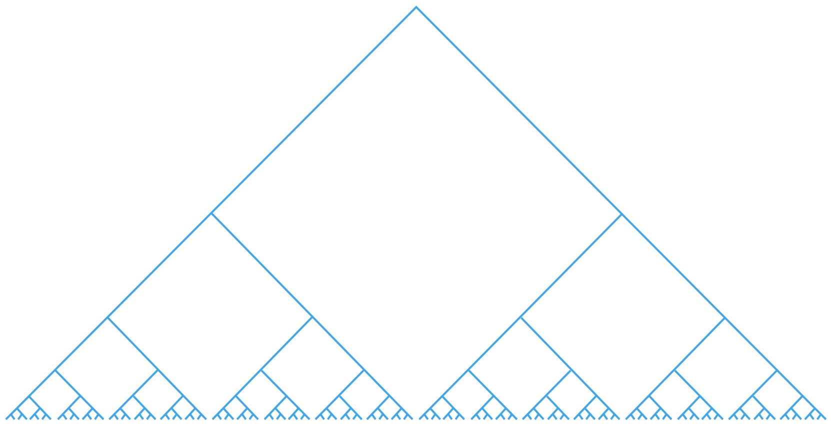


Figure 2.8 A balanced binary search tree with 255 nodes requires, at most, eight comparisons to search it completely.

So, for example, a balanced binary search tree with

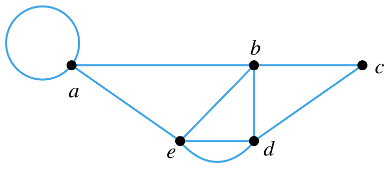
$$255 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128$$

elements requires, at most, eight comparisons to search it completely. See Figure 2.8.

Note that a binary search tree contains more information than its graph structure conveys. The branches of the tree always go downward, and it matters whether they go to the left or to the right. Technically, a binary search tree is a directed graph, with all the edges pointing downward, away from the root. But most books (including this one) omit the arrows.

Exercises 2.1

1. Consider the following undirected graph.

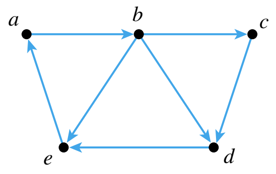


(a) How many edges are there in this graph?

(b) Give the degree of each vertex.

(c) Do these numbers agree with Euler's first observation?

2. Consider the following directed graph.



(a) Give the indegree of each vertex.

(b) Give the outdegree of each vertex.

(c) Compute sum of the indegrees and the sum of the outdegrees. What do you notice?

3. Draw a connected, undirected graph with seven vertices and no circuits. How many edges does it have?

4. Draw an undirected graph with six vertices, each of degree 3, such that the graph is ...

(a) connected.

(b) not connected.

5. A graph is called *simple* if it has no multiple edges or loops. (The graphs in Figures 2.4, 2.5, and 2.6 are simple, but the graphs in Example 2.1 and Figure 2.3 are not simple.) Draw five different connected, simple, undirected graphs with four vertices.

6. An undirected graph is called *complete* if every vertex shares an edge with every other vertex. Draw a complete graph on five vertices. How many edges does it have?

7. Figure 2.9 shows the bridge configuration of modern day Kaliningrad. Represent the four land masses and seven bridges as an undirected graph. Is this graph the same as the one in Example 2.1? Why or why not? Is it possible to travel an Euler circuit on this graph?

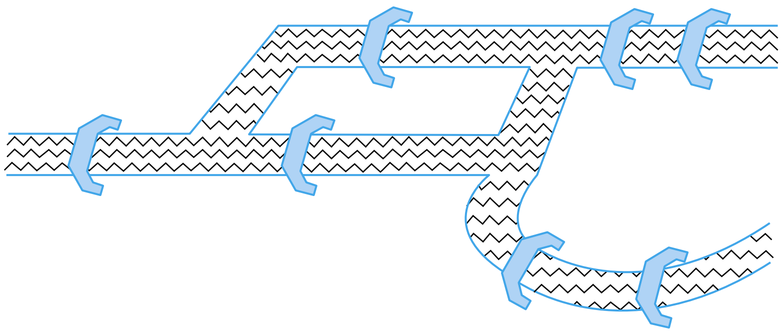
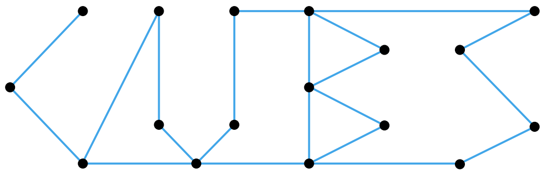


Figure 2.9 Kaliningrad, Russia.

8. Does the following graph have an Euler path? Why or why not?



9. Think of the Internet as one big graph, where each web page is a vertex and each link is an edge.

   (a) Is this a directed graph? Why or why not?

   (b) Is this graph connected? Why or why not?

   (c) Is this graph complete? Why or why not?

   (d) Is this graph simple? Why or why not?

   (e) For a given web page p, what does the outdegree of p represent?

   (f) For a given web page p, what does the indegree of p represent?

10. Find a map of the United States. Draw a graph whose vertices represent the states of Illinois, Missouri, Tennessee, Virginia, West Virginia, Ohio, Indiana, and Kentucky. Draw an edge between any two vertices whose states share a common border. Explain why it takes four colors to color this graph (and hence also to distinguish these states on a map).

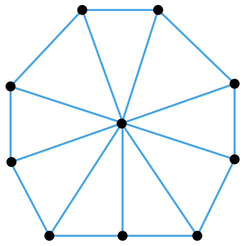
11. As in Exercise 10, draw a graph representing the border relationships for the states of Washington, Oregon, Idaho, California, Nevada, and Utah. What is the fewest number of colors needed to color this graph? Justify your answer.

12. Represent the 13 regions in South America (12 countries plus the territory of French Guiana) as a graph. What is the fewest number of colors needed to color this graph?

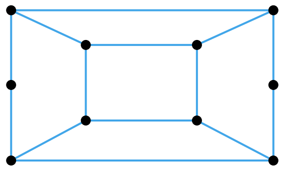
13. Using as few groups as possible, put the words `fish`, `sit`, `stay`, `play`, `diet`, `tree`, `duck`, `dog`, and `hen` into groups such that none of the words in a group have any letters in common. Use a graph model and graph coloring. Justify your answer: explain why your grouping uses the fewest groups possible.

14. Give an example of a graph that requires five colors to make a valid coloring. (Note that, by the Four Color Theorem, your example cannot be planar.)

15. Compute the minimal number of colors needed to color the following graph. Show that your answer is big enough (by describing a coloring), and explain why the graph cannot be colored in fewer colors.



16. Color the vertices of the following graph so that no vertices of the same color share an edge. Use as few colors as possible. Explain why the graph cannot be colored using fewer colors. Be specific.



17. A round-robin tournament among four teams—Canadiens, Canucks, Flames, and Oilers—has the following results: Canucks defeat Canadiens; Canucks defeat Flames; Canucks defeat Oilers; Canadiens defeat Oilers; Flames defeat Canadiens; Oilers defeat Flames.

    (a) Model these results with a directed graph, where each vertex represents a team and each edge represents a game, pointing from the winner to the loser.

(b) Find a circuit in this graph.

(c) Explain why the existence of a circuit in such a graph makes it hard to rank the teams from best to worst.

18. Consider the network in Example 2.4. Plan a round trip starting and ending at San Diego that visits all the other cities in as few miles as possible. In other words, find a circuit that contains every vertex and has minimal weight.

19. Construct a directed network whose vertices represent the numbers

$$11, 12, 13, 15, 17$$

and whose weights tell how much you must add to get from one vertex to another. Include only edges of positive weight.

20. A car rental company has three locations in Mexico City: the International Airport, Oficina Vallejo, and Downtown. Customers can drop off their vehicles at any of these locations. Based on prior experience, the company expects that, at the end of each day, 40% of the cars that begin the day at the Airport will end up Downtown, 50% will return to the Airport, and 10% will be at Oficina Vallejo. Similarly, 60% of the Oficina Vallejo cars will end up Downtown, with 30% returning to Oficina Vallejo and 10% to the Airport. Finally, 30% of the Downtown cars will end up at each of the other locations, with 40% staying at the Downtown location.

Model this situation with a directed network. If the company starts with all of its cars at the Airport, how will the cars be distributed after two days of rentals?

21. Consider the following list of numbers.

$$123, 684, 121, 511, 602, 50, 43$$

(a) Place the numbers, in the order given, into a binary search tree.

(b) The *height* of a binary search tree is the maximum number of edges you have to go through to reach the bottom of the tree, starting at the root. What is the height of the tree in part (a)?

(c) Reorder the numbers so that when they are put into a binary search tree, the height of the resulting tree is less than the height of the tree in part (a). Give both your new list and the search tree it produces.

22. Put the words

```
Cheddar  Swiss  Brie  Panela  Stilton  Mozzarella  Gouda
```

into a binary search tree with the smallest height possible.

23. Put the following words into a binary search tree. Add them to the tree in the order given.

```
i   will   not   eat   them   with   a   fox
```

24. Sociologists use graphs to model social relationships. A *social network* (not to be confused with a network) is a graph where the nodes represent "actors" (e.g., people, companies) and the edges represent relationships, or "ties," between actors (e.g., friendships, business partnerships). Consider the social network in Figure 2.10.

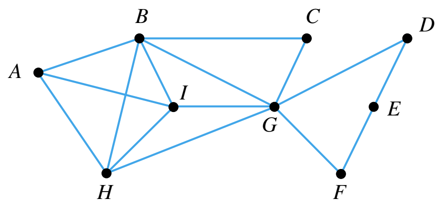


Figure 2.10 Social network for Exercise 24.

   (a) A *clique* in a social network is a group of actors who all have ties to each other. What is the largest clique in the social network in Figure 2.10?

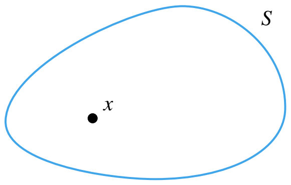
   (b) If you had to choose the most important actor in this social network, who would you pick? Why?

   (c) Suppose the actors represent people and the ties represent acquaintances. If the people in this social network continue to interact, which two (currently unacquainted) actors would you most expect to become acquainted? Which two actors are the least likely to become acquainted? Why?

2.2 Sets

The applications from the previous section should convince you that graphs are a powerful mathematical tool. However, our viewpoint wasn't very rigorous; in order to prove useful theorems about graphs, we need to consider some more mathematical structures that describe relationships.

The simplest way to describe a collection of related objects is as a *set*. Think of the set S as a container where an object x is something that S contains.



We write $x \in S$ to denote that x is contained in S. We also say that "x is a member of S," "x is an element of S," or more simply, "x is in S."

2.2.1 Membership and Containment

We can describe examples of sets by listing the elements in the set or by describing the properties that an element in the set has. To say that set S consists of the elements $x_1, x_2, \dots, x_n$, we write

$$S = \{x_1, x_2, \dots, x_n\}.$$

Suppose there is some property p that some of the elements of a set S have. We can describe the set of all elements of S that have property p as

$$\{x \in S \mid x \text{ has property } p\}.$$

This is sometimes called "set builder" notation, because it explains how to build a list of all the elements of a set.

Example 2.6 Let $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Then $2 \in A$ and $9 \notin A$. If

$$B = \{x \in A \mid x \text{ is odd}\},$$

then the elements of B are $1, 3, 5, 7$.

Example 2.7 There are some common sets that have specific names. The set of integers is denoted by $\mathbf{Z}$, and the set of positive integers, or *natural numbers* is written as $\mathbf{N}$. Note that $0 \in \mathbf{Z}$, but $0 \notin \mathbf{N}$. We use $\mathbf{R}$ for the set of real numbers, and $\mathbf{Q}$ for the set of rational numbers.[2]

Example 2.8 Let P be the set of all polygons. So P contains all triangles, squares, pentagons, etc. If c is a circle, then $c \notin P$. We could describe the set H of all hexagons in set builder notation as

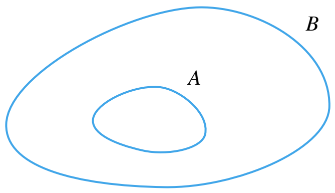
$$H = \{x \in P \mid x \text{ has six sides}\}.$$

2. Recall that a rational number is a number that can be written as a fraction a/b, where a and b are integers.

The language of sets is convenient for describing groups of objects that are related by some common property. Often some property implies another; for example, all integers are real numbers. In terms of sets, this means that the set $\mathbf{Z}$ is contained in the set $\mathbf{R}$. In general, the predicate logic statement that

$$(\forall x)(x \in A \to x \in B) \tag{2.2.1}$$

is written as $A \subseteq B$, and we say "A is contained in B," or "A is a subset of B," or "B contains A." We can express this relationship pictorially using the following *Venn diagram*.



Example 2.9 In Example 2.6, $B \subseteq A$. We have noted that $\mathbf{Z} \subseteq \mathbf{R}$ in Example 2.7, and we could also say that $\mathbf{N} \subseteq \mathbf{Z}$, $\mathbf{Z} \subseteq \mathbf{Q}$, and $\mathbf{Q} \subseteq \mathbf{R}$. This chain of relationships could also be written as

$$\mathbf{N} \subseteq \mathbf{Z} \subseteq \mathbf{Q} \subseteq \mathbf{R}.$$

Example 2.10 In Example 2.8, $H \subseteq P$, but $P \not\subseteq H$, because not every polygon is a hexagon.

Example 2.11 The *empty set* $\emptyset$ is the set that contains no elements. Therefore, the empty set is a subset of any set, that is, $\emptyset \subseteq X$ for all X. This is because the statement $x \in \emptyset$ is false for any x, so the implication

$$(\forall x)(x \in \emptyset \to x \in X)$$

must be true. (See the truth table for the "$\to$" connective on page 6.)

2.2.2 New Sets from Old

Sets describe relationships, but the language of sets can also describe the logic of how things are related. We have seen one example of this already: statement 2.2.1 shows how to interpret the $\subseteq$ symbol in terms of a predicate logic statement containing the $\to$ connective; the relationship of containment has the logic of an implication. The connectives $\vee$, $\wedge$, $\neg$, and $\leftrightarrow$ also have set-theoretic counterparts.

The *union* $A \cup B$ of two sets A and B is the set containing all the elements of both A and B put together. An element is in the union of two sets A and B if the element is A, or B, or both. In set builder notation,

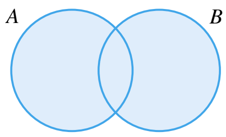
$$A \cup B = \{x \mid (x \in A) \vee (x \in B)\}.$$

This translates to the following equivalence rule in predicate logic.

$$(\forall x)[x \in A \cup B \Leftrightarrow (x \in A) \vee (x \in B)]$$

The statement $x \in A \cup B$ is logically equivalent to the statement $(x \in A) \vee (x \in B)$. This fact is important when writing proofs; one of these statements can always be replaced by the other.

Think of "union" as the set-theoretic counterpart of the logical "or." In the following Venn diagram, $A \cup B$ is the shaded area:



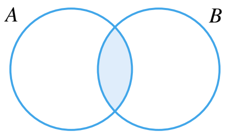
The *intersection* $A \cap B$ is the set containing all the elements that A and B have in common. In order for an element to be in the intersection of A and B, the element must be in both sets. Therefore we write the intersection as

$$A \cap B = \{x \mid (x \in A) \wedge (x \in B)\}$$

in set builder notation. This implies the following logical equivalence for all x.

$$x \in A \cap B \Leftrightarrow (x \in A) \wedge (x \in B)$$

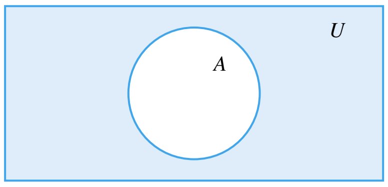
In the following Venn diagram, the shaded area represents $A \cap B$:



Implicit in the above predicate logic statements is a domain, or *universal set* U, of which every set is a subset. If we know what the domain U is, we can speak of the *complement* A' of A, which is the set

$$A' = \{x \in U \mid x \notin A\}.$$

We usually draw U as a big rectangle, so the shaded area below represents A'.



Note that we could also write $A' = \{x \in U \mid \neg(x \in A)\}$ to make the use of the $\neg$ connective explicit.

We say that two sets are equal if they have the same elements. So the statement "$A = B$" translates to the following statement in predicate logic:

$$(\forall x)(x \in A \leftrightarrow x \in B).$$

This translation is important when it comes to proving that two sets are equal. A proof that $A = B$ usually consists of two direct proofs: given $x \in A$, prove that $x \in B$, and conversely, given $x \in B$, prove that $x \in A$. In other words, showing $A = B$ amounts to showing that $A \subseteq B$ and $B \subseteq A$.

Example 2.12 Let the following sets be given.

$$\begin{aligned} X &= \{n \in \mathbf{Z} \mid n = 2k \text{ for some odd integer } k\} \\ F &= \{n \in \mathbf{Z} \mid n = 4k \text{ for some integer } k\} \\ E &= \{n \in \mathbf{Z} \mid n \text{ is even}\} \end{aligned}$$

1. Prove that $F \subseteq E$.
2. Prove that $X = E \cap F'$.

Proof of 1 Let $x \in F$. By the definition of F, $x = 4k$ for some integer k. We can write this equation as $x = 2(2k)$, so x is even by Definition 1.5. Therefore $x \in E$. □

Proof of 2 (We first show that $X \subseteq E \cap F'$.) Let $x \in X$. Then $x = 2k$ for some odd integer k, so x is even. Therefore $x \in E$. Suppose, to the contrary, that $x \in F$. Then $x = 4j$ for some integer j. This implies that $2k = 4j$, or $k = 2j$, which contradicts that k is odd. Therefore $x \in F'$. Since $x \in E$ and $x \in F'$, we have shown that $x \in E \cap F'$.

(We now show that $E \cap F' \subseteq X$.) Suppose $x \in E \cap F'$, so $x \in E$ and $x \in F'$. Since $x \in E$, $x = 2k$ for some integer k. Suppose, to the contrary, that k is even. Then $k = 2l$ for some integer l, so $x = 2(2l) = 4l$. But this contradicts that $x \in F'$. Therefore k must be odd. This establishes that $x \in X$. □

These proofs illustrate how to translate back and forth between the language of sets and the language of logic. For example, the statement "$x \in E \cap F'$" was translated as "$x \in E$ and $x \in F'$." In general, any statement involving sets and the symbols $\cap$, $\cup$, $\subseteq$, $=$, and $'$ translates into a logical statement using the connectives $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and $\neg$, respectively. So all the work we did in propositional and predicate logic (Sections 1.2 and 1.3) will now pay off when dealing with sets.

Think of the symbols $\cap$, $\cup$, $\subseteq$, $=$, and $'$ as tools for making new sets from old ones. For example, given two sets, A and B, we can build a new set, $A \cap B$, consisting of all the elements the two sets have have in common. We can also take two sets A and B and form their *Cartesian product* $A \times B$. The Cartesian product is just the set of all ordered pairs where the first item comes from the first set and the second item comes from the second set. Formally,

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}.$$

We can also have ordered triples, quadruples, and so on. The set

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

is the set of ordered n-tuples where the ith item comes from set A_i.

Two ordered pairs are equal if and only if their corresponding parts are equal. In other words,

$$(a, b) = (c, d) \Leftrightarrow a = c \text{ and } b = d.$$

You are already quite familiar with the Cartesian product $\mathbf{R} \times \mathbf{R}$, the set of all ordered pairs of real numbers. This set is the usual Cartesian plane from high school algebra.

One last construction that is sometimes useful is the *power set* $\mathcal{P}(S)$ of the set S. The power set of S is the set of all subsets of S:

$$\mathcal{P}(S) = \{X \mid X \subseteq S\}.$$

Note that the empty set $\emptyset$ is a member of $\mathcal{P}(S)$, no matter what S is, because the empty set is a subset of any set.

Example 2.13 Suppose you want to form a study group with some of the other students in your class. If S is the set of all students in your class, then $\mathcal{P}(S)$ is the set of all possible study groups you could form. (The empty set would represent the decision not to form a study group at all!)

Example 2.14 Let $A = \{1, 2, 3, 4, 5\}$, $B = \{4, 5, 6, 7, 8\}$, and suppose the universal set is $U = \{1, 2, \ldots 10\}$. Then

$$\begin{aligned}
A \cup B &= \{1, 2, \ldots, 8\} \\
A \cap B &= \{4, 5\} \\
B' &= \{1, 2, 3, 9, 10\} \\
(A \cap B) \times A &= \{(4,1), (4,2), (4,3), (4,4), (4,5), (5,1), (5,2), (5,3), (5,4), (5,5)\} \\
\mathcal{P}(A \cap B) &= \{\emptyset, \{4\}, \{5\}, \{4, 5\}\}
\end{aligned}$$

Notice that, while A and B are sets of numbers, the sets you construct from A and B may have different kinds of elements. For example, $(A \cap B) \times A$ is a set of ordered pairs, and $\mathcal{P}(A \cap B)$ is a set of sets.

2.2.3 Identities

Since there is such a direct relationship between logic and set theory, we can write down many facts about sets that follow immediately from things we know from our study of logic. For example, the addition inference rule

$$p \Rightarrow p \vee q$$

allows us to prove the identity

$$A \subseteq A \cup B$$

in set theory.

Proof that $A \subseteq A \cup B$ Let $x \in A$. By the addition rule, $(x \in A) \vee (x \in B)$. By the definition of set union, $x \in A \cup B$, as required. $\square$

De Morgan's laws for sets follow from the equivalence rule of the same name.

Theorem 2.1 *Let A and B be sets. Then*

1. $(A \cup B)' = A' \cap B'$
2. $(A \cap B)' = A' \cup B'$

The proof of this theorem illustrates how to prove set equalities.

Proof We will prove part 1. (Part 2 is similar.) Let $x \in (A \cup B)'$. In other words, if $P(x)$ is the statement "$x \in A$" and $Q(x)$ is the statement "$x \in B$," we are starting with the assumption $\neg(P(x) \vee Q(x))$. By De Morgan's laws for propositional logic, this is equivalent to $\neg P(x) \wedge \neg Q(x)$, which, in the language of set theory, is the same as $x \in A' \cap B'$. We have shown that

$$x \in (A \cup B)' \quad \Leftrightarrow \quad x \in A' \cap B',$$

hence the sets $(A \cup B)'$ and $A' \cap B'$ are equal. $\square$

Take a look back at this proof, and notice two things. First, the proof establishes an "if and only if" claim, since all of its deductions are based on logical equivalences. Second, the proof illustrates that the way to show an equality $U = V$ of sets is to show that $x \in U \Leftrightarrow x \in V$.

The proof of Theorem 2.1 consists of a simple chain of equivalences. When the structure of a proof is this basic, we might opt to write it in the following format.

Proof of Theorem 2.1, "⇔" version We will prove part 1. (Part 2 is similar.)

$x \in (A \cup B)'$	$\Leftrightarrow$	$\neg(x \in A \cup B)$	Definition of $'$
	$\Leftrightarrow$	$\neg[(x \in A) \vee (x \in B)]$	Definition of $\cup$
	$\Leftrightarrow$	$\neg(x \in A) \wedge \neg(x \in B)$	De Morgan's law (logic)
	$\Leftrightarrow$	$(x \in A') \wedge (x \in B')$	Definition of $'$
	$\Leftrightarrow$	$x \in A' \cap B'$	Definition of $\cap$

Therefore the sets $(A \cup B)'$ and $A' \cap B'$ are equal. □

Proofs of identities can often be written in this format. Notice that each "⇔" is justified by a reason in the rightmost column.

Sets that contain a finite number of elements are called *finite sets*. It is convenient to denote the number of elements in a finite set S by $|S|$. For example, if S is the set containing all the members of the Unites States House of Representatives, then $|S| = 435$.

One handy rule dealing with set sizes is the *inclusion–exclusion principle*.

$$|A \cup B| = |A| + |B| - |A \cap B| \tag{2.2.2}$$

It is easy to see why this rule must hold by looking at the Venn diagrams for $A \cup B$ and $A \cap B$ at the beginning of Section 2.2.2. Counting the elements of $A \cup B$ by counting the elements in A and B and adding would double-count the elements that lie in $A \cap B$. This is the reason for the " $-|A \cap B|$" on the right-hand side of Equation 2.2.2. This equation can help organize counting problems involving sets that overlap.

Example 2.15 The Masters of the Universe Society at a certain college accepts members who have 1600 SAT scores or 4.0 GPA scores in high school. Of the 11 members of the society, 8 had 1600 SAT scores, and 5 had 4.0 GPA scores. How many members had both 1600 SAT scores and 4.0 GPA scores?

Solution: Let A be the set of members with 1600 SAT scores, and let B be the set of members with 4.0 GPA scores. Then $A \cap B$ is the set of members with both. By the inclusion–exclusion principle (Equation 2.2.2),

$$11 = 8 + 5 - |A \cap B|$$

so there are two members with both 1600 SAT scores and 4.0 GPA scores. ♢

We will apply the inclusion–exclusion principle to harder counting problems in Chapter 4.

Exercises 2.2

1. Draw Venn diagrams to illustrate De Morgan's laws for sets (Theorem 2.1).

2. Draw a Venn diagram to show the region $A \cap B'$. This region is also denoted $A \setminus B$, and is called the *set difference*, for obvious reasons.

3. Let $A = \{2, 3, 4\}$, $B = \{3, 4, 5, 6\}$, and suppose the universal set is $U = \{1, 2, \ldots, 9\}$. List all the elements in the following sets.

   (a) $(A \cup B)'$

   (b) $(A \cap B) \times A$

   (c) $\mathcal{P}(B \setminus A)$

4. Let the following sets be given.

$$\begin{aligned} G &= \text{the set of all good citizens.} \\ C &= \text{the set of all charitable people.} \\ P &= \text{the set of all polite people.} \end{aligned}$$

   Write the statement, "Everyone who is charitable and polite is a good citizen," in the language of set theory.

5. Consider the following sets. The universal set for this problem is $\mathbf{N}$.

$$\begin{aligned} A &= \text{The set of all even numbers.} \\ B &= \text{The set of all prime numbers.} \\ C &= \text{The set of all perfect squares.} \\ D &= \text{The set of all multiples of 10.} \end{aligned}$$

   Using **only** the symbols $3, A, B, C, D, \mathbf{N}, \in, \subseteq, =, \neq, \cap, \cup, \times, {}', \emptyset, (,)$, write the following statements in set notation.

   (a) None of the perfect squares are prime numbers.

   (b) All multiples of 10 are even numbers.

   (c) The number 3 is a prime number that is not even.

   (d) If you take all the prime numbers, all the even numbers, all the perfect squares, and all the multiples of 10, you still won't have all the natural numbers.

6. Consider the following sets. The universal set U for this problem is the set of all residents of India.

$$\begin{aligned} A &= \text{the set of all English speakers.} \\ B &= \text{the set of all Hindi speakers.} \\ C &= \text{the set of all Urdu speakers.} \end{aligned}$$

Express the following sets in the symbols of set theory.

(a) Residents of India who speak English, Hindi, and Urdu.

(b) Residents of India who do not speak English, Hindi, or Urdu.

(c) Residents of India who speak English, but not Hindi or Urdu.

7. Consider the following sets. The universal set for this problem is the set of all quadrilaterals.

$$\begin{aligned} A &= \text{The set of all parallelograms.} \\ B &= \text{The set of all rhombuses.} \\ C &= \text{The set of all rectangles.} \\ D &= \text{The set of all trapezoids.} \end{aligned}$$

Using **only** the symbols $x, A, B, C, D, \in, \subseteq, =, \neq, \cap, \cup, \times, \ ', \emptyset, (,)$, write the following statements in set notation.

(a) The polygon x is a parallelogram, but it isn't a rhombus.

(b) There are other quadrilaterals besides parallelograms and trapezoids.

(c) Both rectangles and rhombuses are types of parallelograms.

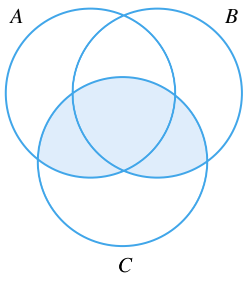
8. Two sets are called *disjoint* if they have no elements in common, i.e., if their intersection is the empty set. Prove that finite sets A and B are disjoint if and only if $|A| + |B| = |A \cup B|$. Use the definition of $\emptyset$ and the inclusion–exclusion principle (Equation 2.2.2) in your proof.

9. In a class of 40 students, everyone has either a pierced nose or a pierced ear. The professor asks everyone with a pierced nose to raise their hands. Nine hands go up. Then the professor asked everyone with a pierced ear to do likewise. This time there are 34 hands raised. How many students have piercings both on their ears and their noses?

10. How many integers in the set $\{n \in \mathbf{Z} \mid 1 \leq n \leq 700\}$ are divisible by 2 or 7?

11. Let A, B, and C be sets, and let $X = A \cup B$.

    (a) Write $|X \cap C|$ in terms of $|A \cap C|$, $|B \cap C|$, and $|A \cap B \cap C|$. Hint: In the following Venn diagram, $X \cap C$ is the shaded area.

    (b) Write $|A \cup B \cup C|$ in terms of A, B, C, $|A \cap B|$, $|A \cap C|$, $|B \cap C|$, and $|A \cap B \cap C|$. (The result is the inclusion–exclusion principle for three sets.)

12. How many integers in the set $\{n \in \mathbf{Z} \mid 1 \leq n \leq 700\}$ are divisible by 2, 5, or 7?

13. Let $S = \{a, b, c\}$. Write down all the elements in the following sets.

    (a) $S \times S$

    (b) $\mathcal{P}(S)$

14. An integer solution to the equation $3x + 4 = 7y$ is an ordered pair of integers (x, y) that satisfies the equation. For example, $(1, 1)$ is one such solution. Write the set of all integer solutions to the equation $3x + 4 = 7y$ in set builder notation.

15. Use Definition 1.6 to write the set of odd integers in set builder notation.

16. List all the elements of $\mathcal{P}(\mathcal{P}(\{1\}))$.

17. Prove part 2 of Theorem 2.1.

18. Give a direct proof that for any set S, $(S')' = S$. (Hint: Follow the format of the proof of Theorem 2.1. You need to show that

$$x \in (S')' \Leftrightarrow x \in S$$

for all x in S. Translate this statement into a statement in predicate logic, and use an equivalence rule from Section 1.2.)

19. Let E be the set of all even integers, and let O be the set of all odd integers.

    (a) Explain why $E \cup O \subseteq \mathbf{Z}$.

    (b) Explain why $\mathbf{Z} \subseteq E \cup O$.

20. Let X, Y, and Z be sets. Use the distributive property from propositional logic (Exercise 14 in Section 1.1) to prove that

$$X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z).$$

21. Let A and B be sets. Prove that $A \cap B \subseteq A \cup B$.

22. Let X be a finite set with $|X| > 1$. What is the difference between $P_1 = X \times X$ and $P_2 = \{S \in \mathcal{P}(X) \mid |S| = 2\}$? Which set, P_1 or P_2, has more elements?

23. Draw an undirected graph G with the following properties. The vertices of G correspond to the elements of $\mathcal{P}(\{0,1\})$. Two vertices (corresponding to $A, B \in \mathcal{P}(\{0,1\})$) are connected by an edge if and only if $A \cap B = \emptyset$.

24. Repeat Exercise 23 using $\mathcal{P}(\{0,1,2\})$ instead of $\mathcal{P}(\{0,1\})$.

25. Let X be any finite set. Consider the graph G described in Exercise 23, replacing $\mathcal{P}(\{0,1\})$ with $\mathcal{P}(X)$. Explain why G must be a connected graph.

2.3 Functions

You have probably seen functions before. For example, in high school algebra, you learned how to graph functions like

$$f(x) = x^2 - 3x + 2 \tag{2.3.3}$$

and do various calculations. But this is a very specific kind of function; in this section we will look at functions from a more general perspective, using the language of sets.

Like sets, functions describe mathematical relationships. In this type of relationship, the value of one object is completely determined by the value of another. In Equation 2.3.3, the value of $f(x)$ is forced if we know x's value.

2.3.1 Definition and Examples

Definition 2.1 A *function* from a set X to a set Y is a well-defined rule that assigns a single element of Y to every element of X. If f is such a function, we write

$$f\colon X \longrightarrow Y$$

and we denote the element of Y assigned to $x \in X$ by $f(x)$. The set X is called the *domain* of the function, and the set Y is called the *codomain*.

Example 2.16 Let $X = \{1, 2, 3\}$ and $Y = \{1, 2, 3, 4\}$. The formula $f(x) = x + 1$ defines a function $f\colon X \longrightarrow Y$. For this function, $f(1) = 2$, $f(2) = 3$ and $f(3) = 4$. Figure 2.11 shows one way to represent this function graphically.

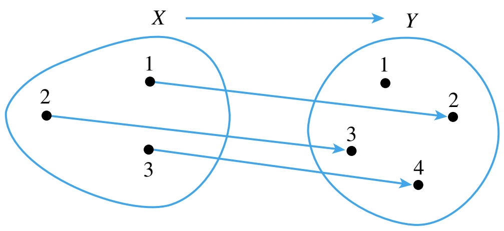


Figure 2.11 The function f maps each element of X to an element of Y.

Example 2.17 The formula $f(x) = x^2 - 3x + 2$ defines a function $f\colon \mathbf{R} \longrightarrow \mathbf{R}$.

Example 2.18 Let W be the set of all words in this book, and let L be the set of all letters in the alphabet. Define a function $f\colon W \longrightarrow L$ by setting $f(w)$ equal to the first letter in the word w. Notice that the choice of word w completely determines $f(w)$, the first letter in the word.

Example 2.19 A propositional statement in two variables defines a function

$$w\colon \{T, F\} \times \{T, F\} \longrightarrow \{T, F\}$$

where $w(A, B)$ is the truth value of the statement when the variables have truth values A and B. For example, the statement $A \vee B$ defines a function whose values are given by the following table.

A	B	$w(A, B)$
T	T	T
T	F	T
F	T	T
F	F	F

Example 2.20 Let F be the set of all nonempty finite sets of integers, so $F \subseteq \mathcal{P}(\mathbf{Z})$. Define a function

$$s\colon F \longrightarrow \mathbf{Z}$$

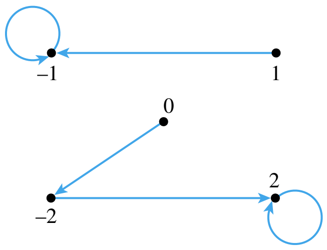
by setting $s(X)$ to be the sum of all the elements of X. For example, $s(\{1,2,3\}) = 6$.

Another word for function is *map*. This reflects the thinking that a function is a way of describing a route from one set to another. In Example 2.20, we are describing a way to get from a finite set of integers to a particular integer, i.e., we are describing a map from the set of all finite sets of integers to the set of integers. This function maps the set $\{1,2,3\}$ to the integer 6.

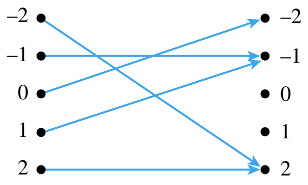
We can represent a function by a graph. The vertices of the graph represent the elements of the domain and codomain of the function. For each element in the domain, there is a directed edge to some element in the codomain.

Example 2.21 Let $N = \{-2,-1,0,1,2\}$. Define a function $s\colon N \longrightarrow N$ by $s(n) = n^2 - 2$. Represent this function with a directed graph.

Solution: Since the domain and codomain are the same set, we can represent the function with just five vertices:



However, it is often more useful to represent the domain separately from the codomain. And following the $s\colon N \longrightarrow N$ notation, we put the vertices for the domain on the left and the vertices for the codomain on the right.



◊

Notice that the definition of a function restricts the features of the graph. Every vertex representing an element of the domain must have outdegree 1. As an exercise, think about what a graph representing the following function would look like.

Example 2.22 Let X be a set. Then the *identity* function

$$1_X : X \longrightarrow X$$

is defined by $1_X(x) = x$.

The $f(x)$ notation suggests the most important aspect of the definition of function: the value of the function is completely determined by the object you "plug in" to the function. More precisely, the condition that a function must be *well defined* means that $f(x)$ has a value for every x in the domain, and

$$a = b \ \Rightarrow \ f(a) = f(b)$$

for any a and b in the domain.

A proposed function $f\colon X \longrightarrow Y$ can fail to be well defined in two ways: (1) some x in the domain X can fail to have a y in the codomain to which to map, or (2) some x in the domain can be mapped (ambiguously) to two different y's in the codomain.

Example 2.23 Let P be the set of all people (alive or dead). Let

$$m\colon P \longrightarrow P$$

be such that $m(x)$ is the birth mother of x. We have to make some reasonable biological assumptions to regard this as a well-defined function: everybody has a birth mother (e.g., no clones) and no person can have two different birth mothers.

Think of functions as tools for describing relationships between the elements of a set, or the elements of two different sets. In our list of functions above, Example 2.23 describes the maternal relationships of all people (alive or dead), while Example 2.20 shows how to describe sets of integers with integers. Example 2.22 is rather trivial and describes a trivial relationship that the elements of any set have.

It might seem obvious that functions given by formulas are always well defined, but difficulties can arise if there is more than one way to write the same element of the domain, as in the following example.

Example 2.24 Let $\mathbf{Q} = \{x/y \mid x, y \in \mathbf{Z},\ y \neq 0\}$ be the set of rational numbers. Set

$$f(x/y) = x + y$$

for any $x/y \in \mathbf{Q}$. Does this yield a well-defined function $f\colon \mathbf{Q} \longrightarrow \mathbf{Z}$?

Solution: No. As a counterexample, note that $2/3 = 4/6$, but

$$f(2/3) = 2+3 = 5 \neq f(4/6) = 4+6 = 10,$$

so f is not well defined. ♢

When defining a function on a domain like this, it is wise to check that your function is well defined.

Example 2.25 Show that the function $f\colon \mathbf{Q} \longrightarrow \mathbf{Q}$ defined by

$$f(x/y) = \frac{x+y}{y}$$

is well defined.

Proof Let $a/b = c/d \in \mathbf{Q}$. Then

$$\begin{aligned} f(a/b) &= \frac{a+b}{b} \\ &= \frac{a}{b} + \frac{b}{b} \\ &= \frac{c}{d} + \frac{d}{d} \\ &= \frac{c+d}{d} \\ &= f(c/d) \end{aligned}$$

so f is well defined. □

2.3.2 One-to-One and Onto Functions

Just as it is useful to describe different properties that relationships have, it is useful to name certain properties of functions.

Definition 2.2 A function $f\colon X \longrightarrow Y$ is *injective* (or *one-to-one*) if, for all a and b in X, $f(a) = f(b)$ implies that $a = b$. In this case we say that f is a *one-to-one mapping* from X to Y.

Definition 2.3 A function $f\colon X \longrightarrow Y$ is *surjective* (or *onto*) if, for all $y \in Y$, there exists an $x \in X$ such that $f(x) = y$. In this case we say that f *maps* X *onto* Y.

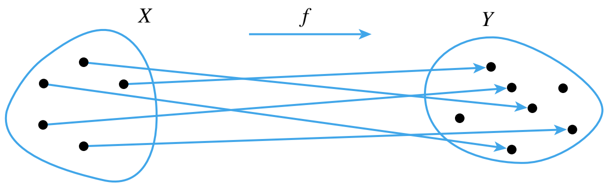


Figure 2.12 A one-to-one function maps each element of X to a different element of Y.

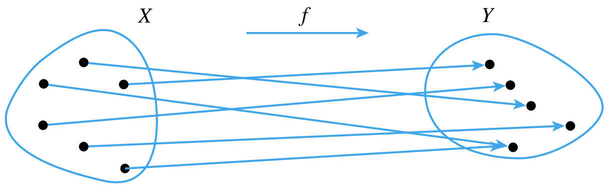


Figure 2.13 An onto function maps something onto each element of Y.

From this point on, we will use the less formal terms *one-to-one* and *onto* to describe injective and surjective functions.[3] The graphs in Figures 2.12 and 2.13 illustrate why these terms are appropriate. A one-to-one function will always map different elements of the domain to different elements of the codomain. In other words, at most, *one* element of the domain maps to any *one* element of the codomain. In terms of graph theory, the indegree of every vertex in the codomain is, at most, 1.

We use the term *image* to describe the set of all values a function can take. An onto function has every element of the codomain in its image. Figure 2.13 illustrates an onto function. Notice that the domain gets mapped *onto* the entire codomain. In the graph of an onto function, the indegree of every vertex in the codomain is at least 1.

Remember (Section 1.4.1) how definitions are used in mathematics: in order to prove a function is one-to-one or onto, we almost always have to use the definition.

Example 2.26 Prove that the function $f\colon \mathbf{Z} \longrightarrow \mathbf{Z}$ defined by $f(x) = 2x + 1$ is one-to-one.

3. Note that we are now using the word "onto" as an adjective, as well as a preposition.

Proof Let $a, b \in \mathbf{Z}$ and suppose $f(a) = f(b)$. Then

$$\begin{aligned} 2a + 1 &= 2b + 1 \\ 2a &= 2b \\ a &= b \end{aligned}$$

We have shown that $f(a) = f(b)$ implies that $a = b$, i.e., that f is one-to-one. □

Let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. So, for example, $\lfloor 4.3 \rfloor = 4$, $\lfloor -2.1 \rfloor = -3$, and $\lfloor 17 \rfloor = 17$. The function that maps x to $\lfloor x \rfloor$ is called the *floor* function.

Example 2.27 Let $f\colon \mathbf{R} \longrightarrow \mathbf{Z}$ be defined by $f(x) = \lfloor x \rfloor$. Prove that f maps $\mathbf{R}$ onto $\mathbf{Z}$.

Proof Let $n \in \mathbf{Z}$. Then, since $\mathbf{Z} \subseteq \mathbf{R}$, $n \in \mathbf{R}$ as well. But since n is an integer, $\lfloor n \rfloor = n$. Therefore $f(n) = n$. □

The proofs in the last two examples are standard. To prove a function f is one-to-one, suppose $f(a) = f(b)$ and show $a = b$. To show that a function f is onto, let y be some element of the codomain, and find some x in the domain such that $f(x) = y$.

To prove that a function is *not* one-to-one or onto, look for a counterexample. The function in Example 2.26 is not onto because, for example, $38 \in \mathbf{Z}$, but there is no integer x such that $2x + 1 = 38$. Likewise, the function in Example 2.27 is not one-to-one because $\lfloor 9.3 \rfloor = \lfloor 9.8 \rfloor$, but $9.3 \neq 9.8$. Note that these two examples show that one-to-one is not the same as onto.

Of course, it is possible for a function to be both one-to-one and onto; the identity function of Example 2.22 is one example. Such a function is called a *one-to-one correspondence*, or more formally, a *bijection*. Compare the following example to Example 2.26.

Example 2.28 Prove that the function $f\colon \mathbf{R} \longrightarrow \mathbf{R}$ defined by $f(x) = 2x + 1$ is a one-to-one correspondence.

Proof We need to show that f is both one-to-one and onto. The proof that it is one-to-one is exactly the same as in Example 2.26. To prove that f is onto, let $y \in \mathbf{R}$ be any real number. Set $x = (y-1)/2$. Then $x \in \mathbf{R}$ and

$$\begin{aligned} f(x) &= f((y-1)/2) \\ &= 2[(y-1)/2]+1 \\ &= y-1+1 \\ &= y \end{aligned}$$

Thus f is onto as well. □

Example 2.29 Let $E = \{n \in \mathbf{Z} \mid n \text{ is even}\}$ and let $O = \{n \in \mathbf{Z} \mid n \text{ is odd}\}$. Define a function

$$f \colon E \times O \longrightarrow \mathbf{Z}$$

by $f(x,y) = x+y$. Is f one-to-one and/or onto? Prove or disprove.

Solution: We first show that f is not onto. Suppose, to the contrary, that f is onto. Since $2 \in \mathbf{Z}$ is an element of the codomain, there is some ordered pair $(x,y) \in E \times O$ such that

$$f(x,y) = x+y = 2.$$

But since x is even and y is odd, $x+y$ is odd, by Exercise 9 of Section 1.5. This contradicts that 2 is even.

We next show that f is not one-to-one. Notice that

$$f(4,-3) = 1 = f(6,-5)$$

but $(4,-3) \neq (6,-5)$. This counterexample shows that f is not one-to-one. ♢

Example 2.30 Let P be a set of n points lying on a circle. Draw lines connecting every point to every other point. Suppose that the points are arranged so that no three of these lines intersect in a single point inside the circle. (Figure 2.14 shows a possible configuration.) Let X be the set of all points of intersection of the lines in the interior of the circle (note that the points in P are not included in X). Let Y be the set of all sets of four of the points on the circle, that is,

$$Y = \{\{A,B,C,D\} \subseteq P \mid A,B,C,D \text{ are all distinct}\}.$$

Describe a one-to-one correspondence $f \colon X \longrightarrow Y$. Show that your function is both one-to-one and onto.

Solution: Let $H \in X$ be a point of intersection. Then H lies on exactly two lines, so we can define $f(H)$ to be the set containing the four distinct endpoints of these two lines.

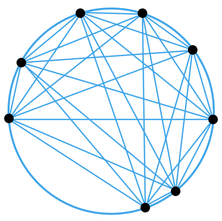


Figure 2.14 A possible configuration of points for Example 2.30 with $n = 8$.

We prove that f is one-to-one by contraposition. Suppose H and K are two distinct points of intersection, so $H \neq K$. If l_1 and l_2 are the two lines intersecting at H, then at least one of the lines intersecting at K must be different from l_1 and l_2, since two lines can intersect in at most one point. Call this third line l_3. Then $f(K)$ contains the endpoints of l_3, but $f(H)$ does not. Hence, $f(K) \neq f(H)$.

To show that f is onto, let $\{A, B, C, D\} \subseteq P$. Without loss of generality, we can relabel these points (if necessary) so that A, B, C, D are in order as you travel clockwise around the circle. Let l_1 be the line through A and C, and let l_2 be the line through B and D. Since B is on the arc going clockwise from A to C, and D is on the arc going counterclockwise from A to C, line l_1 separates B and D, so lines l_1 and l_2 intersect. Call this point of intersection H. Then $f(H) = \{A, B, C, D\}$, as required. ♢

Think of a one-to-one correspondence $f\colon X \longrightarrow Y$ as a way to assign every element of X to a unique element of Y, and *vice versa*. We sometimes write

$$X \longleftrightarrow Y$$

to emphasize the symmetry of the relationship that a one-to-one correspondence defines. Every element of one set has a unique partner in the other set.

2.3.3 New Functions from Old

There are some common ways to make new functions out of old ones. One such construction is the *composition* of two functions. If $f\colon X \longrightarrow Y$ and $g\colon Y \longrightarrow Z$, then $g \circ f$ is a function from X to Z defined by $(g \circ f)(x) = g(f(x))$.

Example 2.31 Let $f\colon \mathbf{R} \longrightarrow \mathbf{R}$ be defined by $f(x) = \lfloor x \rfloor$, and let $g\colon \mathbf{R} \longrightarrow \mathbf{R}$ be defined by $g(x) = 3x$. Then

$$\begin{aligned}(g \circ f)(2.4) &= g(f(2.4)) \\ &= g(2) \\ &= 6\end{aligned}$$

and

$$\begin{aligned}(f \circ g)(2.4) &= f(g(2.4)) \\ &= f(7.2) \\ &= 7\end{aligned}$$

This example shows that $f \circ g$ can be different from $g \circ f$. And note one potentially confusing thing about the notation: In the composition $g \circ f$, we do f first, and then apply g to the result. Order matters, in general.

Sometimes we would like to be able to "undo" a function, so that it maps back to where it came from. If $f\colon X \longrightarrow Y$ is a function, then the *inverse* of f is the function

$$f^{-1} : Y \longrightarrow X$$

that has the property that $f^{-1} \circ f = 1_X$ and $f \circ f^{-1} = 1_Y$.

Not all functions have inverses. If $f\colon X \longrightarrow Y$ has an inverse, then, for any $y \in Y$, $f(f^{-1}(y)) = y$, so f must map X onto Y. In addition, if $f(a) = f(b)$, then we can apply f^{-1} to both sides of this equation to get $a = b$, so f must be one-to-one also. So we see that if a function has an inverse, it must be a one-to-one correspondence.

Conversely, we can construct the inverse of any one-to-one correspondence $f\colon X \longrightarrow Y$ by taking $f^{-1}(y)$ to be the unique element of X that maps onto y. We know such an element exists, because f is onto, and we know this element is unique, since f is one-to-one. This is the only choice we have for f^{-1}; see Exercise 24.

Example 2.32 If $f\colon \mathbf{R} \longrightarrow \{y \in \mathbf{R} \mid y > 0\}$ is the function $f(x) = 2^x$, then the inverse of f is given by $f^{-1}(x) = \log_2 x$.

In this last example, we could have defined $f(x) = 2^x$ as a function from $\mathbf{R}$ to $\mathbf{R}$, but then it wouldn't be invertible because it wouldn't be onto.

One final way to construct functions is by *restriction*. If $f\colon X \longrightarrow Y$ is some function, and $H \subseteq X$, then the restriction of f to H is the function

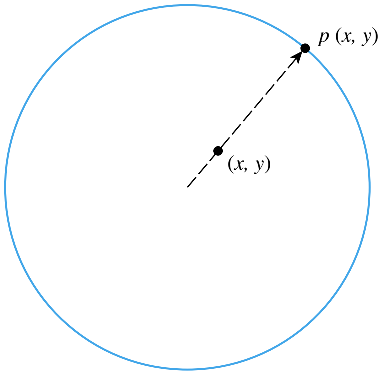
$$f|_H : H \longrightarrow Y$$

defined by taking $f|_H(x) = f(x)$. In other words, we just restrict the domain to a smaller set, and use the same rule that assigns an element of the codomain to each element of this smaller domain. Why bother? Sometimes the new, restricted function is simpler to describe, or has other desirable properties.

Example 2.33 Let D be the unit disk in $\mathbf{R}^2$, that is,

$$D = \{(x, y) \in \mathbf{R}^2 \mid x^2 + y^2 \leq 1\}$$

and let $D^* = D \setminus \{(0,0)\}$. Let $S^1 = \{(x,y) \in \mathbf{R}^2 \mid x^2 + y^2 = 1\}$ be the unit circle. Define a function $p\colon D^* \longrightarrow S^1$ by projecting straight out along a radius until you reach the boundary of the disk. See the following picture.



Getting a formula for p might be a little messy, but things can be cleaner if we consider a restriction. Let H be the circle of radius $\frac{1}{2}$:

$$H = \{(x,y) \in \mathbf{R}^2 \mid x^2 + y^2 = \frac{1}{4}\}$$

We'll leave it as an exercise to show that $p|_H(x,y) = (2x, 2y)$.

Exercises 2.3

1. Let X be a set with four elements. Represent the identity function 1_X of Example 2.22 with a directed graph in two different ways:
   (a) with four vertices, each representing an element in both the domain and the codomain.
   (b) with eight vertices, four for the domain and four for the codomain.
2. See Definitions 2.2 and 2.3. Write the definitions of one-to-one and onto in terms of predicate logic.
3. Show that the function of Example 2.20 is not one-to-one.
4. Show that the function of Example 2.20 is onto.

5. Several languages are spoken in India; let L be the set of all such languages, and let U be the set of all residents of India. Explain why the proposed function $f\colon U \longrightarrow L$ defined by
$$f(u) = \text{the language that } u \text{ speaks}$$
is not well defined.

6. Let P be a set of people, and let Q be a set of occupations. Define a function $f\colon P \longrightarrow Q$ by setting $f(p)$ equal to p's occupation. What must be true about the people in P for f to be a well-defined function?

7. Is the function of Example 2.23 onto? Why or why not? Is it one-to-one? Why or why not?

8. Consider Example 2.23. Let y be some person. What is the relationship of $(m \circ m)(y)$ to y?

9. Is the function depicted in Figure 2.12 onto? Why or why not?

10. Is the function depicted in Figure 2.13 one-to-one? Why or why not?

11. Explain why the proof in Example 2.28 could not be used to prove that the function in Example 2.26 is onto.

12. Consider the situation of Example 2.30. Describe a different one-to-one correspondence $g\colon Y \longrightarrow X$. Show that your function is both one-to-one and onto.

13. Consider the negation function $n\colon \{\mathrm{T}, \mathrm{F}\} \longrightarrow \{\mathrm{T}, \mathrm{F}\}$ given by $n(x) = \neg x$. Is n a one-to-one correspondence? What is n^{-1}?

14. Define a function $f\colon \mathbf{Z} \longrightarrow \mathbf{Z} \times \mathbf{Z}$ by $f(x) = (2x + 3, x - 4)$.
    (a) Is f one-to-one? Prove or disprove.
    (b) Does f map $\mathbf{Z}$ onto $\mathbf{Z} \times \mathbf{Z}$? Prove or disprove.

15. Define a map $t\colon \mathbf{R} \times \mathbf{R} \longrightarrow \mathbf{R} \times \mathbf{R}$ by $t(a, b) = (a + b, a - b)$. Prove that t is a one-to-one correspondence.

16. Let X be a set. Define a map $d\colon X \longrightarrow X \times X$ by $d(x) = (x, x)$.
    (a) Is d one-to-one? Prove or disprove.
    (b) Is d onto? Prove or disprove.

17. Let G be a simple, connected, undirected graph. Let V be the set of vertices in G, and let

$$P = \{\{v_1, v_2\} \mid v_1, v_2 \in V, v_1 \neq v_2\}$$

be the set of all unordered pairs of vertices. Let E be the set of all edges in G. Define a function $f\colon E \longrightarrow P$ as follows: If $e \in E$ is an edge in G, then $f(e) = \{a, b\}$, where a and b are the vertices that e touches.

    (a) Explain why $f(e)$ is always a set of size 2. (This better be true, or f is not well defined.)
    (b) Is f one-to-one? Prove or disprove.
    (c) Does f map E onto P? Prove or disprove.

18. Let $S = \{0, 1, 2, 3, 4, 5\}$, and let $\mathcal{P}(S)^*$ be the set of all nonempty subsets of S. Define a function $m\colon \mathcal{P}(S)^* \longrightarrow S$ by

$$m(H) = \text{ the largest element in } H$$

for any nonempty subset $H \subseteq S$.

    (a) Is m one-to-one? Why or why not?
    (b) Does m map $\mathcal{P}(S)^*$ onto S? Why or why not?

19. Let $X = \{n \in \mathbf{N} \mid n \text{ divides } 30030\}$, and define a function $f\colon X \longrightarrow X$ by

$$f(n) = \frac{30030}{n}.$$

    (a) Prove that f is one-to-one.
    (b) Prove that f is onto.

20. Let $f\colon \mathbf{Z} \longrightarrow \mathbf{Z}$ be defined as

$$f(x) = \begin{cases} x + 3 & \text{if } x \text{ is odd} \\ x - 5 & \text{if } x \text{ is even} \end{cases}.$$

    (a) Show that f is a one-to-one correspondence.
    (b) Find a formula for f^{-1}.

21. Define a function $f\colon \mathbf{N} \longrightarrow \mathbf{Z}$ by

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (1 - n)/2 & \text{if } n \text{ is odd} \end{cases}.$$

(a) Show that f is one-to-one correspondence.

(b) Find a formula for f^{-1}.

22. Define a function $g\colon \mathbf{Z} \longrightarrow \mathbf{N}$, by $g(z) = z^2 + 1$.

    (a) Prove that g is not one-to-one.

    (b) Prove that g is not onto.

23. Compute the inverse of $f(x) = x^3 + 1$. Check your answer by showing that $f(f^{-1}(x)) = x$ and $f^{-1}(f(x)) = x$.

24. Suppose that $f\colon X \longrightarrow Y$ has two inverses, g and h. Prove that $g = h$, that is, prove that for all $y \in Y$, $g(y) = h(y)$.

25. Define functions $f, g\colon \mathbf{N} \longrightarrow \mathbf{N}$ by

$$\begin{aligned} f(x) &= 2x \\ g(x) &= \lfloor x/2 \rfloor \end{aligned}$$

    Is $f \circ g$ the same as $g \circ f$? Explain.

26. Let $f\colon X \longrightarrow Y$ and $g\colon Y \longrightarrow Z$ be one-to-one correspondences. Prove that $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$.

27. Complete Example 2.33. That is, show that $p|_H(x, y) = (2x, 2y)$.

28. Let $q\colon \mathbf{R}^2 \longrightarrow \mathbf{R}^2$ be given by $q(x, y) = (x, 0)$. Geometrically, q is called the *projection* onto the x-axis, because it maps any point straight down (or up) to the x-axis. Let $q|_{D^*}$ be the restriction of q to the unit disk, and let p be the function defined in Example 2.33. Draw a **well-labeled** picture to show that

$$p \circ q|_{D^*} \neq q|_{D^*} \circ p.$$

29. Suppose $f\colon X \longrightarrow Y$ and $g\colon Y \longrightarrow Z$ are both onto. Prove that $g \circ f$ is onto.

*30. Suppose $f\colon X \longrightarrow Y$ and $g\colon Y \longrightarrow Z$ are both one-to-one. Prove that $g \circ f$ is one-to-one.

*31. Let $f\colon X \longrightarrow Y$ and $g\colon Y \longrightarrow Z$ be functions such that $h = g \circ f$ is a one-to-one correspondence.

    (a) Prove that f is one-to-one.

    (b) Prove that g is onto.

*32. Let $f\colon X \longrightarrow Y$ be a function. For any subset $U \subseteq X$, define the set $f(U)$:

$$f(U) = \{y \in Y \mid y = f(x) \text{ for some } x \in U\}.$$

    In particular, $f(X)$ is the image of f.

(a) Let $A \subseteq X$ and $B \subseteq X$. Does $f(A \cap B) = f(A) \cap f(B)$ in general? Prove it, or give a counterexample.

(b) Let $A \subseteq X$ and $B \subseteq X$. Does $f(A \cup B) = f(A) \cup f(B)$ in general? Prove it, or give a counterexample.

Hint: Refer to Exercise 20 of Section 1.3.

2.4 Relations and Equivalences

Although functions are used in almost every area of mathematics, many relationships are not functional. For example, we could never define a function $b(x)$ that gives the brother of x, because x might have more than one brother (or no brothers at all). A *relation* is a more general mathematical object that describes these kind of relationships.

2.4.1 Definition and Examples

Definition 2.4 A *relation* on a set S is a subset of $S \times S$. If R is a relation on S, we say that "a is related to b" if $(a, b) \in R$, which we sometimes write as $a \mathrel{R} b$. If $(a, b) \notin R$, then a is not related to b; in symbols, $a \not\mathrel{R} b$.

Some books call this a "binary relation." It is easy to see how you could generalize this definition: you could have a relation between a pair of sets, or a relation among a list of sets, but these other kinds of relations are less commonly seen.

Example 2.34 The symbols $=, <, >, \leq, \geq$ all define relations on $\mathbf{Z}$ (or on any set of numbers). For example, if $S = \{1, 2, 3\}$, then the relation on S defined by $<$ is the set $\{(1, 2), (1, 3), (2, 3)\}$.

Example 2.35 Let P be the set of all people, living or dead. For any $a, b \in P$, let $a \mathrel{R} b$ if a and b are (or were) brothers. Then R is a relation on P, and the ordered pair $(\text{Cain}, \text{Abel}) \in R$.

Example 2.36 Let W be the set of all web pages. Then

$$L = \{(a, b) \in W \times W \mid a \text{ has a link to } b\}$$

is a relation on W. In other words, $a \mathrel{L} b$ if page a links to page b.

Example 2.37 Let $a, b \in \mathbf{Z}$. If, for some $n \in \mathbf{Z}$, $n \mid (a - b)$, we say that "a is equivalent to b modulo n." The notation for this relation is

$$a \equiv b \mod n.$$

For example, $1, 4, 7, 10, 13, \ldots$ are all equivalent to modulo 3.

Notice that

$$\begin{aligned} a \equiv b \mod n &\Leftrightarrow n \mid (a - b) \\ &\Leftrightarrow a - b = kn \text{ for some } k \in \mathbf{Z} \\ &\Leftrightarrow a = b + kn \text{ for some } k \in \mathbf{Z}, \end{aligned}$$

so adding any multiple of n to a number b gives a number that is equivalent to b modulo n.

2.4.2 Graphs of Relations

Relations and graphs are similar concepts. Given a relation, there is a graph that models it.

Definition 2.5 Let R be a relation on a set X. The *directed graph* associated with (X, R) is the graph whose vertices correspond to the elements of X, with a directed edge from vertex x to vertex y whenever $x \; R \; y$.

Example 2.38 Consider the "$|$" relation on the set $X = \{2, 3, 4, 6\}$. Figure 2.15 shows a directed graph of this relation.

Often a relation R will have the property that $x \; R \; y$ if and only if $y \; R \; x$. Such a relation is called *symmetric*. For a symmetric relation, the arrows would come in pairs—one in each direction. In this case, we replace the the two arrows by a single edge and omit the arrowheads.

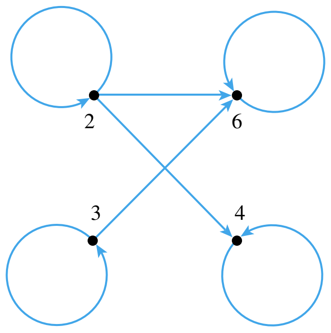


Figure 2.15 A graph of the "$|$" relation.

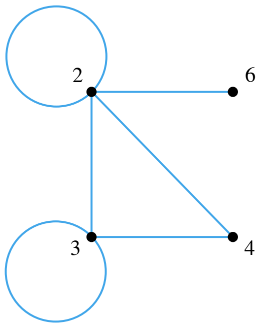


Figure 2.16 A graph of the relation in Example 2.39.

Definition 2.6 Let R be a relation on a set X, and suppose $x\ R\ y \Leftrightarrow y\ R\ x$ for all $x, y \in X$. The *undirected graph* associated with (X, R) is the graph whose vertices correspond to the elements of X, with an (undirected) edge joining any two vertices x and y for which $x\ R\ y$.

Example 2.39 Define a relation on the set $X = \{2, 3, 4, 6\}$ by setting $a\ R\ b$ whenever $ab < 13$. Since $ab < 13 \Leftrightarrow ba < 13$, this is a symmetric relation. An undirected graph of this relation is shown in Figure 2.16.

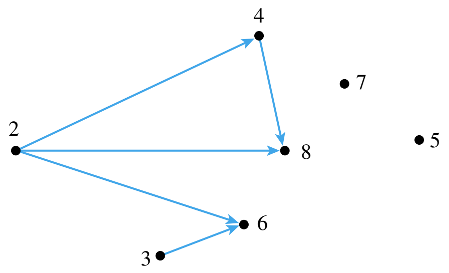
Every relation yields a graph, but there are graphs that don't have relations associated to them. One example is the graph of the Königsberg bridges in Example 2.1. The double edges in this graph cannot be represented by a relation, because any ordered pair (x, y) can occur, at most, once in the Cartesian product $X \times X$.

2.4.3 Relations vs. Functions

What's the difference between a relation and a function? Strictly speaking, a relation on X is a function $X \longrightarrow X$ if each $x \in X$ occurs exactly once as the first element of an ordered pair. But this description is fairly mechanical; what does it really mean? Think of a function as a way of describing how the elements in one set *depend* on the elements in another set. When we graph a function $y = f(x)$, we usually say that y is the *dependent* variable and x is the *independent* variable. This means we get to pick x, and y depends on what we choose for x.

We can't view relations this way, because any given x may be related to several other elements or none at all. Instead, think of relations as describing comparisons within a set or links between elements of a set.

Example 2.40 Let $X = \{2, 3, 4, 5, 6, 7, 8\}$, and say that two elements $a, b \in X$ are related if $a \mid b$ and $a \neq b$. We can represent this relation with a directed graph: the elements of X are the vertices, and there is a directed edge from distinct vertices a and b whenever $a \mid b$.



This example shows how the relationships described by a relation aren't constrained the way relationships described by a function are: the element 2 is related to three other elements, while 5 and 7 aren't related to anything. Thus this relation does not describe a function on X.

2.4.4 Equivalence Relations

Definition 2.7 A relation R on a set S is an *equivalence relation* if it satisfies all three of the following properties.

1. *Reflexivity.* For any $a \in S$, $a \, R \, a$.
2. *Symmetry.* For any $a, b \in S$, $a \, R \, b \Leftrightarrow b \, R \, a$.
3. *Transitivity.* For any $a, b, c \in S$, if $a \, R \, b$ and $b \, R \, c$, then $a \, R \, c$.

In other words, an equivalence relation is a relation that is reflexive, symmetric, and transitive.

Example 2.41 The relation on $\mathbf{Z}$ defined by $=$ is an equivalence relation, because equality is reflexive, symmetric, and transitive.

The idea of an equivalence relation might seem abstract at first, but you are already quite familiar with an important example: fractions. In elementary school, you learned that equivalent fractions represent the same number; for instance, $\frac{3}{6}$ is the same as $\frac{1}{2}$. The next example shows how to prove that this notion is an equivalence relation on the set S of all symbols of the form $\frac{x}{y}$, where x and y are integers. Be careful: the set S is a set of symbols, not a set of numbers. Two different symbols in S can represent the same number; this is the point of the equivalence relation.

Example 2.42 Let S be the set of all symbols of the form $\frac{x}{y}$, where x and y are integers. In other words, $S = \left\{\frac{x}{y} \mid x, y \in \mathbf{Z}\right\}$. Define a relation R on S as follows. For any elements $\frac{x}{y}$ and $\frac{z}{w}$ in S, $\frac{x}{y}\ R\ \frac{z}{w}$ if $xw = yz$. The proof that R is an equivalence relation has three parts:

Proof

1. *Reflexivity.* Suppose that $\frac{x}{y} \in S$. Since $xy = yx$, we have $\frac{x}{y}\ R\ \frac{x}{y}$.
2. *Symmetry.* Suppose that $\frac{x}{y}, \frac{z}{w} \in S$ and suppose that $\frac{x}{y}\ R\ \frac{z}{w}$. By the definition of R, this means that $xw = yz$, which is the same thing as saying $zy = wx$. Thus $\frac{z}{w}\ R\ \frac{x}{y}$, as required.
3. *Transitivity.* Let $\frac{x}{y}, \frac{z}{w}, \frac{p}{q} \in S$ with $\frac{x}{y}\ R\ \frac{z}{w}$ and $\frac{z}{w}\ R\ \frac{p}{q}$. Then $xw = yz$ and $zq = wp$. Using substitution, $xq = (yz/w)(wp/z) = yp$. This shows that $\frac{x}{y}\ R\ \frac{p}{q}$, as required.

□

The next example is a little less familiar, but notice that the proof follows the same template.

Example 2.43 Given any function $f \colon X \to Y$, define a relation on X as follows. For any $a, b \in X$, $a\ R\ b$ if $f(a) = f(b)$. The proof that R is an equivalence relation has three parts:

Proof

1. *Reflexivity.* Suppose that $a \in X$. Since f is a well-defined function, $f(a) = f(a)$, so $a\ R\ a$.
2. *Symmetry.* Suppose that $a, b \in X$ and that $a\ R\ b$. By the definition of R, this means that $f(a) = f(b)$, which is the same thing as saying $f(b) = f(a)$. Thus $b\ R\ a$, as required.
3. *Transitivity.* Let $a, b, c \in X$ with $a\ R\ b$ and $b\ R\ c$. Then $f(a) = f(b)$ and $f(b) = f(c)$, so by substitution, $f(a) = f(c)$. This shows that $a\ R\ c$.

□

Example 2.44 Say that two computers on a network are related if it is possible to establish a network connection between the two (e.g., it is possible to `ping` one from another). In order for this to be an equivalence relation, any computer must be able to `ping` itself (reflexive), any computer must be able to return a `ping` (symmetric), and any computer must be able to pass along a `ping` to any other computer that it can `ping` (transitive).

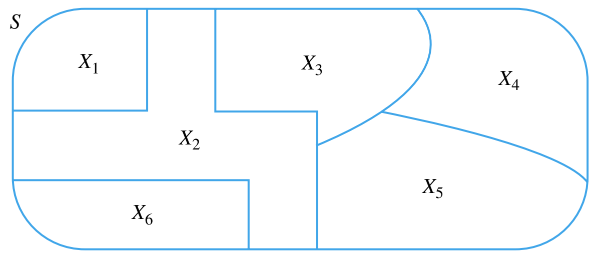


Figure 2.17 The set $P = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ is a partition of the set S.

Example 2.45 A *partition* of a set S is a set P of nonempty subsets of S with the following properties.

1. For any $a \in S$, there is some set $X \in P$ such that $a \in X$. The elements of P are called the *blocks* of the partition.

2. If $X, Y \in P$ are distinct blocks, then $X \cap Y = \emptyset$.

Informally, a partition is a way of dividing a set into nonoverlapping pieces; in a Venn diagram, a partition looks like an arrangement of walls, where the blocks of the partition are the rooms. See Figure 2.17.

Suppose P is a partition of S. Define a relation on S by setting $a \; R \; b$ if a and b belong to the same block. We can prove that R is an equivalence relation.

Proof

1. *Reflexivity.* Suppose that $a \in S$. Then a must be in some block X, and a is evidently in the same block as itself. So $a \; R \; a$.

2. *Symmetry.* Suppose that $a, b \in S$ and suppose that $a \; R \; b$. Thus a and b are in the same block, which is the same as saying that b and a are in the same block. Therefore, $b \; R \; a$.

3. *Transitivity.* Let $a, b, c \in S$ with $a \; R \; b$ and $b \; R \; c$. Then a and b are in the same block, and b and c are in the same block, so a and c are in the same block. Therefore $a \; R \; c$, as required.

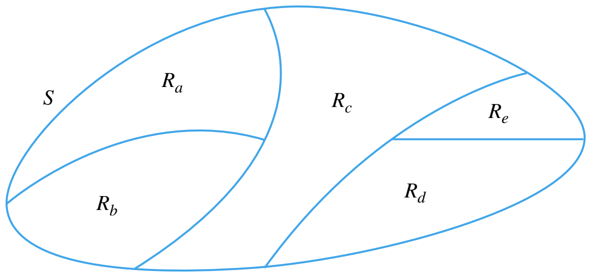
□

Example 2.45 shows that every partition determines an equivalence relation. It is also true that every equivalence relation determines a partition. Partitions and equivalence relations are essentially the same thing.

Theorem 2.2 *Let R be an equivalence relation on a set S. For any element $x \in S$, define $R_x = \{a \in X \mid x \; R \; a\}$, the set of all elements related to x. Let P be the collection of distinct subsets of S formed in this way, that is, $P = \{R_x \mid x \in S\}$. Then P is a partition of S.*

A formal proof of this theorem takes a little bit of work, and we'll omit the proof for the sake of brevity. Informally, such a proof must show that the two properties of a partition are satisfied. The first property is easy: for any $a \in S$, there is a block containing a, namely R_a. The second property is the hard part. Roughly speaking, the transitive and symmetric properties guarantee that if two blocks overlap at all, then they must overlap completely.

The sets R_x are called the *equivalence classes* formed by the equivalence relation. A set S partitioned into equivalence classes looks like this:



In this picture, the elements $a, b, c, d, e \in S$ are called equivalence class *representatives*. As the name suggests, there is usually a choice of representatives for a given equivalence class. By Theorem 2.2, different equivalence classes must have inequivalent representatives.

Example 2.46 Let W be the set of all words in the sentence, "There are more things in heaven and earth, Horatio, than are dreamt of in your philosophy." Define a relation R on W as follows: for any words $w_1, w_2 \in W$, $w_1 \; R \; w_2$ if the first letter of w_1 is the same as the first letter of w_2, without regard to upper or lower cases. (Exercise: show this is an equivalence relation.) The equivalence classes are the sets of all words beginning with the same letter. For example, $R_{\text{things}} = \{\text{There, things, than}\}$.

2.4.5 Modular Arithmetic

Consider the "$\equiv \mod n$" relation of Example 2.37. To check that this is an equivalence relation, we need to prove that it is reflexive, symmetric, and transitive. We'll show transitivity, and leave the proofs of the other two properties as exercises.

Lemma 2.1 The "$\equiv \mod n$" relation on $\mathbf{Z}$ is transitive.

Proof Let $a, b, c \in \mathbf{Z}$, and suppose that $a \equiv b \mod n$ and $b \equiv c \mod n$. This means that $a = b + kn$ and $b = c + ln$ for some integers k and l. Substituting the second equation into the first, we find that

$$a = (c + ln) + kn = c + (l + k)n$$

so $a \equiv c \mod n$, as required for transitivity. □

The set of equivalence classes formed by this equivalence relation is called the *integers modulo* n, and is denoted $\mathbf{Z}/n$. We use $[k]$ to denote the equivalence class of $[k]$. For example, the elements of $\mathbf{Z}/3$ are

$$\begin{aligned}
[0] &= \{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\} \\
[1] &= \{\dots, -8, -5, -2, 1, 4, 7, 10, \dots\} \\
[2] &= \{\dots, -7, -4, -1, 2, 5, 8, 11, \dots\}.
\end{aligned}$$

The next result is the basis for modular arithmetic.

Proposition 2.1 Let $[a]$ and $[b]$ be equivalence classes in $\mathbf{Z}/n$. Suppose that $x \in [a]$ and $y \in [b]$. Then $x + y \in [a + b]$ and $xy \in [ab]$.

Proof Let $[a], [b] \in \mathbf{Z}/n$, and let $x \in [a]$ and $y \in [b]$. By the definition of $\equiv \mod n$, $x = a + kn$ and $y = b + ln$ for some integers k and l. Therefore

$$x + y = (a + kn) + (b + ln) = (a + b) + (k + l)n$$

and

$$xy = (a + kn)(b + ln) = ab + aln + bkn + kln^2 = ab + (al + bk + kln)n,$$

so $x + y \equiv a + b \mod n$ and $xy \equiv ab \mod n$, as required. □

What's so important about this proposition? It shows that if we add or multiply equivalence class *representatives* (no matter which ones we choose), we get a representative of the "right" equivalence class. For example, in $\mathbf{Z}/3$, $-6 \in [0]$ and $11 \in [2]$. Their sum, $-6 + 11 = 5$, is in $[2]$, the natural value for the "sum" of $[0]$ and $[2]$.

In other words, the operations of addition and multiplication on *equivalence classes* are well defined:

$$\begin{aligned}
[a] + [b] &= [a + b] \\
[a] \cdot [b] &= [ab].
\end{aligned}$$

This means we can add and multiply elements in $\mathbf{Z}/n$ by adding and multiplying the numbers we use to represent the equivalence class. These are the operations of *modular arithmetic*. For example, in the modular arithmetic of $\mathbf{Z}/12$,

$$[6] + [8] = [2]$$

because $14 \equiv 2 \mod 12$.

We usually represent an equivalence class in $\mathbf{Z}/n$ by its least non-negative representative, so the operations of modular arithmetic really are just the ordinary arithmetic operations followed by taking the remainder upon division by n.

Although the elements of $\mathbf{Z}/n$ are technically sets of numbers (equivalence classes), we can think of them as a new type of number with new rules for multiplication and addition. Sometimes we'll even omit the [ ]'s around these numbers if the meaning is clear from the context.

Example 2.47 Figure 2.18 shows how to add and multiply in $\mathbf{Z}/6$. Notice the patterns in these tables. The "numbers" 0 and 1 in $\mathbf{Z}/6$ add and multiply as they do in standard integer arithmetic, but the other numbers behave differently. For example, $5 \in \mathbf{Z}/6$ acts like -1, both for addition and multiplication. This reflects the fact that $[5] = [-1]$ as equivalence classes in $\mathbf{Z}/6$.

Modular arithmetic can come in quite handy. For example, if we want a counter to keep track of an n-stage process that needs to be repeated several times, we would start at [0] and add [1] in $\mathbf{Z}/n$ at each stage, and the counter would cycle through n values and repeat. Another application—ISBN check digits—appears in the exercises.

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

·	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Figure 2.18 Addition and multiplication tables for $\mathbf{Z}/6$.

Exercises 2.4

1. Let $X = \{0, 1, 2, 3, 4\}$. Draw the graph associated with the $<$ relation on X. Should this graph be directed or undirected?

2. Let $X = \{0, 1, 2, 3, 4\}$. Define a relation R on X such that $x \; R \; y$ if $x + y = 4$. Draw the graph associated with this relation. Should this graph be directed or undirected?

For Exercises 3–6, define a relation $\leftrightharpoons$ on the set S of all strings of letters: two strings are related if you can get one from the other by reversing one pair of adjacent letters. For example, `cow` $\leftrightharpoons$ `ocw` but `cow` $\not\leftrightharpoons$ `woc`.

3. Consider all the strings you can form with the letters `c`, `a`, and `t` (there are six). Draw the graph whose nodes are these six strings and whose edges represent the $\leftrightharpoons$ relation. Should this be a directed or an undirected graph?

4. Find an Euler path in the graph you made in Exercise 3.

5. Consider the graph formed by the $\leftrightharpoons$ relation on the set of all the strings you can form from the letters `l`, `y`, `n`, and `x`. Does this graph have an Euler path? Why or why not?

6. Does the graph of the $\leftrightharpoons$ relation on the set of all strings formed from the letters `l`, `e`, `o`, `p`, `a`, `r`, and `d` have an Euler path? Why or why not?

7. Suppose you wanted to model an equivalence relation with a graph. Would you use a directed or an undirected graph? What would the equivalence classes look like? Explain.

8. Define a relation on $\mathbf{Z}$ by $a \; R \; b$ if $a^2 = b^2$.

   (a) Prove that R is an equivalence relation.

   (b) Describe the equivalence classes.

9. Explain why the web linking relation in Example 2.36 is not an equivalence relation. (You only need to give one reason why it fails to be an equivalence relation.)

10. Prove that the relation defined in Example 2.46 is an equivalence relation.

11. Explain why the relation R on $\{0,1,2\}$ given by

$$R = \{(0,0),(1,1),(2,2),(0,1),(1,0),(1,2),(2,1)\}$$

is not an equivalence relation. Be specific.

12. Let $A = \{1,2\}$. Write out the subset of $A \times A$ defined by the $\leq$ relation on A.

13. Let $X = \{0,1\}$.

    (a) List (as subsets of $X \times X$) all possible relations on X.

    (b) Which of the relations in part (a) are equivalence relations?

14. Let X be a set. Define a relation R on $\mathcal{P}(X)$ by

$$A\ R\ B \ \Leftrightarrow\ A \cap B = \emptyset$$

for $A, B \in \mathcal{P}(X)$. Determine whether this relation is reflexive, symmetric, and/or transitive.

15. Let S be the set of all provinces in Canada. For $a, b \in S$, let $a\ R\ b$ if a and b border each other. Which properties of an equivalence relation (reflexive, symmetric, transitive) does the relation R satisfy?

16. Let T be the set of all movie actors and actresses. For $x, y \in T$, define $x\ R\ y$ if there is some movie that both x and y appear in. Which properties of equivalence relations does R satisfy?

17. Let W be the set of words in the English language. Define a relation R on W by

$$w_1\ R\ w_2 \Leftrightarrow w_1 \text{ and } w_2 \text{ have a letter in common.}$$

Which properties of equivalence relations does R satisfy? Explain.

18. Let W be the set of words in the English language. Define a relation S on W by

$$w_1\ S\ w_2 \Leftrightarrow w_1 \text{ has at least as many letters as } w_2.$$

Which properties of equivalence relations does S satisfy? Explain.

19. Let X be a finite set. For subsets $A, B \in \mathcal{P}(X)$, let $A\ R\ B$ if $|A| = |B|$. This is an equivalence relation on $\mathcal{P}(X)$. If $X = \{1,2,3\}$, list the equivalence classes.

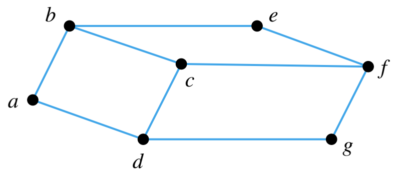
20. A playground consists of several structures, some of which are connected by bridges. The ground is covered with bark chips. Define a relation on the set of structures on the playground as follows: two structures are related if it is possible for a child to get from one to the other without walking on the bark chips (otherwise known as "hot lava").

    (a) Can you imagine a playground for which this would not be an equivalence relation? Explain.

    (b) Suppose this relation is an equivalence. In the language of playground equipment, describe the equivalence classes.

21. Let G be a connected, undirected graph, and let V be the set of all vertices in G. Define a relation R on V as follows: for any vertices $a, b \in V$, $a \; R \; b$ if there is a path from a to b with an even number of edges. (A path may use the same edge more than once.) Prove that R is an equivalence relation.

22. Suppose the equivalence relation of Exercise 21 is defined on the vertices of the following graph. What are the equivalence classes?



23. Recall the definition of even numbers.

    Definition. An integer n is even if $n = 2k$ for some integer k.

    Define a relation R on $\mathbf{Z}$ by $x \; R \; y$ if $x + y$ is even.

    (a) Show that R is an equivalence relation.

    (b) Describe the equivalence classes formed by this relation.

24. Lemma 2.1 states that the "$\equiv \mod n$" relation on $\mathbf{Z}$ is transitive. Show that it is also symmetric and reflexive.

25. Compute the following arithmetic problems in $\mathbf{Z}/8$. Represent your answer with the least positive representative of the appropriate equivalence class.

    (a) $[3] + [7]$

    (b) $[2] \cdot ([4] + [5])$

    (c) $([3] + [4]) \cdot ([5] + [6])$

26. For any $[x] \in \mathbf{Z}/n$ and any $k \in \mathbf{Z}$, we can define $k[x]$ by
$$k[x] = \underbrace{[x] + [x] + \cdots + [x]}_{k \text{ times}}$$
where the result is an element of $\mathbf{Z}/n$. We say $k[x]$ is a *multiple* of $[x]$.
    (a) List all the multiples of [3] in $\mathbf{Z}/9$.
    (b) List all the multiples of [3] in $\mathbf{Z}/8$.

27. Consider the function $p\colon \mathbf{Z} \longrightarrow \mathbf{Z}/n$ defined by $p(n) = [n]$. Prove that this function is onto but not one-to-one.

28. Construct the addition and multiplication tables for $\mathbf{Z}/4$.

29. Construct the multiplication table for $\mathbf{Z}/11$.

Exercises 30–32 deal with the method of using *check digits* in ISBN numbers. Prior to 2007, every commercially available book was given a 10-digit International Standard Book Number, usually printed on the back cover next to the barcode. The final character of this 10-digit string is a special digit used to check for errors in typing the ISBN number. If the first nine digits of an ISBN number are $a_1a_2a_3a_4a_5a_6a_7a_8a_9$, the tenth digit is given by the formula
$$a_{10} = (1a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 + 6a_6 + 7a_7 + 8a_8 + 9a_9) \mod 11,$$
where $a_{10} = \texttt{X}$ if this value is 10.

30. Calculate the tenth digit of the ISBN whose first nine digits are `039481500`.

31. Suppose $a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}$ is a valid ISBN number. Show that
$$(1a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 + 6a_6 + 7a_7 + 8a_8 + 9a_9 + 10a_{10}) \equiv 0 \mod 11.$$

32. Is `0060324814` a valid ISBN number?

*33. Show that the check digit will always detect the error of switching two adjacent digits. That is, show that $a_1 \cdots a_k a_{k+1} \cdots a_9$ and $a_1 \cdots a_{k+1} a_k \cdots a_9$ have different check digits.

*34. Show that the check digit will always detect the error of changing a single digit. Hint: The proof has something to do with the multiplication table for $\mathbf{Z}/11$ (Exercise 29).

*35. Unfortunately, there were too many books and not enough ISBN numbers; effective January 2007, ISBN numbers must be 13 digits long. The check digit scheme for 13-digit ISBN numbers is different. Explain why the obvious modification to the old system won't work. That is, find a 12-digit string $a_1 a_2 \cdots a_{12}$ where the quantity

$$(1a_1 + 2a_2 + \cdots + 12a_{12}) \mod 14$$

doesn't change after changing a single digit.

*36. Will the "obvious modification" in Exercise 32 detect the error of switching two adjacent digits?

2.5 Partial Orderings

Most of the relationships modeled in the previous section express sameness; an equivalence relation is a mathematical way of describing how two objects are similar. However, there are many situations in which we would like to quantify how objects of a set are different from each other. The relations in this section will give us a mathematical way to analyze comparisons and rankings.

2.5.1 Definition and Examples

In addition to equivalence relations, there is another type of relation that is worth studying as a special case. Compare the following with Definition 2.7.

Definition 2.8 A relation R on a set S is a *partial ordering* if it satisfies all three of the following properties.

1. *Reflexivity.* For any $a \in S$, $a \ R \ a$.
2. *Transitivity.* For any $a, b, c \in S$, if $a \ R \ b$ and $b \ R \ c$, then $a \ R \ c$.
3. *Antisymmetry.* For any $a, b \in S$, if $a \ R \ b$ and $b \ R \ a$, then $a = b$.

The reflexivity and transitivity properties are the same as in the definition of an equivalence relation, but the third property is new: it is called "antisymmetry" because it says that $a \ R \ b$ *never* happens when $b \ R \ a$, unless $a = b$.

Example 2.48 If S is a set of numbers, then $\leq$ defines a partial ordering on S.

Since Example 2.48 is a natural example of a relation that is reflexive, antisymmetric, and transitive, we will often use the symbol $\preceq$ to represent a generic partial ordering relation (instead of R). This notation has the advantage of allowing us to write $a \prec b$ when $a \preceq b$ and $a \neq b$.

Example 2.49 Let S be any set, and let $\mathcal{P}(S)$ be the power set of S. Then $\subseteq$ defines a partial ordering on $\mathcal{P}(S)$.

Proof Let $A \in \mathcal{P}(S)$. Then $A \subseteq A$, since $x \in A$ obviously implies that $x \in A$. So $\subseteq$ is reflexive.

Let $A, B, C \in \mathcal{P}(S)$, and suppose $A \subseteq B$ and $B \subseteq C$. Then for all $x \in S$, $x \in A \Rightarrow x \in B$ and $x \in B \Rightarrow x \in C$, so $x \in A \Rightarrow x \in C$. Thus $A \subseteq C$, so $\subseteq$ is transitive.

Finally, let $A, B \in \mathcal{P}(S)$, and suppose that $A \subseteq B$ and $B \subseteq A$. Then $x \in A \Leftrightarrow x \in B$, so $A = B$. Therefore $\subseteq$ is antisymmetric. □

Example 2.50 The "divides" relation $|$ defines a partial ordering on $\mathbf{N}$. The proof of this is an exercise.

If a set X has a partial ordering $\preceq$ on it, we say that $(X, \preceq)$ is a partially ordered set, or *poset*, for short. In a poset, it is possible to have elements a and b such that neither $a \preceq b$ nor $b \preceq a$. Such elements are called *incomparable*. In Example 2.50, 12 and 25 are incomparable because $12 \nmid 25$ and $25 \nmid 12$.

If a poset $(X, \preceq)$ has no incomparable elements, it is called a *total ordering*. For example, the real numbers $\mathbf{R}$ are totally ordered by the $\leq$ relation.

2.5.2 Hasse Diagrams

In the last section, we saw that the key fact about equivalence relations was that they break a set up into partitions. Partial orderings are important because they define a hierarchy among the elements of a set. We use *Hasse diagrams* to describe this hierarchy graphically.

Suppose $\preceq$ is a partial ordering on a set X. A Hasse diagram for $(X, \preceq)$ consists of a label, or *node*, for each element in the set, along with lines connecting related nodes. More specifically, if x, y are distinct elements of X with $x \preceq y$, and there are no elements z such that $x \prec z \prec y$, then there should be an upward sloping line from node x to node y in the Hasse diagram.

Example 2.51 Let $T = \{1, 2, 3\}$ and consider the poset $(\mathcal{P}(T), \subseteq)$. Since there are eight subsets of T, the Hasse diagram has eight nodes. See Figure 2.19. Note that by following the edges up the diagram, you move from one set to another set that contains it. And since $\subseteq$ is transitive, every set you encounter as you move up the diagram will contain the set where you started.

Like binary search trees, Hasse diagrams are graphs with extra information encoded in the way the graph is drawn. Traditionally, Hasse diagrams don't have arrows, but the edges are, in fact, directed: in an edge between nodes x and y, x is drawn above y when $y \preceq x$.

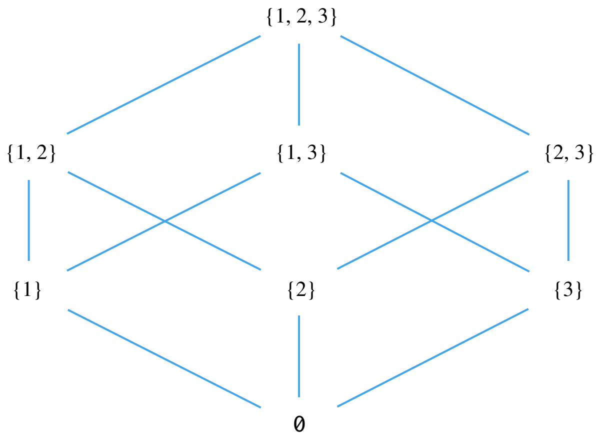


Figure 2.19 The Hasse diagram for $(\mathcal{P}(\{1,2,3\}), \subseteq)$.

Example 2.52 Let $X = \{2,3,4,5,6,7,8\}$, and say two elements $a, b \in X$ are related if $a \mid b$. Figure 2.20 shows the Hasse diagram for this poset. Compare this to the graph in Example 2.40, and notice that there are no edges in the Hasse diagram between related nodes that are connected by a sequence of edges. Here we don't need an edge between 2 and 8, since you can get there via node 4.

A partial order on a set gives you a way of comparing elements and ranking them. In this ranking system, there are often least elements and greatest elements. If $(X, \preceq)$ is a poset, we say that an element $m \in X$ is *minimal* if there is no $x \in X$ $(x \neq m)$ such that $x \preceq m$. Similarly, we say that an element $M \in X$ is *maximal* if there is no $x \in X$ $(x \neq M)$ such that $M \preceq x$. For example, in Figure 2.20, the minimal elements are 2 and 3 and the only maximal element is 8. In Figure 2.19, $\emptyset$ is minimal and $\{0,1,2\}$ is maximal.

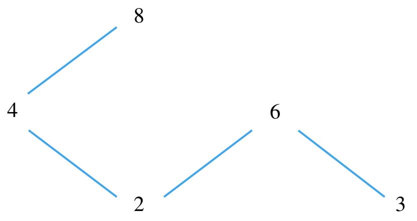


Figure 2.20 The Hasse diagram for $(X, |)$.

2.5.3 Topological Sorting

Partial orderings are useful for describing the relationships among the parts of a complex process, such as manufacturing assembly or scheduling. Suppose that some process consists of a finite set, T, of tasks. Some tasks are dependent on others being done first. (For example, when you get dressed, you must put on your socks before putting on your shoes.) If $x, y \in T$ are two tasks, let $x \preceq y$ if x must be done before y. This is a partial ordering.

In most real-world applications, $(T, \preceq)$ won't be a totally ordered set. This is because some steps of the process do not depend on other steps being done first. (You can put your watch on at any time in the process of getting dressed.) However, if we want to create a schedule in which all of the tasks in T are done in some sequential order, we need to create a total ordering on T. (At some point, we just need to decide on an order to put our clothes on.) Furthermore, this total ordering must be compatible with the partial ordering. In other words, we need a total order relation $\preceq_t$ such that

$$x \preceq y \Rightarrow x \preceq_t \; y$$

for all $x, y \in T$. The technique of finding such a total ordering is known as *topological sorting.*

Performing a topological sort involves making an ordered list of all the tasks. Represent the partial ordering on T with a Hasse diagram. Then repeat the following steps until there are no more tasks.

- Choose a minimal element in your Hasse diagram. (Note that you can always find a minimal element, but there might be more than one.) Add this element to the end of the list you are making.

- Remove the element from your Hasse diagram.

Example 2.53 Julia is interested in completing a Computer Science minor, and has several courses left to take. Some of the courses have prerequisite courses that must be taken first.

Courses Needed	Prerequisites
Calculus I	
Calculus II	Calculus I
Calculus III	Calculus II
Linear Algebra	Calculus II
Programming II	Calculus I
Software Development	Programming II
3D Computer Graphics	Calculus III, Linear Algebra, Programming II

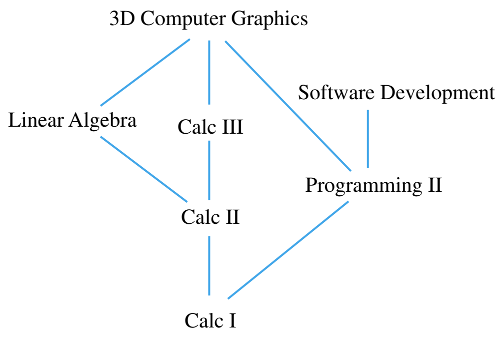


Figure 2.21 The Hasse diagram for Example 2.53.

In what order must Julia take these courses to ensure that she will satisfy the prerequisites for each course? Is this ordering unique? What is the minimum number of semesters it will take Julia to finish?

Solution: Figure 2.21 shows the Hasse diagram for this poset. To perform a topological sort, we start by choosing a minimal element in the Hasse diagram. The only minimal element is Calculus I, so that must be Julia's first class. We then remove Calculus I from the diagram.

After Calculus I has been removed, there are two minimal elements: Calculus II and Programming II. Julia could take either of these next, so the ordering of courses is not unique. Let's choose Calculus II as the second course, and remove it from the diagram. This leaves Linear Algebra, Calculus III, and Programming II as minimal elements. Julia needs to take all three before she takes 3D Computer Graphics, and she needs to take Programming II before she takes software development. Again, the ordering is not unique. One allowable order for the remaining classes is Calculus III, Programming II, Software Development, Linear Algebra, and, finally, 3D Computer Graphics.

If we are trying to minimize the number of semesters, we repeat the process, except at each stage we choose *all* the minimal elements, so Julia takes as many courses as she is allowed to take each semester. This would yield the following schedule.

Semester	Courses
1	Calculus I
2	Calculus II, Programming II
3	Calculus III, Linear Algebra, Software Development
4	3D Computer Graphics

◇

Topological sorting is a pretty simple technique, once you have written the dependencies in the form of a Hasse diagram. This simplicity illustrates the power of a graphical approach to a discrete problem.

2.5.4 Isomorphisms

Sometimes two problems are so closely related that solving one gives a solution to the other. When two mathematical objects are fundamentally the same in every aspect of structure and function, we can define a map called an *isomorphism* to describe the similarity.

Example 2.54 Let $F \subseteq \mathbf{N}$ be the set of all factors of 30, and consider the poset $(F, |)$ given by the "divides" relation. As in Example 2.52, we can construct the Hasse diagram for this partially ordered set. See Figure 2.22.

Notice how this Hasse diagram has the same structure as the Hasse diagram for $(\mathcal{P}(\{1,2,3\}), \subseteq)$ in Example 2.51. These two posets are the same in a very specific way.

Definition 2.9 Let $(X_1, \preceq_1)$ and $(X_2, \preceq_2)$ be partially ordered sets. Then $(X_1, \preceq_1)$ is *isomorphic* to $(X_2, \preceq_2)$ if there is a one-to-one correspondence $f\colon X_1 \longrightarrow X_2$ such that

$$a \preceq_1 b \;\Leftrightarrow\; f(a) \preceq_2 f(b)$$

for all $a, b \in X_1$. In this case, we say that f is an *isomorphism* and we write $(X_1, \preceq_1) \cong (X_2, \preceq_2)$ to denote that these posets are isomorphic.

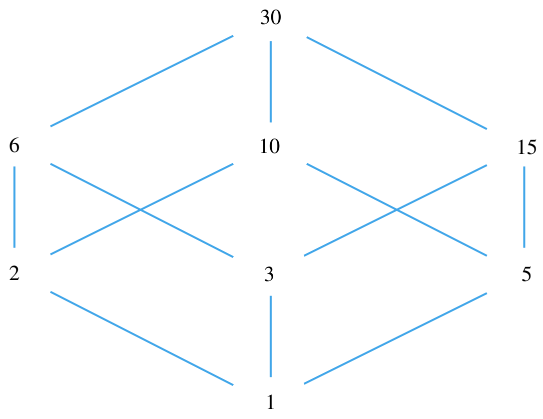


Figure 2.22 The Hasse diagram for $(F, |)$.

The following table defines a one-to-one correspondence $f\colon \mathcal{P}(\{1,2,3\}) \longrightarrow F$ that describes the isomorphism between $(F, |)$ and $(\mathcal{P}(\{1,2,3\}), \subseteq)$.

X	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
$f(X)$	1	2	3	5	6	10	15	30

Since every element of F appears exactly once in this table, f is a one-to-one correspondence. To see that $X \subseteq Y$ if and only if $f(X) \mid f(Y)$ for all $X, Y \subseteq \{1,2,3\}$, it is sufficient to observe that the edges in the the Hasse diagram for $(F, |)$ correspond exactly to the edges in the Hasse diagram for $(\mathcal{P}(\{1,2,3\}), \subseteq)$, and that this correspondence is given by f. This shows that $(F, |) \cong (\mathcal{P}(\{1,2,3\}), \subseteq)$.

At this point it is natural to wonder whether this isomorphism is a happy coincidence, or perhaps there is something deeper going on. Are the partial orders given by $\subseteq$ and $|$ related in some fundamental way? In the exercises, you will have the opportunity to check that the Hasse diagrams for these relations agree when applied to the factors of $210 = 2 \cdot 3 \cdot 5 \cdot 7$ and the set $\{1,2,3,4\}$. But will this always happen?

Let's take a closer look at the relationship between the posets $(F, |)$ and $(\mathcal{P}(\{1,2,3\}), \subseteq)$. Since $30 = 2 \cdot 3 \cdot 5$, every factor of 30 can be written (uniquely) as a product of the form

$$\begin{pmatrix} 2 \\ \text{or} \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ \text{or} \\ 1 \end{pmatrix} \begin{pmatrix} 5 \\ \text{or} \\ 1 \end{pmatrix}.$$

Choosing a factor $k \in F$ amounts to choosing whether or not to include each prime 2, 3, or 5 in this product. Similarly, choosing a subset X of the set $\{1,2,3\}$ amounts to choosing which of the three members $1, 2, 3$ to include in X. This explains why the above function f is a one-to-one correspondence. The number $3 \cdot 5$ corresponds to the subset $\{2,3\}$, for example.

Furthermore, under this one-to-one correspondence, the $|$ relation plays exactly the same role as the $\subseteq$ relation. Checking, for example, that $3 \cdot 5 \mid 2 \cdot 3 \cdot 5$ is the same as checking that all of the factors on the left are included on the right, while checking that $\{2,3\} \subseteq \{1,2,3\}$ involves checking that all of the elements on the left are included on the right. This explains why f is order-preserving.

The proof of the following theorem generalizes this observation.

Theorem 2.3 *Let $q = p_1 p_2 \cdots p_n$ be the product of the first n primes, and let $F \subseteq \mathbf{N}$ be the set of all factors of q. Then $(F, |) \cong (\mathcal{P}(\{1, 2, \cdots, n\}), \subseteq)$.*

Proof Define a function $f\colon \mathcal{P}(\{1,2,\dots,n\}) \longrightarrow F$ by setting $f(X)$ to be the product of all the p_i's with $i \in X$. For example,

$$f(\{1,3\}) = p_1p_3 = 2 \cdot 5 = 10.$$

Also, define $f(\emptyset) = 1$. In product notation,

$$f(X) = \prod_{i \in X} p_i$$

expresses the fact that $f(X)$ is the product of all the p_i's having $i \in X$. We claim that f is a one-to-one correspondence.

Proof that f is a one-to-one correspondence Let $k \in F$ be a factor of q. Since q is a product of distinct primes, any factor of q must be a product of these primes. Therefore $k = \prod_{i \in X} p_i$ for some $X \in \mathcal{P}(\{1,2,\dots,n\})$. So f is onto.

To show that f is one-to-one, suppose that X and Y are subsets of $\{1,2,\dots,n\}$ with $f(X) = f(Y)$, that is,

$$\prod_{i \in X} p_i = \prod_{i \in Y} p_i.$$

Since the prime factorization of a number is unique, X and Y must have the same elements, so $X = Y$. □

To show that f is an isomorphism, we also need to show that $X \subseteq Y$ if and only if $f(X) \mid f(Y)$. This property of isomorphisms is called "preserving" the partial ordering.

Proof that f is order-preserving Suppose that $X \subseteq Y$ for some $X, Y \subseteq \{1,2,\dots,n\}$. Then Y consists of two parts: those elements in X and those not in X. In symbols, $Y = X \cup (Y \setminus X)$ is a union of disjoint sets. Therefore

$$\begin{aligned}\prod_{i \in Y} p_i &= \left(\prod_{i \in X} p_i\right)\left(\prod_{i \in Y \setminus X} p_i\right) \\ &= \left(\prod_{i \in X} p_i\right) \cdot k\end{aligned}$$

where $k \in \mathbf{N}$. In other words, $f(Y) = k \cdot f(X)$, so $f(X) \mid f(Y)$.

Conversely, suppose $f(X) \mid f(Y)$ for some $X, Y \subseteq \{1,2,\dots,n\}$. Then

$$\prod_{i \in Y} p_i = k\left(\prod_{i \in X} p_i\right)$$

for some $k \in \mathbf{N}$. Since prime factorizations are unique, every p_i factor on the right side of this equation must also be on the left side. Therefore, $X \subseteq Y$.
□

□

The above theorem tells us that these posets will always be isomorphic, and the proof tells us *why* they are isomorphic. In short, choosing a factor of q amounts to choosing a list of prime numbers, and this choice corresponds to a choice of a subset of $\{1, 2, \ldots, n\}$. The whole proof is based on this observation.

2.5.5 Boolean Algebras ‡

The isomorphic posets described by the Hasse diagrams in Figures 2.20 and 2.22 are examples of a special kind of partially ordered set called a *Boolean algebra*. These mathematical structures highlight a beautiful connection between partial orders and propositional logic.

Let $(X, \preceq)$ be a poset. For any elements $a, b \in X$, define the *meet* of a and b (denoted $a \wedge b$) to be the greatest lower bound of a and b, if such a lower bound exists. Formally, $a \wedge b$ is an element of X with the following properties.

1. $a \wedge b \preceq a$ and $a \wedge b \preceq b$.
2. If some $x \in X$ satisfies $x \preceq a$ and $x \preceq b$, then $x \preceq a \wedge b$.

It follows from property 2 that if the meet of two elements exists, then it is unique. Similarly, we can define the *join* of $a, b \in X$ to be the element $a \vee b \in X$ satisfying

1. $a \preceq a \vee b$ and $b \preceq a \vee b$.
2. If some $x \in X$ satisfies $a \preceq x$ and $b \preceq x$, then $a \vee b \preceq x$.

if such an element exists.

Example 2.55 In Example 2.52, $4 \wedge 6 = 2$, $2 \vee 3 = 6$, but $4 \vee 6$ does not exist. In Example 2.51, $\{1\} \vee \{2, 3\} = \{1, 2, 3\}$ and $\{1\} \wedge \{2, 3\} = \emptyset$.

If every pair of elements $a, b \in X$ has both a meet and a join, then $(X, \preceq)$ is called a *lattice*. The notation for meet and join is the same as the notation for "and" and "or" in propositional logic, and this is no accident. In a lattice, meet and join satisfy the following properties, for all $a, b, c \in X$.

Commutativity: $a \wedge b = b \wedge a$ and $a \vee b = b \vee a$.

Associativity: $a \vee (b \vee c) = (a \vee b) \vee c$ and $a \wedge (b \wedge c) = (a \wedge b) \wedge c$.

Absorption: $a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$.

If, in addition, a lattice has the following three properties, it is called a *Boolean algebra.*

Distributivity: $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ and $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

Boundedness: There exist elements $\mathbf{0}, \mathbf{1} \in X$ such that $x \preceq \mathbf{1}$ and $\mathbf{0} \preceq x$ for all $x \in X$.

Complements: For any $x \in X$ there exists an element $\neg x \in X$ such that $x \wedge \neg x = \mathbf{0}$ and $x \vee \neg x = \mathbf{1}$.

From our study of set theory, we know that the poset $(\mathcal{P}(\{1, 2, \ldots, n\}), \subseteq)$ satisfies all the properties of a Boolean algebra. The correspondence is suggested by the set notation: meet is $\cap$, join is $\cup$, $\mathbf{0}$ is $\emptyset$, $\mathbf{1}$ is $\{1, 2, \ldots, n\}$, and $\neg$ is the set complement. The next theorem states that this is the only finite Boolean algebra, up to isomorphism.

Theorem 2.4 *Let X be a finite set. Suppose that the poset $(X, \preceq)$ is a Boolean algebra. Then $|X| = 2^n$ for some $n \in \mathbf{N}$, and*

$$(X, \preceq) \cong (\mathcal{P}(\{1, 2, \ldots, n\}), \subseteq).$$

The proof of this theorem is beyond the scope of this book, as are the theories of lattices and Boolean algebras. But the above definitions are still worth knowing, because they illustrate the rich connections between set theory, propositional logic, and partial orders.

Exercises 2.5

1. Refer to Definition 1.10. Show that the divisibility relation $|$ makes the set $\mathbf{N}$ of natural numbers a partially ordered set.

2. Explain why the divisibility relation $|$ does not define a partial ordering on the set $\mathbf{Z}$ of integers.

3. Consider the poset $(\mathbf{N}, |)$. Are there any minimal elements? Are there any maximal elements? Explain.

4. Let $A = \{a, b, c, \ldots, z\}$. In the poset $(\mathcal{P}(A), \subseteq)$, find a pair of incomparable elements.

5. Let W be the set of all web pages. For $x, y \in W$, let $x \; R \; y$ if you can navigate from x to y by following links. (Let's say it is always possible to "navigate" from a page to itself; just do nothing.) Explain why R is not a partial ordering.

6. Explain why the relation R on $\{0, 1, 2, 3\}$ given by

$$R = \{(0,0), (1,1), (2,2), (3,3), (0,1), (1,2), (2,3), (0,2), (1,3)\}$$

is **not** a partial ordering on $\{0, 1, 2, 3\}$. Be specific.

7. Explain why the relation R on $\{0, 1, 2, 3\}$ given by

$$R = \{(0,0), (1,1), (2,2), (3,3), (0,1), (1,2), (0,2), (2,1)\}$$

is **not** a partial ordering on $\{0, 1, 2, 3\}$. Be specific.

8. The divides relation "$|$" defines a partial ordering on the set $\{1, 2, 3, 6, 8, 10\}$. Draw the Hasse diagram for this poset. What are the maximal elements?

9. Let X be a set of different nonzero monetary values (in US or Canadian cents). In other words, $X \subseteq \mathbf{N}$. Define a relation $\models$ on X as follows. For $a, b \in X$, $a \models b$ if b can be obtained from a by adding a (possibly empty) collection of dimes (10 cents) and quarters (25 cents). So, for example, $25 \models 35$, but $25 \not\models 30$. Prove that $\models$ is a partial ordering on X.

10. Let $X = \{5, 10, 15, 20, 25, 30, 35, 40\}$, and let $\models$ be as in Problem 9.

    (a) Draw the Hasse diagram for the poset $(X, \models)$.

    (b) List all minimal elements of $(X, \models)$.

    (c) Give a pair of incomparable elements in $(X, \models)$.

11. Let X be the following set (of sets of letters).

$$X = \{\{b\}, \{b, e\}, \{b, r\}, \{b, e, r\}, \{a, r\}, \{b, a, r\}, \{b, e, a, r, s\}\}$$

Then X is a partially ordered set under the $\subseteq$ relation.

    (a) Draw the Hasse diagram for this partial ordering.

    (b) Name all minimal elements, if any exist.

    (c) Name a pair of incomparable elements, if any exist.

12. Let $A = \{a, b, c\}$. Use Hasse diagrams to describe all partial orderings on A for which a is a minimal element. (There are 10.)

13. Suppose you want to write a program that will collect information on a customer's tastes and customize web content accordingly. By monitoring online shopping habits, you are able to collect a set of pairwise preferences on a set X of products. If $x, y \in X$ are two different products, we say that $x \preceq y$ if the customer prefers y over x. (In order to satisfy the reflexive property, we stipulate that $x \preceq x$ for all $x \in X$.) Suppose you know the following things about your customer.

Customer prefers:	Over:
lettuce	broccoli
cabbage	broccoli
tomatoes	cabbage
carrots	cabbage
carrots	lettuce
asparagus	lettuce
mushrooms	tomatoes
corn	tomatoes
corn	carrots
eggplant	carrots
eggplant	asparagus
onions	mushrooms
onions	corn

In order for $(X, \preceq)$ to be a poset, we must also assume that the customer's preferences are transitive.

(a) Draw the Hasse diagram for $(X, \preceq)$.

(b) What is/are the customer's favorite vegetable(s)? [i.e., what are the maximal element(s)?] What is/are the least favorite?

(c) Use topological sorting to rank order these vegetables according to the customer's preferences. Is the ranking unique?

14. A *partition* of a positive integer n is a list of positive integers $a_1, a_2, \ldots, a_k$ such that $a_1 + a_2 + \cdots + a_k = n$. For example, the following are distinct partitions of 5.

$$5 \qquad 1,1,1,2 \qquad 1,2,2 \qquad 1,1,1,1,1$$

The order of the list doesn't matter; $1, 2, 2$ is the same partition as $2, 1, 2$. There is a natural partial ordering on the set of partitions of n: if P_1 and P_2 are partitions, define $P_1 \preceq P_2$ if P_1 can be obtained by combining parts of P_2. For example, $1, 2, 2 \preceq 1, 1, 1, 1, 1$ because $1, 2, 2 = 1, 1+1, 1+1$. On the other hand, $2, 3$ and $1, 4$ are incomparable elements in this poset.

(a) Write the partitions of 6 in a Hasse diagram. (There are 11 partitions of 6.)

(b) Is this a total ordering? Why or why not?

15. Let $X = \{1, 2, 3, 4\}$. Draw the Hasse diagram for the poset $(\mathcal{P}(X), \subseteq)$.

16. Let $F \subseteq \mathbf{N}$ be the set of all factors of 210. Draw the Hasse diagrams for $(F, |)$ and $(\mathcal{P}(\{1,2,3,4\}), \subseteq)$ in a way that shows these two posets are isomorphic.

17. Let $A \subseteq \mathbf{N}$ be the set of all factors of 12, and let $B \subseteq \mathbf{N}$ be the set of all factors of n. Find a natural number $n \neq 12$ so that $(A, |) \cong (B, |)$. Give a table of values for the one-to-one correspondence that describes the isomorphism.

18. Let B be the set of all four-digit binary strings, that is,

$$B = \{0000, 0001, 0010, 0011, \dots, 1110, 1111\}.$$

Define a relation $\lhd$ on B as follows: Let $x, y \in B$, where $x = x_1x_2x_3x_4$ and $y = y_1y_2y_3y_4$. We say that $x \lhd y$ if $x_i \leq y_i$ for $i = 1, 2, 3, 4$. In other words, $x \lhd y$ if y has a 1 in every position where x does. So, for example, $0101 \lhd 0111$ and $0000 \lhd 0011$, but $1010 \not\lhd 0111$. The relation $\lhd$ is called the *bitwise* $\leq$. Show that $(B, \lhd)$ is a poset.

19. Prove that $(B, \lhd) \cong (\mathcal{P}(\{1,2,3,4\}), \subseteq)$.

20. In $(B, \lhd)$, give a counterexample to show that

$0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 0011, 1100, 1101, 1110, 1111$

is not a valid topological sort of the elements of B.

21. Perform a topological sort on the elements of B.

22. Let $F \subseteq \mathbf{N}$ be the set of all factors of 210. In the poset $(F, |)$, find the following.

    (a) $30 \wedge 21$, the meet of 30 and 21.

    (b) $35 \vee 15$, join of 35 and 15.

    (c) $2 \wedge 7$.

    (d) $2 \vee 7$.

    (e) $\neg 30$, the complement of 30.

23. Let $W = \{a, b, c, d, e, f, g, h, i, j, k, l\}$. Define a partial order on W by the Hasse diagram in Figure 2.23.

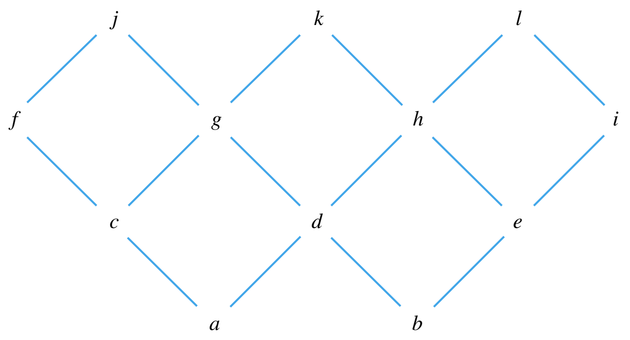


Figure 2.23 Hasse diagram for Exercise 23.

(a) Find two elements of W whose meet exists but whose join does not exist.

(b) Find two elements of W whose join exists but whose meet does not exist.

(c) Find two elements of W whose meet and join both do not exist.

24. Let $m, n \in \mathbf{Z}$ and suppose $m \leq n$. In the poset $(\mathbf{Z}, \leq)$, what are $m \wedge n$ and $m \vee n$?

25. Let T be the set of all two-digit ternary strings, that is,

$$T = \{00, 01, 02, 10, 11, 12, 20, 21, 22\}.$$

Consider the poset $(T, \triangleleft)$, where $\triangleleft$ is the bitwise $\leq$. Let $F \subseteq \mathbf{N}$ be the set of all factors of 36. Draw Hasse diagrams to show that $(T, \triangleleft) \cong (F, |)$.

26. Consider the poset $(T, \triangleleft)$ of the previous problem. This poset is, in fact, a lattice.

(a) Theorem 2.4 can be used to show that $(T, \triangleleft)$ is not a Boolean algebra. How?

(b) Find an element of $(T, \triangleleft)$ that has no complement. Explain why.

27. Let P be a set of posets. Prove that $\cong$ is an equivalence relation on P.

2.6 Graph Theory

We began this chapter with an informal introduction to graphs, and we have seen several ways to model mathematical relationships with them. We have also studied how sets, functions, and relations express these ideas in formal mathematical language. Now that we have developed this new machinery, we can revisit graphs from a more rigorous perspective. In this section we explore how to prove theorems about graphs. Some of the topics in this section—in particular, some of the proofs—will be quite challenging, because the viewpoint here will be more theoretical than in preceding sections. Fasten your seat belts.

2.6.1 Graphs: Formal Definitions

In Section 2.1, we gave an informal description of a graph in terms of dots and lines. Now that we have discussed sets and functions, we can give a mathematical definition. Warning: giving a definition of a graph is tricky; choices need to be made, and other books might make different choices.

Definition 2.10 A *directed graph* G is a finite set of vertices V_G and a finite set of edges E_G, along with a function $i\colon E_G \longrightarrow V_G \times V_G$. For any edge $e \in E_G$, if $i(e) = (a, b)$, we say that edge e *joins* vertex a to vertex b.

Note that this definition allows for loops (when $a = b$) and multiple edges (when $i(e_1) = i(e_2)$ for $e_1 \neq e_2$).[4] The definition of an undirected graph differs only in the way the word "join" is defined.

Definition 2.11 An *undirected graph* G is a finite set of vertices V_G and a finite set of edges E_G, along with a function $i\colon E_G \longrightarrow V_G \times V_G$. For any edge $e \in E_G$, if $i(e) = (a, b)$, we say that vertices a and b are *joined* by edge e, or equivalently, e *joins* a to b and e *joins* b to a. (Here it is possible that $a = b$; if $i(e) = (a, a)$, then e is a loop *joining* a to itself.)

While these definitions are mathematically rigorous, they can be hard to conceptualize. It is still important to think of a graph as a picture: each vertex corresponds to a dot, and each edge corresponds to an arrow (for directed graphs) or a line (for undirected graphs).

2.6.2 Isomorphisms of Graphs

What really matters about a graph is the relationship it describes among its vertices. You can draw a given graph any way you choose, as long as you connect the vertices correctly. Two graphs that share the same fundamental structure are called *isomorphic*. The precise definition is similar to the definition

4. Some authors call this a "multigraph."

of isomorphic posets. Note that, since Definitions 2.10 and 2.11 both stipulate the meaning of the word "join," the following definition can apply to directed and undirected graphs simultaneously.

Definition 2.12 Let G be a graph with vertex set V_G and edge set E_G, and let H be a graph with vertex set V_H and edge set E_H. Then G is *isomorphic* to H if there are one-to-one correspondences

$$\alpha : V_G \longrightarrow V_H \quad \text{and} \quad \beta : E_G \longrightarrow E_H$$

such that, for any edge $e \in E_G$,

$$e \text{ joins vertex } v \text{ to vertex } w \quad \Leftrightarrow \quad \beta(e) \text{ joins vertex } \alpha(v) \text{ to vertex } \alpha(w).$$

In this case, we write $G \cong H$.

For example, the graphs in Figure 2.24 are isomorphic. The one-to-one correspondence of vertices is given by $\alpha(x_i) = y_i$, and the correspondence of edges is given by $\beta(a_i) = b_i$. By inspecting the graphs, it is evident that the criterion

$$a_i \text{ joins vertex } x_j \text{ to vertex } x_k \quad \Leftrightarrow \quad b_i \text{ joins vertex } y_j \text{ to vertex } y_k$$

holds for all i, j, and k.

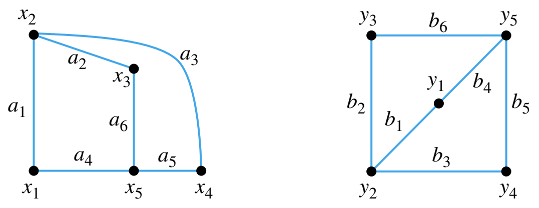


Figure 2.24 Two isomorphic graphs.

Definition 2.12 is quite complicated, but it applies to any graph. For graphs without multiple edges, the following theorem gives a simpler condition to check.

Theorem 2.5 *Let G and H be graphs without multiple edges, with vertex sets V_G and V_H, respectively. For vertices x and y, write $x \; R \; y$ if an edge joins x to y. If there is a one-to-one correspondence $f\colon V_G \longrightarrow V_H$ with the property that*

$$x \; R \; y \; \Leftrightarrow \; f(x) \; R \; f(y)$$

for all $x, y \in V_G$, then $G \cong H$.

Proof We need to check that all the conditions of Definition 2.12 are satisfied. Let α be the given one-to-one correspondence $f: V_G \longrightarrow V_H$. We define $\beta: E_G \longrightarrow E_H$ as follows. Given an edge $e \in E_G$, let x and y be the vertices that e joins, so $x \; R \; y$. Define $\beta(e) \in E_H$ to be the edge between $f(x)$ and $f(y)$. We know there is an edge joining $f(x)$ to $f(y)$ because $x \; R \; y \Rightarrow f(x) \; R \; f(y)$. Since H has no multiple edges, there is only one such edge. So β is well-defined.

To show that β is onto, let $a \in E_H$ be an edge of H, and let v and w be the vertices that a joins. Then there exist $p, q \in V_G$ such that $f(p) = v$ and $f(q) = w$, since f is onto. Furthermore, there is an edge joining p to q, since $f(p) \; R \; f(q)$. So this edge in E_G maps to a via β.

To show that β is one-to-one, suppose $e_1, e_2 \in E_G$ with $\beta(e_1) = \beta(e_2)$. Let x_1, y_1 and x_2, y_2 be the vertices joined by e_1 and e_2, respectively. Then $\beta(e_1)$ joins $f(x_1)$ to $f(y_1)$, and it also joins $f(x_2)$ to $f(y_2)$. Therefore $f(x_1) = f(x_2)$ and $f(y_1) = f(y_2)$, so $x_1 = y_1$ and $x_2 = y_2$, since f is one-to-one. Since G has no multiple edges, $e_1 = e_2$.

We have shown that β is a one-to-one correspondence. By the way we defined β,

$$e \text{ joins vertex } v \text{ to vertex } w \quad \Leftrightarrow \quad \beta(e) \text{ joins vertex } \alpha(v) \text{ to vertex } \alpha(w).$$

so $G \cong H$. □

It is usually tricky to prove that two graphs are isomorphic, but it is less tricky to use Theorem 2.5 than Definition 2.12. To check that the two graphs in Figure 2.24 are isomorphic using the theorem, we only need to define a function f on on vertices by $f(x_i) = y_i$. Since each pair of vertices defines at most one edge, there is no need for a separate function on edges. Of course, we must also check that f is a one-to-one correspondence, and that an edge joins x_i to x_j whenever an edge joins y_i to y_j.

It follows immediately from the definition that isomorphic graphs must have the same number of vertices and the same number of edges. It is also true, but harder to show, that corresponding vertices of isomorphic graphs must have the same degree. This is left as Exercise 1.

In fact, an isomorphism between two graphs guarantees that the graphs share any property having to do with the structure of the graph. For example, if $G \cong H$, and G has an Euler circuit, then so does H. If G is planar, so is H. The only differences between isomorphic graphs lie in the way they are labeled and drawn.

2.6.3 Degree Counting

In Section 2.1, we stated several observations of Euler, which we now make precise with modern statements and proofs. The first theorem describes a useful condition on the degrees of the vertices in any graph. First we must be precise about the meaning of "degree."

Definition 2.13 Let G be an undirected graph, and let $x \in V_G$ be a vertex. Let D_1 be the set of all edges $e \in E_G$ such that $i(e) = (x, b)$ for some b, and let D_2 be the set of all edges $e \in E_G$ such that $i(e) = (a, x)$ for some a. Then the *degree* of x is $|D_1| + |D_2|$.

In other words, in an undirected graph, the degree of a vertex is the number of edges that join to it, where loops are counted twice. It is left as an exercise to write down similar definitions for the *indegree* and *outdegree* of a vertex in a directed graph.

Theorem 2.6 *Let G be an undirected graph. The sum of the degrees of the vertices of G equals twice the number of edges in G.*

Proof Since each edge joins two vertices (or possibly a single vertex to itself), each edge contributes 2 to the sum of the degrees of the vertices. □

This simple theorem is surprisingly useful. In the discussion of the Königsberg bridges, it tells us that it is impossible to have exactly one island with an odd number of bridges to it: if one vertex had odd degree and the rest even, the sum of the degrees would be odd, contradicting the theorem. By the same reasoning, the number of odd-degree vertices must be even.

Example 2.56 There are 11 teams in a broomball league. The league organizer decides that each team will play five games against different opponents. Explain why this idea won't work.

Solution: The situation can be modeled by a graph. The 11 teams are the vertices, and the edges join pairs of teams that will play each other. Since the organizer wants each team to play five games, the degree of each vertex is 5. This means that the sum of the degrees of the vertices is 55, contradicting Theorem 2.6. ◊

One thing to notice about this problem is that we were able to model and solve this problem using graphs, but we never had to actually try to draw the graph. Theorems (like 2.6) that tell you when you *can't* do something can save you a lot of work.

2.6.4 Euler Paths and Circuits

Recall the definitions of paths and circuits.

Definition 2.14 Let G be a graph. A *path* in G is a sequence

$$v_0, e_1, v_1, e_2, v_2, \dots , v_{n-1}, e_n, v_n$$

of vertices v_i and edges e_i such that edge e_i joins vertex v_{i-1} to vertex v_i, where $n \geq 1$. A *circuit* is path with $v_0 = v_n$. A path or a circuit is called *simple* if $e_1, e_2, \dots , e_n$ are all distinct. Recall that a graph is *connected* if there is a path connecting any two pairs of vertices.

An Euler path (resp. Euler circuit) in a graph G is a simple path (resp. simple circuit) using all the edges of G. Euler's theorems on these constructions are interesting because they describe local conditions (the degrees of vertices) that completely determine the existence of a global object (an Euler path or circuit). Such theorems should surprise us; they are amazing in the same way that taking the temperature of the area under our tongue can tell us that our immune system is fighting off a virus. Euler managed to discover the perfect "symptom" to test.

Theorem 2.7 *If all the vertices of a connected graph G have even degree, then G has an Euler circuit.*

Proof Choose any vertex v of G. Follow a sequence of contiguous edges and vertices, starting with v, without repeating any edges. Since every vertex has even degree, this sequence will never get stuck at any vertex $w \neq v$, because any edge we follow in to w will have another edge to follow out from w. Since there are a finite number of edges, this sequence must end, so it must produce a circuit ending at v. Call this circuit C_1.

If C_1 contains all the edges of G, we are done. If not, remove the edges of C_1 from G. The resulting graph G' will still have all vertices of even degree, because every edge in C_1 going in to a vertex had a matching edge going out. Now repeat the construction of the previous paragraph on G' to get another circuit C_2. Continue repeating this construction, producing circuits $C_3, C_4, \dots , C_k$, until all the edges have been used.

By construction, every edge of G appears exactly once in the list of circuits $C_1, C_2, \dots , C_k$. Since G is connected, every circuit C_i must share a vertex with some other circuit C_j. We can therefore combine any such pair C_i, C_j of circuits into a new circuit by starting at this common vertex, following circuit C_i, and then following C_j. This combining process reduces the number of circuits on our list by one. Continue until one circuit remains. By construction, this remaining circuit is an Euler circuit. □

The preceding is an example of a *constructive* proof; it describes how to construct the hypothesized object. As such, this proof has added value because it gives us a way to tell a computer how to produce an Euler circuit. The kind of exact language required to write a proof is similar to the kind of instructions you need to give to a computer to get it to do what you want.

The converse of Theorem 2.7 is also true, and easier to prove.

Theorem 2.8 *If a graph G has an Euler circuit, then all the vertices of G have even degree.*

Proof Let v be a vertex in G. If the degree of v is zero, there is nothing to prove. If the degree of v is nonzero, then some edges on the Euler circuit touch it, and since the Euler circuit contains all of G's edges, the number of touches by these edges equals the degree of v. If we traverse the circuit, starting and ending at v, we must leave v the same number of times as we return to v. Therefore the number of touches by an edge in the Euler circuit is even. □

Euler's observations about paths have related proofs. These are left as exercises.

2.6.5 Hamilton Paths and Circuits

Euler's theorems tell us just about all there is to know about the existence of paths and circuits that visit every edge of a graph exactly once. So it is natural to ask the same questions about paths and circuits that visit every *vertex* of a graph exactly once. These constructions are named in honor of the nineteenth-century Irish mathematician William Rowan Hamilton.

Definition 2.15 A *Hamilton path* in a graph G is a path

$$v_0, e_1, v_1, e_2, \dots, e_n, v_n$$

such that $v_0, v_1, \dots, v_n$ is a duplicate-free list of all the vertices in G. A Hamilton circuit is a circuit $v_0, e_1, v_1, e_2, \dots, e_n, v_n, e_{n+1}, v_0$, where $v_0, e_1, v_1, e_2, \dots, e_n, v_n$ is a Hamilton path.

Although it is easy to tell whether a given graph has an Euler circuit, it is actually quite difficult, in general, to determine whether a graph has a Hamilton circuit. For graphs without too many vertices, it usually isn't hard to find a Hamilton circuit when it is possible to do so. However, for graphs that don't have Hamilton circuits, it is often very tricky to prove that this is the case. The next two examples illustrate this phenomenon.

Example 2.57 Find a Hamilton circuit in the graph in Figure 2.25.

Solution: One possible Hamilton circuit can be found as follows. (Try this with a piece of tracing paper.) Starting at vertex a, travel counterclockwise around

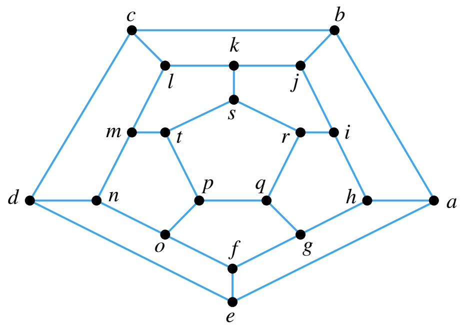


Figure 2.25 Can you find a Hamilton circuit in this graph? See Example 2.57.

the outer ring. Notice that you can't go back to a until you have visited all the vertices, so you will need to change direction when you get to e. Go from e to f, and then go around clockwise from there, but you must change direction again at i in order to leave room to get back to a. Then it is easy to see how to finish. The sequence of vertices

$$a, b, c, d, e, f, o, n, m, l, k, j, i, r, s, t, p, q, g, h, a$$

forms a Hamilton circuit. ◊

Example 2.58 Prove that the graph in Figure 2.26 has no Hamilton circuit.

Proof Notice that this graph has five vertices of degree 2: f, h, j, l, n. Any Hamilton circuit must pass through these vertices, so it must include the edges that touch these vertices. But this makes it impossible to leave the outer ring: we can't add edges to those forced by the vertices f, h, j, l, n because that would

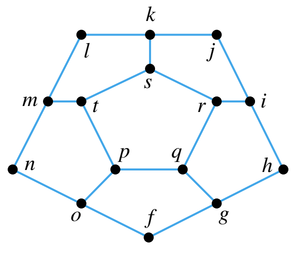


Figure 2.26 Why is there no Hamilton circuit in this graph? See Example 2.58.

involve including three edges at some vertex, which can't happen in a Hamilton circuit: if a single vertex touches three edges in a path, the vertex must occur twice in that path. □

The previous two arguments are *ad hoc*; they don't generalize well to other examples the way Euler's theorems do. There are existence theorems about Hamilton circuits, but they are quite difficult to prove, and they don't give a complete characterization of graphs containing Hamilton circuits. In fact, the problem of finding a Hamilton circuit in a graph belongs to a famous class of problems called *NP-complete*. Roughly speaking, these are problems that are difficult to solve, but whose solutions are easy to check.[5] It may not be clear whether a given graph has a Hamilton circuit, but it is really easy to check the answer of a student who claims to have found one.

Euler and Hamilton circuits raise interesting mathematical questions, but they also have several important applications. For example, consider a graph model of a network of roads (edges) connecting various points of interest (vertices). A street cleaner would be interested in an Euler circuit, because it would provide a way to go down every street without having to retrace any steps. On the other hand, a salesperson wanting an efficient route through each point of interest might find a Hamilton circuit more useful.

2.6.6 Trees

In Section 2.1, we saw how to organize data using a binary search tree. In future chapters, we will see several other uses for trees of various types. We wrap up our look at the mathematical theory of graphs with some theorems about trees in general.

Definition 2.16 A *tree* is a graph T with a specified vertex r, called the *root*, with the property that, for any vertex v in T $(v \neq r)$, there is a unique simple path from r to v.

This definition works for directed or undirected graphs. Usually we draw trees as undirected graphs, but we often draw the tree with the root at the top and an implicit downward direction on all the edges. The next theorem gives an alternate characterization of a tree.

Theorem 2.9 *Let G be an undirected graph, and let $r \in G$. Then G is a tree with root r if and only if G is connected and has no simple circuits.*

5. We will say more about NP-completeness in Chapter 5.

Proof ($\Rightarrow$) Suppose that G is an undirected tree with root r. Let a, b be two vertices in G. By Definition 2.16, there are paths from r to a and r to b. Therefore, there is a path from a to b via r, so G is connected.

Suppose, to the contrary, that

$$v_0, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_0$$

is a simple circuit in G. We can relabel this circuit, if necessary, so that $v_0 \neq r$. If $r = v_i$ for some i, then

$$v_0, e_1, \ldots, r \quad \text{and} \quad r, e_{i+1}, \ldots, e_k, v_0$$

are two different simple paths from v_0 to r, contradicting the definition of a tree. If $r \neq v_i$ for any i, then there is a simple path from r to v_0, and combining this path with the sequence

$$e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_0$$

yields another simple path from r to v_0, a contradiction.

($\Leftarrow$) Now suppose that G is a connected undirected graph with no simple circuits. Let $v \neq r$ be a vertex in G. Since G is connected, there is a path from r to v. If this path does not repeat any vertices, then it does not repeat any edges, so it is simple. If this path has the form

$$r, e_1, \ldots, e_i, a, e_{i+1}, \ldots, e_k, a, e_{k+1}, \ldots, e_n, v$$

for some vertex a, then we can replace it with a shorter path

$$r, e_1, \ldots, e_i, a, e_{k+1}, \ldots, e_n, v$$

that still goes from r to v. Furthermore, we can repeat this shortening procedure until there are no repeated vertices. So there is a simple path from r to v. To show that G is a tree with root r, we need to show that this path is unique. But if there were two distinct simple paths

$$r, e_1, v_1, \ldots, v_{n-1}, e_n, v \quad \text{and} \quad r, d_1, w_1, \ldots, w_{m-1}, d_m, v$$

from r to v, we could combine them to form a circuit

$$r, e_1, v_1, \ldots, v_{n-1}, e_n, v, d_m, w_{m-1}, \ldots, w_1, d_1, r$$

in G. While this circuit may not be simple, it must contain a simple circuit, since there is at least one d_i that differs from all the e_j's. (This detail is left as Exercise 19.) This contradicts that G has no simple circuits. $\square$

An important consequence of Theorem 2.9 is that you can choose any vertex to be the root of an undirected tree. Figure 2.27 shows the same undirected tree

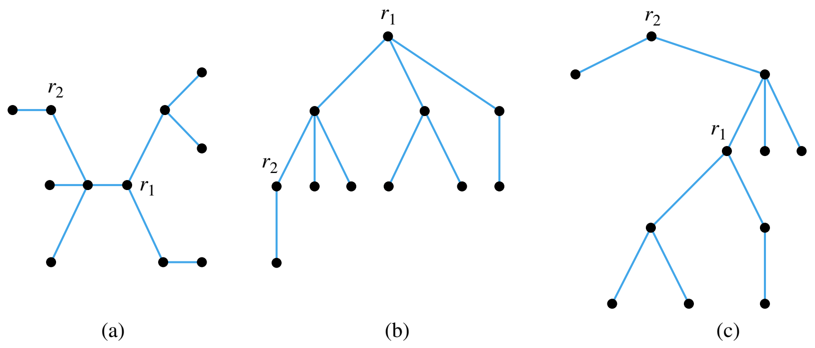


Figure 2.27 You can choose the root in an undirected tree.

drawn three different ways: first (a) without a root, then (b) with root r_1, and finally (c) with root r_2. Imagine that the graph is made of beads and string. In graph (a), pick up the bead that you want to be the root (r_1 or r_2), and let the rest of the beads hang down to get graphs (b) or (c). This process is intuitively obvious, but the previous proof gives us a solid mathematical justification: if a graph G is connected and has no simple circuits, it is a tree with root r, for any vertex r in G. And any undirected tree is a connected graph with no simple circuits, so any vertex can be the root. A corollary follows from this observation.

Corollary 2.1 In an undirected tree, there is a unique simple path between any two vertices in the tree.

Proof Let a, b be two vertices in an undirected tree T. Then T can be viewed as a tree with root a, so by Definition 2.16 there is a unique simple path from a to b. □

The characterization of trees in Definition 2.16 implies quite a lot about the structure of these graphs. Let v be a nonroot vertex of a tree T. Since there is a unique simple path from the root r of T to v, there is a well-defined natural number $d(v)$ that equals the number of edges in this path. Call the number $d(v)$ the *depth* of the vertex v. By convention, $d(r) = 0$. The *height* of the tree T is the maximum value of $d(v)$ over all the vertices v in T. For example, the tree in Figure 2.27(c) has height 4, and $d(r_1) = 2$ in this tree. Of course, the depth function and the height of a tree depend on the choice of root; the tree in Figure 2.27(b) has height 3.

Another implication of the definition is that we can make an undirected tree into a tree in a well-defined way, by choosing a direction on each edge away from the root. More precisely, start at the root, and direct each edge outward. These edges will end in vertices of depth 1, and all the vertices of depth 1 will now have directed edges to them. Repeat this process, forming directed edges to all the vertices of depth 2, then 3, etc., until all the vertices (and hence all the edges) have been covered. Thus, every edge will point from a vertex of depth i to a vertex of depth $i+1$. This construction explains why it isn't really necessary to draw the directions on the edges of a tree, as long as we designate a root. It also helps us prove the following fact.

Theorem 2.10 *Let T be a tree with n vertices. Then T has $n-1$ edges.*

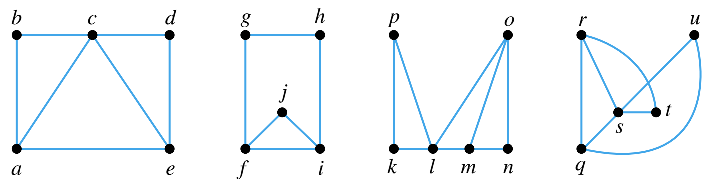
Proof Suppose T is a tree with n vertices, and let r be the root of T. By the above construction, we can make T into a directed tree, if necessary, without changing the number of edges and vertices. Every edge must point to a nonroot vertex. Furthermore, every nonroot vertex has a single edge pointing to it, since there is a unique path to each vertex from the root. So the number of edges equals the number of nonroot vertices: $n-1$. □

Exercises 2.6

1. See Definitions 2.12 and 2.13. Let G and H be isomorphic, undirected graphs with vertex bijection α and edge bijection β. Let $x \in V_G$ be a vertex in G. Prove that the degree of x equals the degree of $\alpha(x)$.

2. Draw two nonisomorphic, undirected graphs, each having four vertices and four edges. Explain how you know they are not isomorphic.

3. Find a pair, G and H, of isomorphic graphs among the four graphs below, and give the one-to-one correspondence $\alpha\colon V_G \longrightarrow V_H$ of vertices. (For each vertex of G, tell which vertex of H to which it corresponds.)



4. Give a rigorous definition for the *indegree* and *outdegree* of a vertex in a directed graph. (Model your definition on Definition 2.13.)

5. Let X be a finite set, and let $\mathcal{P}(X)$ be the power set of X. Let G be the graph whose vertices represent the elements of $\mathcal{P}(X)$, where A and

B are joined by an edge if $A \cap B = \emptyset$. Similarly, let H be the graph with a vertex for each element of $\mathcal{P}(X)$, but where A and B share an edge if $A \cup B = X$. Prove that G is isomorphic to H.

6. Let (X_1, R_1) and (X_2, R_2) be isomorphic partial orders. Let G_1 and G_2 be their respective associated graphs, as defined in Definition 2.5. Prove that G_1 and G_2 are isomorphic graphs.

7. Let R_1 and R_2 be relations on a set X. Give a reasonable definition for an isomorphism of relations.

8. Between 1970 and 1975, the National Football League was divided into two conferences, with 13 teams in each conference. Each team played 14 games in a season. Would it have been possible for each team to play 11 games against teams from its own conference and 3 games against teams from the other conference? Use a graph model to answer this question (without drawing the graph).

9. A league of 10 teams is playing a "round-robin" style tournament, where each team plays every other team exactly once. How many games total need to be played? Justify your answer using a graph model—say what the vertices and edges of your graph represent, and what (if any) theorems you use.

10. The *complete* graph on n vertices (denoted K_n) is the undirected graph with exactly one edge between every pair of distinct vertices. Use Theorem 2.6 to derive a formula for the number of edges in K_n.

11. A truncated icosidodecahedron (also known as a "great rhombicosidodecahedron" or "omnitruncated icosidodecahedron") is a polyhedron with 120 vertices. Each vertex looks the same: a square, a hexagon, and a decagon come together at each vertex. How many edges does a truncated icosidodecahedron have? Explain how you arrive at your answer. (Note: the picture in Figure 2.28 doesn't show the vertices or edges on the back of the polyhedron.)

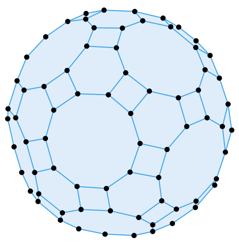


Figure 2.28 A truncated icosidodecahedron. See Exercises 11 and 14.

12. Prove the following theorem of Euler.

    Theorem 2.11 *If a connected graph has exactly two vertices v and w of odd degree, then there is an Euler path from v to w.*

    Hint: Add an edge, and use Theorem 2.7.

13. Prove the following theorem of Euler.

    Theorem 2.12 *If a graph has more than two vertices of odd degree, it does not have an Euler path.*

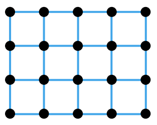
    Hint: Try a proof by contradiction, and model your argument on the proof of Theorem 2.8.

14. Is there an Euler path on the graph formed by the vertices and edges of a truncated icosidodecahedron? Why or why not?

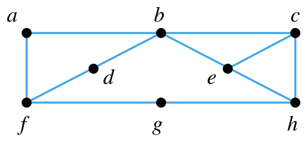
15. Prove that K_n, the complete graph on n vertices, has a Hamilton circuit for all $n \geq 3$.

16. For what values of n does K_n have an Euler circuit? Explain.

17. Find a Hamilton circuit in the following graph.



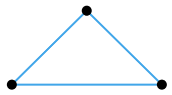
18. Explain why the following graph does not have a Hamilton circuit.



19. Finish the proof of Theorem 2.9 by proving that, if a graph G has a circuit with some edge e that differs from all the other edges in the circuit, then G has a simple circuit. (Hint: Focus on the sequence of vertices in the given circuit.)

20. Copy the three graphs from Figure 2.27 and label the vertices to show that the graphs are isomorphic.

21. Let G be a graph with vertex set V_G and edge set E_G. A *subgraph* of G is a graph with vertex set $V_H \subseteq V_G$ and edge set $E_H \subseteq E_G$. Write down all the subgraphs of the following graph.



22. Let G be a graph with vertex set V_G and edge set E_G. Let $V \subseteq V_G$ and $E \subseteq E_G$. Is there always a subgraph of G with vertex set V and edge set E? Explain.

23. Let G be a connected, undirected graph. Prove that there is a subgraph T of G such that T contains all the vertices of G, and T is a tree. (Such a subgraph is called a *spanning tree.*) Give a constructive proof that explains how to construct a spanning tree of a graph.

Chapter 3

Recursive Thinking

The branches of a blue spruce tree (Figure 3.1) follow an interesting pattern. The trunk of the tree is a large central stem, with branches coming off each side. These branches also have thinner stems coming off them, and so on, until you

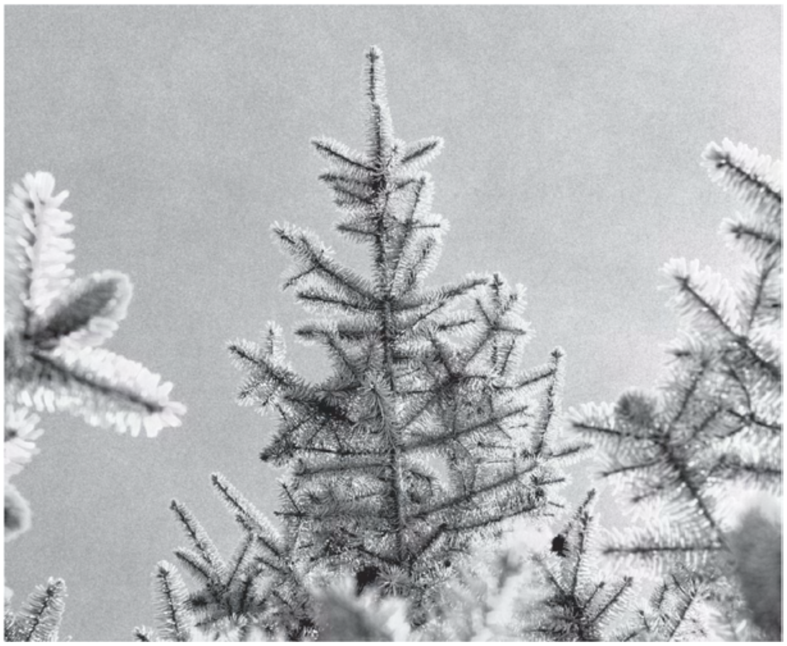


Figure 3.1 A recursive blue spruce.

reach the level of the needles. A branch looks like a smaller copy of the whole tree, and even a small twig looks like a miniature branch. This is a picture of recursion.

The natural phenomenon of recursion pervades many areas of mathematics. In this chapter, you will learn how to work with recursive structures. You will develop the ability to see recursive patterns in mathematical objects. And you will study mathematical induction, a powerful tool for proving theorems about recursive structures.

3.1 Recurrence Relations

3.1.1 Definition and Examples

The simplest and most concrete type of recursive object in mathematics is a *recurrence relation.* Suppose we wish to define a function

$$P\colon \mathbf{N} \longrightarrow \mathbf{Z}$$

that inputs a natural number and returns an integer. The easiest way to do this is to give an explicit formula:

$$P(n) = \frac{n(n+1)}{2}. \tag{3.1.1}$$

To evaluate $P(n)$ for some given n, you just plug n into the formula.

It is always nice to have an explicit formula for a function, but sometimes these are hard to come by. Sometimes a function comes up in mathematics that is natural to define recursively. Here is a second way of defining our function $P(n)$:

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ n + P(n-1) & \text{if } n > 1. \end{cases} \tag{3.1.2}$$

This is a *recursive definition* because P is defined in terms of itself: P occurs in the formula that defines P. This may seem a little sneaky, but it is perfectly legal. For any $n \in \mathbf{N}$, we can use the definition in Equation 3.1.2 to compute $P(n)$.

Example 3.1 Use Equation 3.1.2 to compute $P(5)$.

Solution: We'll compute $P(5)$ two different ways. The first approach is called "bottom-up" because we start at the bottom with $P(1)$ and work our way up to $P(5)$. By the first part of the definition, $P(1) = 1$. By the second part of the definition, with $n = 2$, we get $P(2) = 2 + P(1) = 2 + 1 = 3$. Again, by the second part with $n = 3$, $P(3) = 3 + P(2) = 3 + 3 = 6$. Repeating this process, $P(4) = 4 + 6 = 10$, and finally, $P(5) = 5 + 10 = 15$.

Alternatively, we can do a "top-down" computation. Whenever $n > 1$, we can apply the second part of the function definition to replace "$P(n)$" with "$n + P(n-1)$." This replacement justifies the first four uses of the = sign.

$$\begin{aligned} P(5) &= 5 + P(4) \\ &= 5 + 4 + P(3) \\ &= 5 + 4 + 3 + P(2) \\ &= 5 + 4 + 3 + 2 + P(1) \\ &= 5 + 4 + 3 + 2 + 1 \\ &= 15. \end{aligned}$$

The second-to-last = sign is the result of applying the first part of the definition to replace "$P(1)$" with "1." ♢

Note that the equation

$$P(n) = n + P(n-1)$$

by itself would not define a recurrence relation, because the bottom-up calculation could never get started, and the top-down calculation would never stop. A well-defined recurrence relation needs a nonrecursive *base case* that gives at least one value of the function explicitly.

3.1.2 The Fibonacci Sequence

One of the earliest examples of recursive thinking is the famous Fibonacci sequence. In the early 13th century, the Italian mathematician Leonardo Pisano[1] Fibonacci proposed the following problem. [10]

> A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair which from the second month on becomes productive?

Let $F(n)$ represent the number of pairs of rabbits present on month n. Assuming that it takes two months for the first pair of rabbits to become productive, we have

$$F(1) = F(2) = 1$$

as a base case, and the first pair of offspring show up on the third month, so $F(3) = 2$. Every month, the number of new pairs of rabbits equals the number of rabbits present two months before. So we can define a recurrence relation as follows.

1. "Pisano" means "of Pisa." In fact, if you visit Pisa, Italy, you will find a statue of Fibonacci in the Camposanto, near the famous leaning tower.

Definition 3.1 The *Fibonacci numbers* $F(n)$ satisfy the following recurrence relation:

$$F(n) = \begin{cases} 1 & \text{if } n = 1 \text{ or } n = 2 \\ F(n-1) + F(n-2) & \text{if } n > 2. \end{cases}$$

The sequence $F(1), F(2), F(3), \dots$ is called the *Fibonacci sequence.*

The second part of this definition, $F(n) = F(n-1) + F(n-2)$, is the recursive part. The $F(n-1)$ term represents the number of rabbit pairs present the previous month. The $F(n-2)$ term represents the number of new rabbit pairs—equal to the number of rabbit pairs present two months ago. Therefore the sum of these two terms is the total number of rabbit pairs present for month n, the current month. The first few terms of the famous Fibonacci sequence are then

$$1, 1, 2, 3, 5, 8, 13, 21, 34, \dots .$$

Despite its somewhat whimsical origins, the Fibonacci sequence has a remarkable number of applications. This sequence of numbers is found in a variety of contexts, including plant growth, stock prices, architecture, music, and drainage patterns.

Example 3.2 The pine cone in Figure 3.2 contains Fibonacci numbers. Notice the spiral patterns emanating out from the center, both in clockwise and counterclockwise directions. There are $F(6) = 8$ counterclockwise spirals and $F(7) = 13$ clockwise spirals. This isn't merely coincidental; Fibonacci patterns are often found in nature where growth occurs in stages, with each successive stage dependent on previous stages. The recursive nature of plant growth is reflected in the presence of recursively defined sequences. The study of leaf patterns in plants is called *phyllotaxis*, and the more general study of how shapes form in living things is called *morphogenesis*. These studies apply recursive ideas.

3.1.3 Modeling with Recurrence Relations

Fibonacci's rabbit example shows how to think recursively about a problem by describing it with a recurrence relation. Remember that any recurrence relation has two parts: a base case that describes some initial conditions, and a recursive case that describes a future value in terms of previous values. Armed with this way of thinking, we can model other problems using recurrence relations.

Example 3.3 Ursula the Usurer lends money at outrageous rates of interest. She demands to be paid 10% interest *per week* on a loan, compounded weekly. Suppose you borrow \$500 from Ursula. If you wait four weeks to pay her back, how much will you owe?

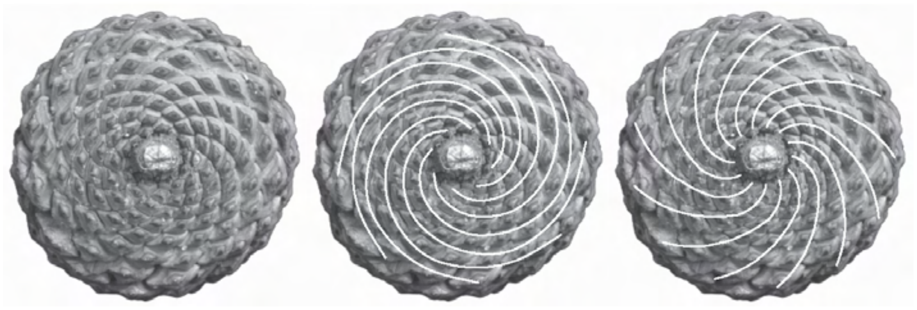


Figure 3.2 Fibonacci numbers appear in many different kinds of plant growth, including this pine cone. Image courtesy of Pau Atela and Christophe Golé. [3]

Solution: Let $M(n)$ be how much money you owe Ursula on the nth week. Initially, you owe \$500, so $M(0) = 500$. Each subsequent week, the amount you owe increases by 10%. Therefore, we have the following recurrence relation:

$$M(n) = \begin{cases} 500 & \text{if } n = 0 \\ 1.10 \cdot M(n-1) & \text{if } n > 0. \end{cases}$$

Then the amount you owe after four weeks is

$$\begin{aligned} M(4) &= 1.10 \cdot M(3) \\ &= 1.10 \cdot 1.10 \cdot M(2) \\ &= 1.10 \cdot 1.10 \cdot 1.10 \cdot M(1) \\ &= 1.10 \cdot 1.10 \cdot 1.10 \cdot 1.10 \cdot M(0) \\ &= 1.10 \cdot 1.10 \cdot 1.10 \cdot 1.10 \cdot 500 \\ &= \$732.05. \end{aligned}$$

$\diamondsuit$

Example 3.4 If you have ever tried making patterns with a collection of coins, you have probably noticed that you can make hexagons in a natural way by packing circles as tightly as possible. Figure 3.3 shows how 19 circles fit into a hexagonal shape with 3 circles on each edge. Let $H(n)$ be the number of circles you need to form a hexagon with n circles on each edge. From Figure 3.3, it is clear that $H(2) = 7$ and $H(3) = 19$. Find a recurrence relation for $H(n)$.

Solution: The base case of the recurrence relation could be $H(2) = 7$, or we could agree that $H(1) = 1$, representing a "trivial" hexagon with just one circle.

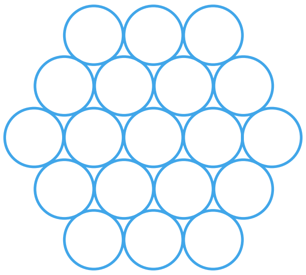


Figure 3.3 Circles packed to form a hexagon.

For the recursive case, we need to describe how to make a hexagonal pattern of edge size n from a hexagonal pattern of edge size $n-1$. We would need to add six edges, each made up of n circles, but each circle on a vertex of the new hexagon will be included in two edges. Thus the number of circles added will be $6n-6$; subtracting 6 accounts for double-counting the circles on the vertices. This gives the following recurrence relation:

$$H(n) = \begin{cases} 1 & \text{if } n = 1 \\ H(n-1) + 6n - 6 & \text{if } n > 1. \end{cases}$$

If you don't quite believe in this formula, try some calculations: $H(2) = H(1) + 6 \cdot 2 - 6 = 7$, $H(3) = H(2) + 6 \cdot 3 - 6 = 19$, etc. ♢

To construct a recurrence relation, it often helps to see how successive cases of the problem are built on previous ones, like the layers of an onion. The recurrence relation describes how to count the next layer in terms of the previous one(s). If $P(n)$ is the function we want to describe recursively, think of $P(n)$ as the general case of the problem, while $P(n-1)$ represents the next simplest case. The recursive part of the recurrence relation is then an equation of the form

$$P(n) = \text{ some function of } P(n-1) \text{ and } n$$

that describes how to add a layer to your onion. The next few examples illustrate this paradigm.

Example 3.5 Let X be a finite set with n elements. Find a recurrence relation $C(n)$ for the number of elements in the power set $\mathcal{P}(X)$.

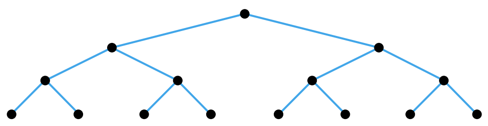
Solution: The base case is when $n = 0$ and X is the empty set, in which case $\mathcal{P}(X) = \{\emptyset\}$, so $C(0) = 1$. Now suppose $|X| = n$ for some $n > 0$. Choose some element $x \in X$ and let $X' = X \setminus \{x\}$. Then X' has $n-1$ elements, so $|\mathcal{P}(X')| = C(n-1)$. Furthermore, every subset of X is either a subset of X', or a subset of the form $U \cup \{x\}$, where $U \subseteq X'$, and these two cases are mutually exclusive. Therefore $\mathcal{P}(X)$ has twice as many elements as $\mathcal{P}(X')$. So

$$C(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2 \cdot C(n-1) & \text{if } n > 0 \end{cases}$$

is a recurrence relation for $|\mathcal{P}(X)|$. ◇

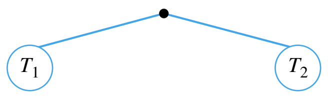
Recall that the *depth* $d(v)$ of a vertex v in a binary tree is the number of edges in the path from the root to v. The *height* of a binary tree is the maximum value of $d(v)$ over all the vertices in the tree.

Example 3.6 Call a binary tree *complete* if every leaf has depth n and every nonleaf node has two children. For example,



is a complete binary tree of height 3. Let T be a complete binary tree of height n. Find a recurrence relation $V(n)$ for the number of nodes in T.

Solution: Notice that a complete tree of height n has two complete trees of height $n-1$ inside of it.



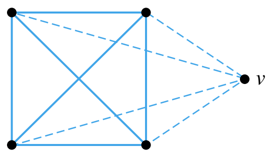
In addition to the nodes in T_1 and T_2, there is one more node: the root. Therefore a complete binary tree of height n has

$$V(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2 \cdot V(n-1) + 1 & \text{if } n > 0 \end{cases}$$

nodes. ◇

Example 3.7 Recall that the complete graph K_n on n vertices is the undirected graph that has exactly one edge between every pair of vertices. Find a recurrence relation $E(n)$ for the number of edges in K_n.

Solution: Given a complete graph on $n-1$ vertices, we can add a vertex and edges to make a complete graph on n vertices. We need to add $n-1$ new edges, because the new vertex needs to be connected to all of the vertices of the original given graph. For example, the following figure shows how to make K_5 from K_4.



The new vertex is labeled v, and the four new edges are dashed. A complete graph on one vertex has no edges, so we obtain the following recurrence relation:

$$E(n) = \begin{cases} 0 & \text{if } n = 1 \\ E(n-1) + n - 1 & \text{if } n > 1. \end{cases}$$

◇

Look back at these last three examples, and see if you can recognize the recursive way of thinking. A power set contains all the elements of a smaller power set, and more. A complete binary tree has two smaller complete binary trees inside it. And a complete graph on $n-1$ vertices can be augmented to form a complete graph on n vertices. The hard part of these problems was identifying the recursive structure in these objects; this structure made writing down the recurrence relation a fairly straightforward task.

Exercises 3.1

1. Refer to the recurrence relation for the Fibonacci sequence in Definition 3.1.

   (a) Answer Fibonacci's question by calculating $F(12)$.

   (b) Write $F(1000)$ in terms of $F(999)$ and $F(998)$.

   (c) Write $F(1000)$ in terms of $F(998)$ and $F(997)$.

2. In Fibonacci's model, rabbits live forever. The following modification of Definition 3.1 accounts for dying rabbits:

$$G(n) = \begin{cases} 0 & \text{if } n \leq 0 \\ 1 & \text{if } n = 1 \text{ or } n = 2 \\ G(n-1) + G(n-2) - G(n-8) & \text{if } n > 2. \end{cases}$$

(a) Compute $G(n)$ for $n = 1, 2, \dots, 12$.

(b) In this modified model, how long do rabbits live?

3. Consider the following recurrence relation:

$$H(n) = \begin{cases} 0 & \text{if } n \leq 0 \\ 1 & \text{if } n = 1 \text{ or } n = 2 \\ H(n-1) + H(n-2) - H(n-3) & \text{if } n > 2. \end{cases}$$

(a) Compute $H(n)$ for $n = 1, 2, \dots, 10$.

(b) Using the pattern from part (a), guess what $H(100)$ is.

4. The *Lucas numbers* $L(n)$ have almost the same definition as the Fibonacci numbers:

$$L(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3 & \text{if } n = 2 \\ L(n-1) + L(n-2) & \text{if } n > 2. \end{cases}$$

(a) How is the definition of $L(n)$ different from the definition of $F(n)$ in Definition 3.1?

(b) Compute the first 12 Lucas numbers.

5. Compute the first seven terms of the following recurrence relations:

(a) $C(n)$ of Example 3.5.

(b) $V(n)$ of Example 3.6.

(c) $E(n)$ of Example 3.7.

6. Consider the recurrence relation defined in Example 3.3. Suppose that, as in the example, you borrow \$500, but you pay her back \$100 each week. Each week, Ursula charges you 10% interest on the amount you still owe, after your \$100 payment is taken into account.

(a) Write down a recurrence relation for $M(n)$, the amount owed after n weeks.

(b) How much will you owe after four weeks?

7. Every year, Alice gets a raise of \$3,000 plus 5% of her previous year's salary. Her starting salary is \$50,000. Give a recurrence relation for $S(n)$, Alice's salary after n years, for $n \geq 0$.

8. Suppose that today (year 0) your car is worth \$10,000. Each year your car loses 10% of its value, but at the end of each year you add customizations to your car which increase its value by \$50. Write a recurrence relation to model this situation.

9. Refer to Example 3.4. Calculate $H(7)$.

10. Circles can be packed into the shape of an equilateral triangle. Let $T(n)$ be the number of circles needed to form a triangle with n circles on each edge. From Figure 3.3 (or by experimenting with coins), it is easy to see that $T(2) = 3$ and $T(3) = 6$. Write down a recurrence relation for $T(n)$.

11. Let $H(n)$ be as in Example 3.4, and let $T(n)$ be as in Exercise 10. Write $H(n)$ in terms of $T(n-1)$. Explain your reasoning. (Hint: use Figure 3.4.)

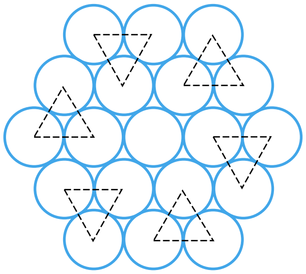


Figure 3.4 Hint for Exercise 11.

12. Recall that the factorial function is defined as

$$n! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot (n-1) \cdot n$$

and, by convention, $0! = 1$. Give a recurrence relation for $n!$.

13. Let $S(n) = 1^2 + 2^2 + \cdots + n^2$ be the sum of the first n perfect squares. Find a recurrence relation for $S(n)$.

14. The ancient Indian game of *Chaturanga*—from which the modern game of chess was apparently derived—was played on a board with 64 squares. A certain folktale tells the story of a Raja who promised a reward of one grain of rice on the first square of the board, two grains on the second square, four on the third, and so on, doubling the number of grains on each successive square.

    (a) Write a recurrence relation for $R(n)$, the number of grains of rice on the nth square.

    (b) Compute $R(64)$. Assuming a grain of rice weighs 25 milligrams, how many kilograms of rice must be placed on the 64th square?

15. Find recurrence relations that yield the following sequences:

    (a) $5, 10, 15, 20, 25, 30, \ldots$

    (b) $5, 11, 18, 26, 35, 45, \ldots$

16. Let $f\colon \mathbf{N} \longrightarrow \mathbf{R}$ be any function on the natural numbers. The sum of the first n values of $f(n)$ is written as

$$\sum_{k=1}^{n} f(k) = f(1) + f(2) + \cdots + f(n)$$

in *sigma notation.*

   (a) Write $1^2 + 2^2 + \cdots + n^2$ in sigma notation.

   (b) Give a recurrence relation for $\displaystyle\sum_{k=1}^{n} f(k)$.

17. Let $f\colon \mathbf{N} \longrightarrow \mathbf{R}$ be any function on the natural numbers. The product of the first n values of $f(n)$ is written as

$$\prod_{k=1}^{n} f(k) = f(1) \cdot f(2) \cdot \cdots \cdot f(n)$$

in *product notation.*

   (a) Write $n!$ in product notation, for $n > 0$.

   (b) Give a recurrence relation for $\displaystyle\prod_{k=1}^{n} f(k)$.

18. *Calculus required.* Use the reduction formula

$$\int x^n e^x \, dx = x^n e^x - n \int x^{n-1} e^x \, dx$$

to give a simple (noncalculus) recurrence relation for

$$I(n) = \int_0^1 x^n e^x \, dx$$

where $n \geq 0$. Make sure that your recurrence relation has a base case.

19. Suppose we model the spread of a virus in a certain population as follows. On day 1, one person is infected. On each subsequent day, each infected person gives the cold to two others.

   (a) Write down a recurrence relation for this model.

   (b) What are some of the limitations of this model? How does it fail to be realistic?

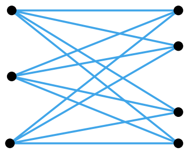
20. Let X be a set with n elements. Let $E \subseteq \mathcal{P}(X)$ be the set of all subsets of X with an even number of elements, and let $O \subseteq \mathcal{P}(X)$ be the subsets of X with an odd number of elements. Let $E(n) = |E|$ and $O(n) = |O|$.

(a) Find a recurrence relation for $E(n)$ in terms of $O(n-1)$ and $E(n-1)$.

(b) Find a recurrence relation for $O(n)$ in terms of $O(n-1)$ and $E(n-1)$.

(c) Find the first five values of $E(n)$ and $O(n)$.

21. The *complete bipartite graph* $K_{m,n}$ is the simple undirected graph with $m+n$ vertices split into two sets V_1 and V_2 ($|V_1| = m$, $|V_2| = n$) such that vertices x, y share an edge if and only in $x \in V_1$ and $y \in V_2$. For example, $K_{3,4}$ is the following graph.



(a) Find a recurrence relation for the number of edges in $K_{3,n}$.

(b) Find a recurrence relation for the number of edges in $K_{n,n}$.

3.2 Closed-Form Solutions and Induction

Although recurrence relations have a certain elegant quality, it can be tedious to compute with them. For example, armed only with Definition 3.1, it would take 998 steps to compute the Fibonacci number $F(1000)$. In this section we will explore ways to find a nonrecursive closed-form solution to a recurrence relation. Much more than appears here could be said about solving recurrence relations. The essential concept in this section is the idea of mathematical induction, the method by which we verify the correctness of a closed-form solution.

3.2.1 Guessing a Closed-Form Solution

Suppose that $P(n)$ is a function defined by a recurrence relation. We would like to have an expression

$$P(n) = \text{a nonrecursive function of } n$$

because then we could just calculate $P(n)$ by plugging n into the formula, rather than by using the recurrence relation over and over. A formula like this is called a *closed-form solution* to the recurrence relation. Recall Example 3.1: we found that the first five values of the recurrence relation

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ n + P(n-1) & \text{if } n > 1 \end{cases}$$

were $1, 3, 6, 10$, and 15. A closed-form solution to this recurrence relation is

$$P(n) = \frac{n(n+1)}{2}.$$

We can check that the values of $P(n)$ given by this nonrecursive formula match the values given by the recurrence relation:

n	1	2	3	4	5
$\frac{n(n+1)}{2}$	1	3	6	10	15

This table of values is pretty convincing evidence that we have found the right closed-form solution, but it is not a proof. Later in this section we will see how to verify that such a solution is correct, but first we will consider the problem of finding closed-form solutions.

Given a recurrence relation, the most general way of finding a closed-form solution is by guessing. Unfortunately, this is also the hardest technique to master.

Example 3.8 Find a closed-form solution for the recurrence relation from Example 3.3:

$$M(n) = \begin{cases} 500 & \text{if } n = 0 \\ 1.10 \cdot M(n-1) & \text{if } n > 0. \end{cases} \tag{3.2.1}$$

Solution: It almost always helps to write out the first few values of $M(n)$. In this case, it also helps to leave things unsimplified: the pattern is easier to see if you don't multiply out the terms.

$$\begin{aligned} M(0) &= 500 & & \\ M(1) &= 500 \cdot 1.10 &&= 500(1.10)^1 \\ M(2) &= 500 \cdot 1.10 \cdot 1.10 &&= 500(1.10)^2 \\ M(3) &= 500 \cdot 1.10 \cdot 1.10 \cdot 1.10 &&= 500(1.10)^3 \\ M(4) &= 500 \cdot 1.10 \cdot 1.10 \cdot 1.10 \cdot 1.10 &&= 500(1.10)^4. \end{aligned}$$

The evident pattern in these calculations suggests that

$$M(n) = 500(1.10)^n \tag{3.2.2}$$

is a closed-form solution to the recurrence relation. ◇

Please note: *this only a guess.* We haven't *proved* that Equation 3.2.2 represents the same function as Equation 3.2.1. There is still more work to do. Stay tuned.

3.2.2 Polynomial Sequences: Using Differences ‡

Finding a closed-form solution for $P(n)$ can be thought of as guessing a formula for the sequence $P(0), P(1), P(2), \ldots$. If we knew that

$$P(n) = \text{a polynomial function of } n$$

then it would be easier to guess at a formula. One way to detect such a sequence is to look at the differences between terms. Given any sequence

$$a_0, a_1, a_2, a_3, \ldots, a_{n-1}, a_n$$

the differences

$$a_1 - a_0, a_2 - a_1, a_3 - a_2, \ldots, a_n - a_{n-1}$$

form another sequence, called the *sequence of differences*. A linear sequence will have a constant sequence of differences (because a line has constant slope). From calculus, we know that a quadratic sequence will have a linear sequence of differences, a cubic sequence will have a quadratic sequence of differences, etc. Given a sequence, we can calculate its sequence of differences, then calculate the sequence of differences of that, and so on. If we eventually end up with a constant sequence, then we have reason to believe that the original sequence is given by a polynomial function. The degree of the conjectured polynomial is the number of times we had to calculate the sequence of differences. The next example illustrates this technique.

Example 3.9 Find a closed-form solution for the recurrence relation from Example 3.4:

$$H(n) = \begin{cases} 1 & \text{if } n = 1 \\ H(n-1) + 6n - 6 & \text{if } n > 1. \end{cases} \tag{3.2.3}$$

Solution: Calculate the first few terms, and then look at sequences of differences:

$$\begin{array}{ccccccccccc}
1 & & 7 & & 19 & & 37 & & 61 & & 91 \\
& \bigvee & & \bigvee & & \bigvee & & \bigvee & & \bigvee & \\
& 6 & & 12 & & 18 & & 24 & & 30 & \\
& & \bigvee & & \bigvee & & \bigvee & & \bigvee & & \\
& & 6 & & 6 & & 6 & & 6 & &
\end{array}$$

The second sequence of differences is constant. This suggests that the sequence may have a formula of the form

$$H(n) = An^2 + Bn + C,$$

so all we need to do is find A, B, and C. Using $H(1) = 1$, $H(2) = 7$, and $H(3) = 19$, we get a system of three equations in three variables:

$$\begin{aligned} 1 &= A + B + C \\ 7 &= 4A + 2B + C \\ 19 &= 9A + 3B + C. \end{aligned}$$

Solving this system is a simple—but somewhat lengthy—exercise in algebra. (Add/subtract equations to eliminate variables, etc. We'll omit the details.) The solution is $A = 3$, $B = -3$, and $C = 1$, so

$$H(n) = 3n^2 - 3n + 1 \tag{3.2.4}$$

is a good candidate for a closed-form solution. ♢

The technique of using sequences of differences is more of a "brute force" approach than pure guessing; there are certain mechanical procedures you go through in order to arrive at a formula. But the result of these procedures is still only a guess. To be sure that our guess is right, we need to *prove* that the formula matches the recurrence relation for all n.

3.2.3 Inductively Verifying a Solution

If we use Equation 3.2.4 to calculate the first six values of $H(n)$, we get 1, 7, 19, 37, 61, and 91. These numbers match the values given by the recurrence relation in Equation 3.2.3 perfectly. This is pretty good evidence that Equation 3.2.4 is the correct closed-form solution for the recurrence relation. But this is not a proof. For all we know, the 7th value won't match, or the 8th, or the 739th. Without a proof, we can't be sure.

The general template for verifying a solution to a recurrence relation follows. We have

$$\begin{aligned} R(n) &= \text{some recurrence relation} \\ f(n) &= \text{hypothesized closed-form formula,} \end{aligned}$$

and we would like to show that $R(n) = f(n)$ for all values of n. For the purposes of this discussion, let's say that the first value of n for which $R(n)$ is defined is $n = 1$. As we have seen, recurrence relations usually start at $n = 1$ or $n = 0$, but any starting value of n is possible.

To prove that $R(n) = f(n)$ for all $n \geq 1$, we need to use the technique of *mathematical induction*. The idea of this kind of proof is analogous to climbing a staircase. Going up a staircase is a fairly repetitive task; if you know how to ascend one stairstep, you know how to ascend a staircase of any height. Of course, you need to start at the bottom of the staircase. In the proof, this is the base case.

Base Case: Verify that $R(1) = f(1)$.

Checking the base case is usually fairly easy. After all, you probably wouldn't have chosen $f(n)$ as a candidate solution if it didn't at least match for the case $n = 1$.

Next, you must be able to take one step up the staircase. Note that it doesn't matter where on the staircase you are; taking one step up requires the same amount of skill, whether you are at the bottom, top, or somewhere in the middle. So let's suppose, for the sake of the argument, you are standing on the $(k-1)$th stair. This is the inductive hypothesis.

Inductive Hypothesis: Let $k > 1$ be some (unspecified) integer. Suppose as *inductive hypothesis* that $R(k-1) = f(k-1)$.

The inductive hypothesis will be a crucial part of our proof. Any valid proof by induction must use the inductive hypothesis somewhere in its argument.

Finally, you need to be able to go up to the next stairstep. See Figure 3.5.

Inductive Step: Prove that $R(k) = f(k)$.

To do this, use the recurrence relation to compute the kth value. When you need to plug in the $(k-1)$th value, use the inductive hypothesis. Then use algebra to show that the answer matches the closed form solution.

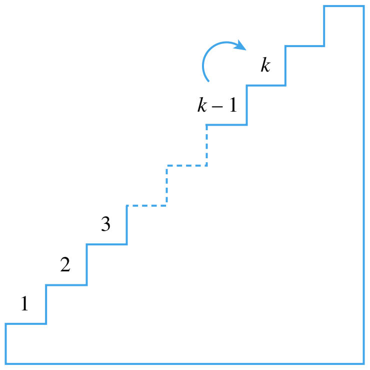


Figure 3.5 The inductive step shows that you can climb up one stairstep.

Why does this work? An induction argument proves the following assertion:

Let $k > 1$. If the recurrence relation matches the closed-form solution for $n = k - 1$, then the recurrence relation matches the closed-form solution for $n = k$.

In other words,

$$R(k-1) = f(k-1) \;\Rightarrow\; R(k) = f(k).$$

Since our proof supposed an arbitrary value of k, we are allowed to apply this assertion for *any* particular value of $k > 1$. Using $k = 2$, we get the assertion

> If the recurrence relation matches the closed-form solution for $n = 1$, then the recurrence relation matches the closed-form solution for $n = 2$.

But we know that the recurrence relation matches the closed-form solution for $n = 1$; this was the base case. Therefore the above assertion (and *modus ponens*) tells us that they match for $n = 2$. Now apply the assertion again, with $k = 3$:

> If the recurrence relation matches the closed-form solution for $n = 2$, then the recurrence relation matches the closed-form solution for $n = 3$.

So we know they match for $n = 3$. By repeating this argument, we can verify the cases $n = 4, 5, 6, \ldots$, up to whatever number we wish. In other words, the recurrence relation matches the closed-form solution for *any* value of n. This is exactly what we needed to show. In the symbols of logic, an induction argument establishes the following chain of implications:

$$\begin{aligned} R(1) = f(1) &\Rightarrow R(2) = f(2) \\ &\Rightarrow R(3) = f(3) \\ &\Rightarrow R(4) = f(4) \\ &\Rightarrow R(5) = f(5) \\ &\Rightarrow \cdots . \end{aligned}$$

The base case gets this chain of implications started, and since the inductive step works for any value of k, we are allowed to continue the chain of implications indefinitely to conclude that $R(n) = f(n)$ for any $n \geq 1$. The following example shows what a typical proof by induction should look like.

Example 3.10 Let $H(n)$ be defined by the following recurrence relation:

$$H(n) = \begin{cases} 1 & \text{if } n = 1 \\ H(n-1) + 6n - 6 & \text{if } n > 1. \end{cases}$$

Let $f(n) = 3n^2 - 3n + 1$. Prove that $H(n) = f(n)$ for all $n \geq 1$.

Proof We use induction on n.

> Base Case: If $n = 1$, the recurrence relation says that $H(1) = 1$, and the formula says that $f(1) = 3 \cdot 1^2 - 3 \cdot 1 + 1$, which is 1, so they match.

Inductive Hypothesis: Suppose as inductive hypothesis that

$$H(k-1) = 3(k-1)^2 - 3(k-1) + 1$$

for some $k > 1$.

Inductive Step: Using the recurrence relation,

$$\begin{aligned} H(k) &= H(k-1) + 6k - 6, \text{ by the second part of the recurrence relation} \\ &= 3(k-1)^2 - 3(k-1) + 1 + 6k - 6, \text{ by inductive hypothesis} \\ &= 3k^2 - 6k + 3 - 3k + 3 + 1 + 6k - 6 \\ &= 3k^2 - 3k + 1. \end{aligned}$$

So, by induction, $H(n) = f(n)$ for all $n \geq 1$. □

The solution of Example 3.10 makes a good proof template for verifying a closed-form solution to a simple recurrence relation; such a proof almost always should look like this. The next example establishes a closed-form solution to the recurrence relation of Example 3.5 for the number of elements in a power set. The proof is very similar to Example 3.10, but the induction starts at $n = 0$ instead of $n = 1$.

Example 3.11 Let $C(n)$ be defined by the following recurrence relation:

$$C(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2 \cdot C(n-1) & \text{if } n > 0. \end{cases}$$

Prove that $C(n) = 2^n$ for all $n \geq 0$.

Proof We use induction on n. Let $f(n) = 2^n$.

Base Case: If $n = 0$, the recurrence relation says that $C(0) = 1$, and the formula says that $f(0) = 2^0 = 1$, so $C(0) = f(0)$.

Inductive Hypothesis: Let $k > 0$. Suppose as inductive hypothesis that

$$C(k-1) = 2^{k-1}.$$

Inductive Step: Using the recurrence relation,

$$\begin{aligned} C(k) &= 2 \cdot C(k-1), \text{ by the second part of the recurrence relation} \\ &= 2 \cdot 2^{k-1}, \text{ by inductive hypothesis} \\ &= 2^k. \end{aligned}$$

So, by induction, $C(n) = 2^n$ for all $n \geq 0$. □

Take a moment to compare the proofs in Examples 3.10 and 3.11. In the exercises at the end of this section, you should mimic these examples when you are asked to prove that a given recurrence relation has a given closed-form solution. It is important to master this standard type of proof.

Mathematical induction is a difficult topic. In this section we have taken a very narrow view of the subject: verifying a closed-form solution. There are many other uses of mathematical induction, and these proofs can be quite challenging to write. The methods in this section will provide a foundation for more complicated inductive proofs.

We end this section with an example that shows why proofs are important.

Example 3.12 Let $P(n)$ be defined by the following recurrence relation:

$$P(n) = \begin{cases} 1 & \text{if } n = 0 \\ 3 \cdot P(n-1) - (n-1)^2 & \text{if } n > 0. \end{cases}$$

Does $P(n) = (n+2) \cdot 2^{n-1}$ for all $n \geq 0$?

Solution: In order to show that $(\forall n \geq 0)(P(n) = (n+2) \cdot 2^{n-1})$ is false, we need to show that its negation, $(\exists n \geq 0)(P(n) \neq (n+2) \cdot 2^{n-1})$, is true. That is, we need to find a value of n for which the closed-form formula does not match the recurrence relation. Using the recurrence relation,

$$\begin{aligned} a_0 &= 1 \\ a_1 &= 3 \cdot 1 - (1-1)^2 = 3 \\ a_2 &= 3 \cdot 3 - (2-1)^2 = 8 \\ a_3 &= 3 \cdot 8 - (3-1)^2 = 20 \end{aligned}$$

and using the closed-form formula,

$$\begin{aligned} (0+2) \cdot 2^{0-1} &= 1 \\ (1+2) \cdot 2^{1-1} &= 3 \\ (2+2) \cdot 2^{2-1} &= 8 \\ (3+2) \cdot 2^{3-1} &= 20, \end{aligned}$$

then the recurrence relation matches the closed-form solution for the first four values of n. Remember that we have always said that, while this may be good evidence that these two ways of calculating $P(n)$ will always agree, it is not a proof. Indeed, if we persist in our calculations, we find that

$$a_4 = 3 \cdot 20 - (4-1)^2 = 51$$

while

$$(4+2) \cdot 2^{4-1} = 48,$$

so the results do not match for $n = 4$. Therefore it is not the case that $P(n) = (n+2)\cdot 2^{n-1}$ for all $n \geq 0$. ♢

Once you have found a counterexample to a statement, you know that the statement is false in general. But failure to find a counterexample to a statement does not mean that the statement is true. In the previous example, we had to calculate five values of each formula to find a mismatch, but it could have taken 50, or 500, or any number of values to find our counterexample. This is why you will never know for sure that a closed-form formula matches a recurrence relation unless you prove it.

Exercises 3.2

1. Prove that the closed form solution in Equation 3.2.2 matches the recurrence relation in Equation 3.2.1. (See Example 3.8.)

2. Prove that the closed form solution in Equation 3.1.1 matches the recurrence relation in Equation 3.1.2. (See page 142.)

3. Consider the following recurrence relation:

$$B(n) = \begin{cases} 2 & \text{if } n = 1 \\ 3 \cdot B(n-1) + 2 & \text{if } n > 1. \end{cases}$$

   Use induction to prove that $B(n) = 3^n - 1$.

4. Consider the following recurrence relation:

$$P(n) = \begin{cases} 0 & \text{if } n = 0 \\ 5 \cdot P(n-1) + 1 & \text{if } n > 1. \end{cases}$$

   Prove by induction that $P(n) = \dfrac{5^n - 1}{4}$ for all $n \geq 0$.

5. Consider the following recurrence relation:

$$C(n) = \begin{cases} 0 & \text{if } n = 0 \\ n + 3 \cdot C(n-1) & \text{if } n > 0. \end{cases}$$

   Prove by induction that $C(n) = \dfrac{3^{n+1} - 2n - 3}{4}$ for all $n \geq 0$.

6. Guess a closed-form solution for the following recurrence relation:

$$K(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2 \cdot K(n-1) - n + 1 & \text{if } n > 0. \end{cases}$$

Prove that your guess is correct.

7. Guess a closed-form solution for the following recurrence relation:

$$P(n) = \begin{cases} 5 & \text{if } n = 0 \\ P(n-1) + 3 & \text{if } n > 0. \end{cases}$$

Prove that your guess is correct.

8. Consider the following recurrence relation:

$$P(n) = \begin{cases} 1 & \text{if } n = 0 \\ P(n-1) + n^2 & \text{if } n > 0. \end{cases}$$

   (a) Compute the first eight values of $P(n)$.
   (b) Analyze the sequences of differences. What does this suggest about the closed-form solution?
   (c) Find a good candidate for a closed-form solution.
   (d) Prove that your candidate solution is the correct closed-form solution.

9. Consider the following recurrence relation:

$$G(n) = \begin{cases} 1 & \text{if } n = 0 \\ G(n-1) + 2n - 1 & \text{if } n > 0. \end{cases}$$

   (a) Calculate $G(0)$, $G(1)$, $G(2)$, $G(3)$, $G(4)$, and $G(5)$.
   (b) Use sequences of differences to guess at a closed-form solution for $G(n)$.
   (c) Prove that your guess is correct.

10. Find a polynomial function $f(n)$ such that $f(1), f(2), \ldots, f(8)$ is the following sequence:

$$2, 7, 12, 17, 22, 27, 32, 37.$$

11. Find a polynomial function $f(n)$ such that $f(1), f(2), \ldots, f(8)$ is the following sequence:

$$1, 1, 2, 4, 7, 11, 16, 22.$$

12. Analyze the sequence

$$1, 6, 15, 100, 501, 1746, 4771, 11040, 22665, 42526, 74391$$

using sequences of differences. From what degree polynomial does this sequence appear to be drawn? (Don't bother finding the coefficients of the polynomial.)

13. Refer to Example 3.6. Guess at a closed-form solution to the recurrence relation for the number of nodes in a complete binary tree of height n. Prove that your guess is correct.

14. Refer to Example 3.7. Guess at a closed-form solution to the recurrence relation for the number of edges in K_n, the complete graph on n vertices. Prove that your guess is correct.

15. Guess a closed-form solution for the following recurrence relation:

$$P(n) = \begin{cases} 1 & \text{if } n = 0 \\ P(n-1) + 2^n & \text{if } n > 0. \end{cases}$$

(Hint: Consider powers of 2.) Prove that your guess is correct.

16. Recall that $n! = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$ for $n > 0$, and by definition, $0! = 1$. Prove that $F(n) = n!$ for all $n \geq 0$, where

$$F(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot F(n-1) & \text{if } n > 0. \end{cases}$$

17. Consider the following recurrence relation:

$$H(n) = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot H(n-1) + 1 & \text{if } n > 0. \end{cases}$$

Prove that $H(n) = n!(1/1! + 1/2! + 1/3! + \cdots + 1/n!)$ for all $n \geq 1$.

18. Is $1 + \frac{17}{6}n - 2n^2 + \frac{7}{6}n^3$ a closed-form solution for the following recurrence relation?

$$P(n) = \begin{cases} 1 & \text{if } n = 0 \\ 4 \cdot P(n-1) - n^2 & \text{if } n > 0. \end{cases}$$

Prove or disprove.

19. Recall the Fibonacci numbers defined in Definition 3.1 on page 144. Recall also that $\lceil x \rceil$ is the least integer k such that $k \geq x$, called the *ceiling* of x. Is it true that

$$F(n) = \left\lceil e^{\left(\frac{n-2}{2}\right)} \right\rceil$$

for all $n \geq 1$? Prove or disprove.

*20. Recall the definition of Fibonacci numbers in Definition 3.1.

(a) Compute the sequence of differences of the first nine Fibonacci numbers. What seems to be true about this sequence?

(b) Prove your assertion in part (a).

(c) Explain why a closed-form formula for the Fibonacci numbers cannot be a polynomial function.

*21. Suppose you are given a sequence of numbers $a_1, a_2, a_3, \ldots, a_k$. Explain how to construct a polynomial $p(x)$ such that $p(n) = a_n$ for all $n = 1, 2, 3, \ldots, k$. (Note that this fact, along with Exercise 20, shows that it is possible for a closed-form formula to match a recurrence relation for arbitrarily many terms, without being a valid closed-form solution.)

3.3 Recursive Definitions

The definition of a recurrence relation is *self-referential*—we state a rule for calculating the values of a function in terms of itself. Recurrence relations have two parts: a base case that describes the simplest case of the function and a recursive step that describes the function in terms of a simpler version of itself. This is the essence of recursive thinking. In this section, we will apply this idea to an assortment of different objects.

3.3.1 Definition and Examples

We will use the term *object* somewhat loosely—an object could be a number, a mathematical structure, a function, or almost anything else we want to describe. A *recursive definition* of a given object has the following parts:

B. a base case, which usually defines the simplest possible such object, and

R. a recursive case, which defines a more complicated object in terms of a simpler one.

The best way to understand recursive definitions is to see some examples.

Example 3.13 Any recurrence relation is a recursive definition of a function. For example, the recurrence relation

$$H(n) = \begin{cases} 1 & \text{if } n = 1 \\ H(n-1) + 6n - 6 & \text{if } n > 1 \end{cases}$$

for the hexagonal numbers (Example 3.4) can be written as a recursive definition with a base case and a recursive case:

B. $H(1) = 1$.

R. For any $n > 1$, $H(n) = H(n-1) + 6n - 6$.

Example 3.14 Let X be a set of actors and actresses, defined as follows.

B. Kevin Bacon $\in X$.

R. Let x be an actor or actress. If, for some $y \in X$, there has been a movie in which both x and y appear, then $x \in X$.

In other words, Kevin Bacon is in X, and anyone who has been in some movie with someone in X is also in X. For example, in order to show that Arnold Schwarzenegger $\in X$, we note that Arnold Schwarzenegger appeared in *Conan the Barbarian* with James Earl Jones, who appeared in *Clear and Present Danger* with Harrison Ford, who appeared in *The Fugitive* with Tommy Lee Jones, who appeared in *JFK* with Kevin Bacon.

Although this example might seem silly, it illustrates an important way to think about how things are connected. It is not hard to see how the same definition could describe the set of all computers exposed to a certain virus, or the collection of people who have heard about a tornado warning, etc. For more on the Kevin Bacon example and its relation to graph theory, see Hopkins [15].

A sequence of symbols written together in some order is called a *string*. The next example gives a useful recursive definition. Note that there are two base cases in this example.

Example 3.15 Given a list of symbols $a_1, a_2, \ldots, a_m$, a *string* of these symbols is:

$\mathbf{B_1}$. the empty string, denoted by λ, or

$\mathbf{B_2}$. any symbol a_i, or

R. xy, the *concatenation* of x and y, where x and y are strings.

Since λ represents the empty string, we do not write λ after it has been concatenated with another string. For example, $\texttt{cubs}\lambda = \texttt{cubs}$. It shouldn't be hard to convince yourself that this definition describes any possible "word" in the given symbols.

Example 3.16 A special kind of string called a *palindrome* can be defined as follows.

$\mathbf{B_1}$. λ is a palindrome.

B_2. Any symbol a is a palindrome.

R. If x and y are palindromes, then yxy is a palindrome.

Note that any word that is the same forward as backward is a palindrome, such as `racecar` or `HANNAH`. We can build up the palindrome `racecar` from the definition as follows.

1. Since it is a symbol, `e` is a palindrome, by **B_2**, the second part of the definition.
2. Similarly, `c` is a palindrome.
3. Using **R**, `cec` is a palindrome.
4. By **B_2**, `a` is a palindrome.
5. By **R**, `aceca` is a palindrome.
6. By **B_2**, `r` is a palindrome.
7. By **R**, `racecar` is a palindrome.

As an exercise, think about why we have to define λ as a palindrome.

The next example has one base case and two recursive cases.

Example 3.17 The set X of all binary strings (strings with only 0's and 1's) having the same number of 0's and 1's is defined as follows.

B. λ is in X.

R_1. If x is in X, so are $1x0$ and $0x1$.

R_2. If x and y are in X, so is xy.

Notice that both recursive cases preserve the property of having the same number of 1's as 0's. Both of these cases form new strings from old by adding 0's and 1's, and they always add the same amount of each.

Strings can be useful in a number of contexts: text in word processing, genetic sequences in bioinformatics, etc. Thinking recursively can help us define operations that manipulate the symbols in a string. For example, the next recursive definition describes how to reverse the order of a string.

Example 3.18 If s is a string, define its *reverse* s^R as follows.

B. $\lambda^R = \lambda$.

R. If s has one or more symbols, write $s = ra$ where a is a symbol and r is a string (possibly empty). Then $s^R = (ra)^R = ar^R$.

In other words, you reverse a string by moving the last symbol to the front and reversing the rest of the string. To see how this definition works, consider the string $\texttt{pit}$. Its reverse, $(\texttt{pit})^R$, is calculated as follows:

$$\begin{aligned}
(\texttt{pit})^R &= \texttt{t}(\texttt{pi})^R \quad \text{by part } \mathbf{R}\\
&= \texttt{ti}(\texttt{p})^R \quad \text{by part } \mathbf{R}\\
&= \texttt{ti}(\lambda\texttt{p})^R \quad \text{(insertion of empty string)}\\
&= \texttt{tip}\lambda^R \quad \text{by part } \mathbf{R}\\
&= \texttt{tip}\lambda \quad \text{by part } \mathbf{B}\\
&= \texttt{tip} \quad \text{(removal of empty string).}
\end{aligned}$$

Most of the work in reversing $\texttt{pit}$ was devoted to showing that $(\texttt{p})^R = \texttt{p}$. The definition doesn't say that the reversal of a one-symbol string is the same one-symbol string, but this fact follows from the definition by the above argument. It is a good idea to state such facts as theorems.

Theorem 3.1 *If a is a symbol, then $a^R = a$.*

Proof As above, $a^R = (\lambda a)^R = a\lambda^R = a\lambda = a$. □

Example 3.19 Define a *line map* as follows.

B. A blank rectangle is a line map.

R. A line map with a straight line drawn all the way across it is a new line map.

The recursive definitions of strings and line maps are similar: the base case is a blank object, and the recursive case defines how to add a new piece to an object to make it more complex. This is a useful way to think about objects that are built from identifiable pieces.

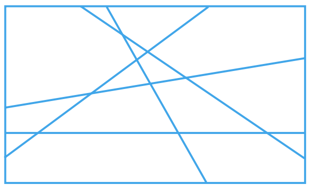


Figure 3.6 A line map.

3.3.2 Writing Recursive Definitions

Just as some functions are easy to define using recurrence relations, some objects have natural recursive definitions. The trick to writing a recursive definition is to see the desired object as being built out of levels. The recursive case of the definition must describe a level in terms of the next simplest level. The base case should describe the simplest possible such object. Some examples will make this idea less abstract.

Example 3.20 Suppose you start browsing the Internet at some specified page p. Let X be the set of all pages you can reach by following links, starting at p. Give a recursive definition for the set X.

Solution: Observe that if you can reach some page x, then you can reach any page to which x has a link. This gives the recursive part of the definition:

R. If $x \in X$ and y is some page such that x links to y, then $y \in X$.

The base case is the page where you start:

B. $p \in X$.

Notice the similarity to Example 3.14. ◊

The reasoning in this last example was "top-down." We thought about the recursive part first: what pages can you get to from any given page? The next example uses "bottom-up" thinking: start with the simplest case, and think about how to build up a slightly more complicated case.

Example 3.21 Give a recursive definition for the set of all odd natural numbers.

Solution: To find the base case, think about the simplest possible case of an odd natural number. A reasonable choice for the simplest odd number is 1. For the recursive case, think about how to get a new odd number from an old odd number. Observe that, if x is odd, then $x + 2$ is odd also. So we can define the set X of odd numbers as follows.

B. 1 is in X.

R. If x is in X, so is $x + 2$. ◊

At this point you should object: we already have a definition for odd numbers (Definition 1.6). According to this definition, the set of odd natural numbers should be

$$\{n \in \mathbf{N} \mid n = 2k + 1 \text{ for some integer } k\}.$$

Fair enough. Definition 1.6 does stipulate what odd numbers are, once and for all. We should therefore view the recursive definition in Example 3.21 as an equivalent way to describe the set of odd natural numbers. Of course, we need to prove that these two definitions are equivalent; we'll do this in the next section.

Example 2.5 illustrated how to organize data into a binary search tree. More generally, any graph with this type of structure is called a *binary tree.* These graphs have a natural recursive definition.

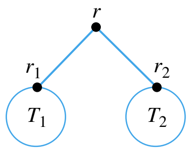
Example 3.22 Give a recursive definition for the set of all binary trees.

Solution: For convenience, let's allow the "empty tree" consisting of no vertices and no edges to be a simple case of a binary tree. A single vertex could also be considered to be a binary tree. Using these two building blocks, a new tree can be formed from two existing trees by joining the two trees under a common root node. So we can write the following definition for a binary tree. Note that this definition also defines the root of the binary tree.

B_1. The empty tree is a binary tree.

B_2. A single vertex is a binary tree. In this case, the vertex is the root of the tree.

R. If T_1 and T_2 are binary trees with roots r_1 and r_2, respectively, then the tree



is a binary tree with root r. Here the circles represent the binary trees T_1 and T_2. If either of these trees T_i $(i = 1, 2)$ is the empty tree, then there is no edge from r to T_i. ◇

As we saw in Section 3.1, seeing the recursive structure to an object can be the key to defining a recurrence relation. In Example 3.6, we observed that a complete binary tree is made of two smaller complete binary trees. The definition in Example 3.22 generalizes this observation.

3.3.3 Recursive Geometry

We can think of a recurrence relation $R(n)$ as a rule for constructing a sequence of numbers $R(1), R(2), R(3), \ldots$. For example, the recurrence relation

$$R(n) = \begin{cases} 2 & \text{if } n = 1 \\ \sqrt{R(n-1)} & \text{if } n > 1 \end{cases}$$

produces the sequence

$$2, 1.414, 1.189, 1.091, 1.044, 1.022, 1.011, 1.005, \ldots$$

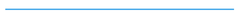
to three decimal places. We say that the *limit* of this sequence is 1 because the numbers in this sequence get closer and closer to 1 as n gets larger.

Similarly, we can consider the limit of a recursive definition of geometric patterns. This is one way to construct *fractals*, a special type of shape with infinite layers of self-similarity. The following examples illustrate this process.

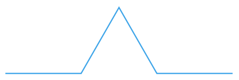
Example 3.23 Define a sequence of shapes as follows.

B. $K(1)$ is an equilateral triangle.

R. For $n > 1$, $K(n)$ is formed by replacing each line segment



of $K(n-1)$ with the shape



such that the central vertex points outwards.

Figure 3.7 shows the first three terms of this sequence.

The limit of this sequence of shapes is a fractal known as the Koch snowflake, shown in Figure 3.8.

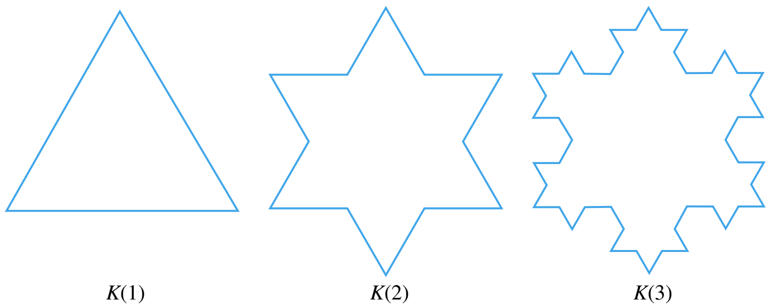


Figure 3.7 The curves $K(1)$, $K(2)$, and $K(3)$.

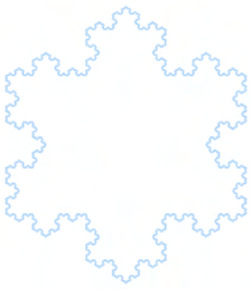


Figure 3.8 The Koch snowflake fractal.

The next example illustrates how to find a recursive structure, given a fractal.

Example 3.24 The fractal model for the Badda-Bing axiomatic system of Example 1.17 is shown in Figure 3.9. Notice that the pattern starts with a

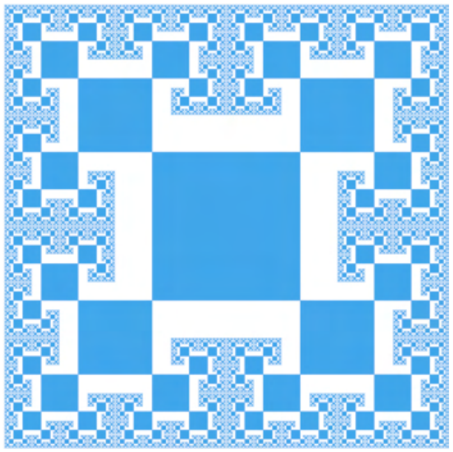


Figure 3.9 A fractal model for the Badda-Bing geometry.

central square in the middle, with smaller squares on each vertex of this square, and smaller squares on all those vertices, and so on. Each time the squares get smaller, they do so by a factor of 1/2. These observations lead to a definition for a sequence of shapes $B(1), B(2), B(3), \ldots$ whose limit is the Badda-Bing fractal.

B. $B(1)$ is a square.

R. For $n > 1$, $B(n)$ is formed from $B(n-1)$ by adding squares to every vertex v that lies on only one square S. Such a new square must have v

as a vertex, be oriented the same way as S, touch S at v only, and have side length 1/2 the side length of S.

Figure 3.10 shows the first three terms of this sequence.

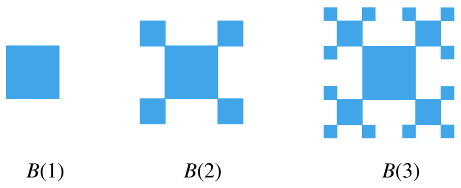
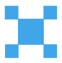
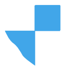
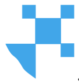


Figure 3.10 The first three terms of a sequence whose limit is the Badda-Bing fractal.

The above definition defines a sequence of shapes $B(1), B(2), B(3), \ldots$ that approximate the Badda-Bing fractal. None of these shapes *is* the fractal, exactly. The actual fractal consists of infinitely many squares, while each approximating shape $B(n)$ contains finitely many. However, it is possible to define the fractal as a recursive set, following the idea in Example 3.21. The following statements define a set B of points in the plane.

B.  is in S.

R. If  is in S (in any orientation), so are .

This definition is somewhat informal, but it helps us think about the recursive nature of the fractal. If you were to draw the fractal using this definition, you would start with the five squares in the base case, then you would apply the recursive case to the four squares on the corners, then you would apply the recursive case to the twelve little squares on the corners, and so on. This hypothetical "bottom-up" construction continues forever, with the squares becoming arbitrarily small and numerous, accounting for the fractal nature of the set B.

Of course, we haven't proved that the limit of the sequence $B(1), B(2), B(3), \ldots$ is the set B; we aren't going to study limits in enough detail to make such claims. The purpose of this example is get us thinking recursively.

The mathematical study of fractals is relatively new; most of the important discoveries in this field weren't made until the second half of the 20th century, when computers became widely available to mathematicians. While these objects can be fascinating in themselves, their self-similar structure can help us visualize the concept of recursion. In a fractal, you can see copies of the fractal one layer down; each arm of the snowflake in Figure 3.8 has smaller, identically shaped arms coming off of it. This is the sort of observation you must make when you write recursive definitions. How does a string contain a smaller string on the inside? How can we build up a greater odd number from a lesser one? Fractals are pretty, but they are important because they show us a picture of recursive thinking.

3.3.4 Recursive Jokes

One measure of how well you understand a certain concept is whether you understand humor based on the concept. Here are some recursive jokes. Maybe these aren't very funny, but hopefully you don't need them explained to you.

In order to understand recursion, you must understand recursion.

It isn't unusual for the following to be in the index of a book:

Recursion, *see Recursion.*

Some acronyms are recursive jokes: VISA (VISA International Service Association), GNU (GNU's Not Unix), and PHP (PHP Hypertext Protocol). An early spin-off of the text editor EMACS was named EINE (EINE Is Not EMACS), and one of its successors was called ZWEI (ZWEI Was EINE Initially).

Certain well-known camp songs have recursive definitions, for example, "99 bottles of beer on the wall ...." This song is a good example of top-down thinking.

Exercises 3.3

1. Write the recurrence relation for the Fibonacci numbers (Definition 3.1) in the form of a recursive definition, with two base cases and one recursive case.

2. See Example 3.16. Why is the first part of the definition necessary? (In other words, why must λ be defined as a palindrome?)

3. Give a recursive definition for the set X of all binary strings with an even number of 0's.

4. See Example 3.17. Give a recursive definition for the set Y of all binary strings with *more* 0's than 1's.

5. Use the definition of the reverse of a string in Example 3.18 to compute $(\texttt{cubs})^R$. Justify each step using the definition.

6. Refer to Example 3.15. Suppose that the symbols can be compared, so for any i and j with $i \neq j$, either $a_i < a_j$ or $a_j < a_i$. Modify the definition so that it defines the set of all strings whose symbols are in increasing order.

7. Let K be the set of all cities that you can get to from Toronto by taking flights (or sequences of flights) on commercial airlines. Give a recursive definition of K.

8. Create your own example of an object or situation whose recursive definition is the same as the Kevin Bacon movie club in Example 3.14.

9. Define a set X of numbers as follows.

   B. $2 \in X$.

   R₁. If $x \in X$, so is $10x$.

   R₂. If $x \in X$, so is $x + 4$.

   (a) List all the elements of X that are less than 30.

   (b) Explain why there are no odd numbers in X.

10. Give a recursive definition for the set of even integers (including both positive and negative even integers).

11. Give a recursive definition for the set of all powers of 2.

12. Define a set X recursively as follows.

   B. 3 and 7 are in X.

   R. If x and y are in X, so is $x + y$. (Here it is possible that $x = y$.)

   Decide which of the following numbers are in X. Explain each decision.

   (a) 24

   (b) 1,000,000

   (c) 11

13. Define a set X recursively as follows.

   B. $12 \in X$.

   R₁. If $x \in X$ and x is even, then $x/2 \in X$.

   R₂. If $x \in X$ and x is odd, then $x + 1 \in X$.

   List all the elements of X.

14. Give a recursive definition for the set X of all natural numbers that are one or two more than a multiple of 10. In other words, give a recursive definition for the set $\{1, 2, 11, 12, 21, 22, 31, 32, \dots\}$.

15. In Example 3.22, we gave a recursive definition of a binary tree. Suppose we modify this definition by deleting part $\mathbf{B_1}$, so that an empty tree is not a binary tree. A tree satisfying this revised definition is called a *full binary tree.*

    (a) Give an example of a full binary tree with five nodes.

    (b) Give an example of a binary tree with five nodes that is not a full binary tree.

16. Let G be an undirected graph, possibly not connected. The different pieces that make up G are called the *connected components* of G. More precisely, for any vertex v in G, the connected component G_v containing v is the graph whose vertices and edges are those that lie on some path starting at v. Give a recursive definition for G_v. (Hint: Mimic Example 3.14.)

17. Refer to Example 3.23. Suppose that the perimeter of $K(1)$ is 3. Give a recurrence relation for the perimeter $P(n)$ of the nth shape $K(n)$ in the sequence. Guess at a closed-form solution to $P(n)$. What does this say about the perimeter of the Koch snowflake fractal?

18. The Sierpinski gasket fractal is shown in Figure 3.11. In the first part of Example 3.24, we saw how to define a sequence of shapes to approximate a given fractal. Using this example as a guide, give a recursive definition for $S(n)$, where the limit of the sequence $S(1), S(2), S(3), \ldots$ is the Sierpinski gasket.

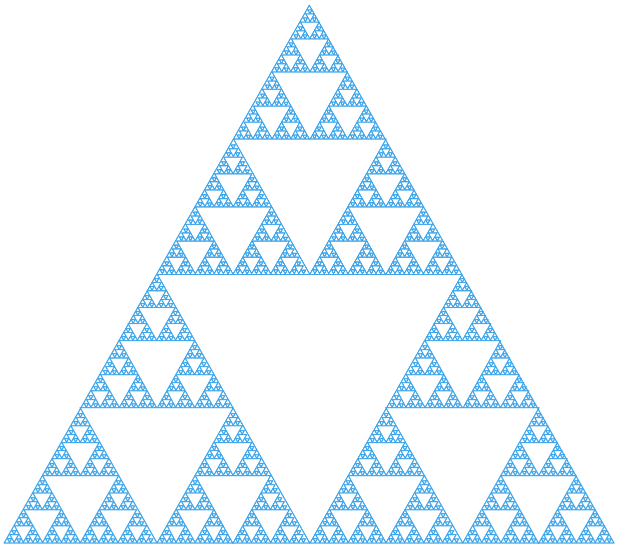


Figure 3.11 The Sierpinski gasket.

19. In the second part of Example 3.24, we saw an informal way to define a fractal as a recursively defined set. Using this example as a guide, give an informal definition of the Sierpinski gasket as a recursively defined set.

20. Give a recursive definition for $T(n)$, where the sequence $T(1), T(2), T(3), \ldots$ is the fractal tree shown in Figure 3.12. Figure 3.13 shows the first three terms of this sequence.

21. Given an informal definition of the fractal tree in Figure 3.12 as a recursively defined set.

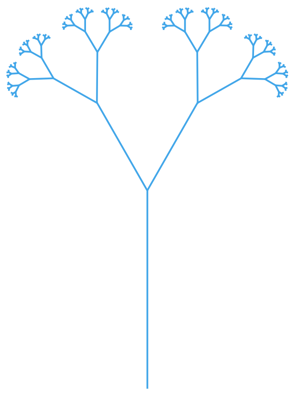


Figure 3.12 A fractal tree. The "buds" at the top are actually tiny branches whose shape is similar to the larger branches lower in the tree. These branches become smaller and smaller (and more numerous) as you move up the tree.

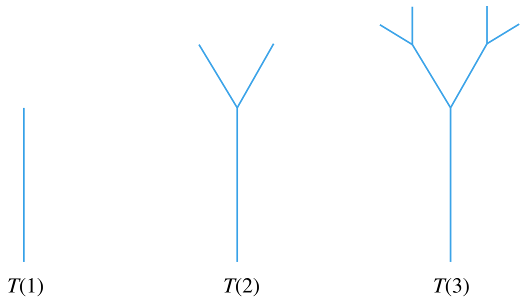


Figure 3.13 The first three terms of a sequence whose limit is the fractal in Figure 3.12.

22. Give a recursive definition for $C(n)$, where the sequence $C(1), C(2), C(3), \ldots$ is the fractal shown in Figure 3.14. (Hint: Figure 3.15 shows the first and

third terms of this sequence. In Figure 3.15, if the largest circle has radius 4, then the other circles have radii 2 and 1.)

23. Give an informal definition of the shape in Figure 3.14 as a recursively defined set.

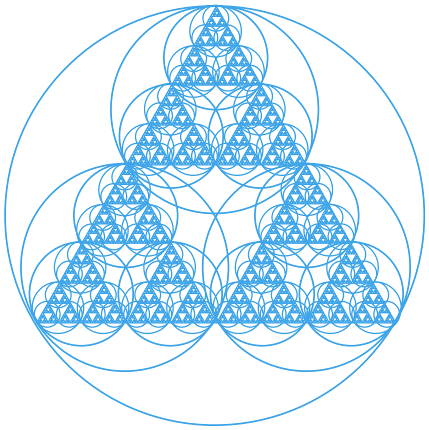


Figure 3.14 A fractal composed of circles.

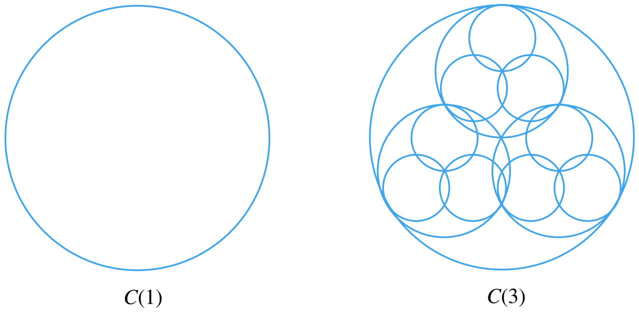


Figure 3.15 The first and third terms of a sequence whose limit is the fractal in Figure 3.14.

3.4 Proof by Induction

In Section 3.2, we used induction to verify a closed-form solution to a recurrence relation. There are lots of other statements in mathematics that lend themselves to proofs by induction. Here are some examples.

- The sum of the first n natural numbers is $\frac{n(n+1)}{2}$.
- A binary tree of height n has less than 2^{n+1} nodes.
- A convex n-gon has $\frac{n(n-3)}{2}$ diagonals.

What do these examples have in common? They are all statements containing the variable n, and in each case, n stands for some natural number—each statement takes the form "Statement(n)" for all n. Furthermore, these statements all involve objects that have some sort of recursive structure. In this section we extend the technique of induction to prove facts about recursively defined objects.

3.4.1 The Principle of Induction

To prove that a closed form solution $f(n)$ matches a recurrence relation $R(n)$, we had to prove the following:

For all $n \geq 1$, $R(n) = f(n)$.

We did this by first checking the base case $R(1) = f(1)$, and then proving that $R(k) = f(k)$ follows from the inductive hypothesis $R(k-1) = f(k-1)$, for any $k > 1$. The principle of mathematical induction generalizes this approach.

The Principle of Mathematical Induction. To prove the statement

"Statement(n), for all $n \geq 1$,"

it suffices to prove

1. Statement(1), and
2. Statement($k-1$) $\Rightarrow$ Statement(k), for $k > 1$.

This principle is plausible by the same reasoning used in Section 3.2: Step 2 establishes a chain of implications:

$$\text{Statement}(1) \Rightarrow \text{Statement}(2) \Rightarrow \text{Statement}(3) \Rightarrow \cdots,$$

while step 1 gets the chain of implications started. It follows that these two steps, taken together, establish Statement(n) for all $n \geq 1$.

We label this a "principle" because, strictly speaking, we can't prove it as a theorem. In advanced treatments of the foundations of mathematics, this statement is usually assumed as an axiom.[2] Recall that part 1 of the principle is called the *base case* and part 2 is called the *inductive step*. In proving the inductive step, the assumption that Statement($k-1$) is true is called the *inductive hypothesis*.

2. Often an equivalent condition called the *well-ordering principle* is assumed as an axiom: every nonempty set of positive integers contains a least element.

3.4.2 Examples

Mathematical induction is often the technique to use when you can think about a problem recursively. The discussion of the next result illustrates how to take a recursive viewpoint.

Theorem 3.2 *For any* $n \geq 1$, $1 + 2 + 3 + \cdots + n = \dfrac{n(n+1)}{2}$.

Let $S_n = 1 + 2 + 3 + \cdots + n$ be the sum of the first n natural numbers. Then S_n has a recursive definition:

B. $S_1 = 1$.

R. $S_n = S_{n-1} + n$ for $n > 1$.

Understanding this definition is the key to following the inductive argument. Before digesting the proof, notice how we use the notation of the Principle of Mathematical Induction.

$$\begin{array}{ll}
\text{Statement}(n): & 1 + 2 + 3 + \cdots + n = \dfrac{n(n+1)}{2}. \\
\text{Statement}(1): & 1 = \dfrac{1(1+1)}{2}. \\
\text{Statement}(k-1): & 1 + 2 + 3 + \cdots + (k-1) = \dfrac{(k-1)(k-1+1)}{2}. \\
\text{Statement}(k): & 1 + 2 + 3 + \cdots + k = \dfrac{k(k+1)}{2}.
\end{array}$$

Proof (Induction on n.)

Base Case: If $n = 1$, then the sum of the first n natural numbers is 1, and $n(n+1)/2 = 1 \cdot 2/3 = 1$, so Statement(1) is true.

Inductive Hypothesis: Suppose as inductive hypothesis that

$$1 + 2 + \cdots + (k-1) = \frac{(k-1)(k-1+1)}{2}$$

for some $k > 1$.

Inductive Step: Adding k to both sides of this equation gives

$$\begin{aligned}
1 + 2 + \cdots + (k-1) + k &= \frac{(k-1)(k-1+1)}{2} + k \\
&= \frac{(k-1)(k) + 2k}{2} \\
&= \frac{k^2 + k}{2} \\
&= \frac{k(k+1)}{2}
\end{aligned}$$

as required. □

The recursive definition of S_n guides this proof: to move up one level from S_{k-1} to S_k, the definition says you just add k. This is exactly what we did to move from Statement$(k-1)$ to Statement(k) in the inductive step of the proof.

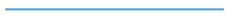
Theorem 3.2 is basically the statement that a certain recurrence relation has a certain closed-form solution, so the proof should remind you of those in Section 3.2. The next result looks different, but the underlying logic is the same, and we follow the same basic template.

Theorem 3.3 *Let $K(1), K(2), K(3), \ldots$ be the sequence of shapes whose limit is the Koch Snowflake of Example 3.23. Then $K(n)$, the nth term in this sequence, is composed of $4^{n-1} \cdot 3$ line segments.*

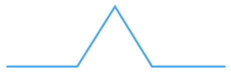
The recursive definition of this sequence is important:

B. $K(1)$ is an equilateral triangle.

R. For $n > 1$, $K(n)$ is formed by replacing each line segment



of $K(n-1)$ with the shape



such that the central vertex points outward.

The base case and the inductive steps of the following proof correspond to the base case and the recursive part of the definition, respectively.

Proof (Induction on n.)

Base Case: The base case of the definition states that $K(1)$ consists of three line segments, and $3 = 4^{1-1} \cdot 3$, so the theorem is true when $n = 1$.

Inductive Hypothesis: Suppose as inductive hypothesis that $K(k-1)$ is composed of $4^{k-1-1} \cdot 3 = 4^{k-2} \cdot 3$ line segments, for some $k > 1$.

Inductive Step: By the recursive part of the definition, $K(k)$ is formed by replacing each line segment with four others, so the number of line segments is multiplied by four. Therefore $K(k)$ is composed of $4 \cdot 4^{k-2} \cdot 3 = 4^{k-1} \cdot 3$ line segments, as required. □

The preceding proof closely resembles the proofs in Section 3.2 because Statement(n) was written in terms of a numerical formula, much like a closed-form solution. The next example is a little different because the claim is stated completely in the language of strings.

Theorem 3.4 *The string reversal function of Example 3.18 works. In other words, for any $n \geq 1$, $(a_1 a_2 \cdots a_{n-1} a_n)^R = a_n a_{n-1} \cdots a_2 a_1$.*

Proof (Induction on n.)

Base Case: If $n = 1$, then $a^R = a$ by Theorem 3.1, so the reversal function correctly reverses a one-element string.

Inductive Hypothesis: Suppose as inductive hypothesis that the reversal function works for any string of length $k-1$, for some $k > 1$.

Inductive Step: Given a string $a_1 a_2 a_3 \cdots a_k$ of length k,

$$\begin{aligned}(a_1 a_2 a_3 \cdots a_{k-1} a_k)^R &= a_k (a_1 a_2 a_3 \cdots a_{k-1})^R && \text{by definition of } {}^R\text{, part } \mathbf{R} \\ &= a_k (a_{k-1} \cdots a_3 a_2 a_1) && \text{by inductive hypothesis} \\ &= a_k a_{k-1} \cdots a_3 a_2 a_1\end{aligned}$$

as required. □

Take another look at this last proof. It follows the template of the verification of a closed-form solution to a recurrence relation, but the recurrence relation is replaced by the recursive definition of the string reversal function. Otherwise, all the components are the same: check the base case, state the inductive hypothesis, use the recursive definition, apply the inductive hypothesis, and simplify to show the desired result.

Proofs involving recursive definitions often require induction. The next example is quite different; it indicates the diversity of results that can be proved using this technique.

Mathematicians say that a map can be N-colored if there is a way to color all of the regions of the map using only N colors so that no two regions with a common border share the same color. Recall the definition of a line map given in Example 3.19. Notice that the line map in Figure 3.6 can be two-colored. Figure 3.16 shows a possible two-coloring. The next theorem says that this was no accident.

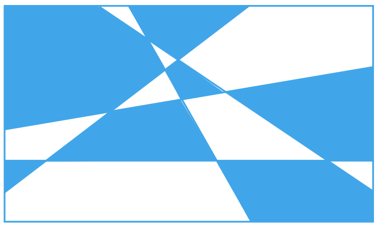


Figure 3.16 A two-coloring of a line map.

Theorem 3.5 *Any line map can be two-colored.*

Proof (Induction on the number of lines.)

> Base Case: If a line map contains 0 lines, then it is just a blank rectangle, so it can be two-colored trivially using a single color.
>
> Inductive Hypothesis: Suppose as inductive hypothesis that any line map with $k-1$ lines can be two-colored, for some $k > 0$.
>
> Inductive Step: Let M be a line map with k lines. Remove one line from M, call it l. By inductive hypothesis, the resulting map can be two-colored, so two-color it. Now put back l, and reverse the colors of all the regions on one side of the line l. Each side will still be correctly two-colored, and any regions having l as a border will be opposite colors. Hence the map M is correctly two-colored. □

In this proof, we started our induction at 0 instead of 1. We could have started at 1, and then the base case would read as follows.

> If a line map contains just 1 line, then one side of the line can be colored white and the other black, so the map can be two-colored.

The rest of the proof would be exactly the same. Sometimes there is a choice of where to start the base case of an inductive argument.

Figure 3.17 illustrates the reasoning in the inductive step of the proof of Theorem 3.5. To construct a line map, remove a line l, apply the inductive hypothesis, put l back, and reverse the colors.

We have been following the same basic template for proofs by induction since Example 3.10, where we first verified the closed-form solution of a recurrence relation. Since then we have seen several examples. Although these examples differ in a variety of ways, they all match a basic pattern, which we can now state with more generality. In the following template, [object] stands for some sort of recursively defined object with "levels" for each natural number, and [property P] stands for the property of the [object] we are trying to justify.

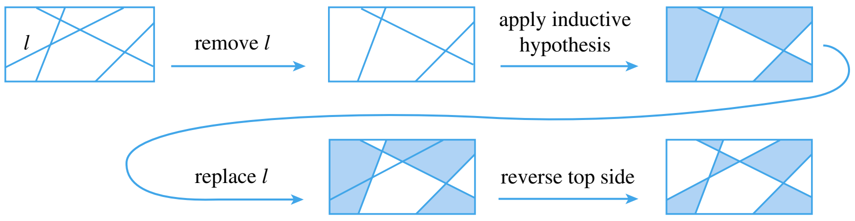


Figure 3.17 The reasoning of the inductive step in the proof of Theorem 3.5.

Template for Inductive Proofs. An inductive proof of the statement "All [object]s have [property P]" should have the following components.

> Base Case: Prove that [property P] holds for the simplest [object].
>
> Inductive Hypothesis: Suppose as inductive hypothesis that [property P] holds for an [object] of level $k-1$, for some k.
>
> Inductive Step: Suppose as given an [object] of level k. Use the inductive hypothesis and the recursive definition of [object] to conclude that the given level k [object] has [property P].

For example, this is how the template fits the proof of Theorem 3.5.

> Base Case: Prove that a line map with 0 lines can be two-colored.
>
> Inductive Hypothesis: Suppose as inductive hypothesis that a line map with $k-1$ lines can be two-colored, for some $k > 0$.
>
> Inductive Step: Suppose as given a line map with k lines. Follow the reasoning in Figure 3.17.

If you are having trouble getting started with an inductive proof, try following this template.

3.4.3 Strong Induction

So far, in all of the inductive proofs we have considered, the inductive step involved shows something about a level k object using a hypothesis about a level $k-1$ object. However, there are times when we will need to use several previous levels ($k-1, k-2, \dots$, etc.) to justify the inductive step. The verification of a closed-form solution for the Fibonacci sequence is a simple example.

Recall that the recurrence relation for these numbers,

$$F(n) = \begin{cases} 1 & \text{if } n = 1 \text{ or } n = 2 \\ F(n-1) + F(n-2) & \text{if } n > 2, \end{cases}$$

is a little more complicated than the other recurrence relations we have studied, since the recursive part refers to *two* previous values of F. Observe how the proof differs accordingly.

Theorem 3.6 *For $n \geq 1$, the nth Fibonacci number is*

$$F(n) = \frac{\alpha^n - \beta^n}{\alpha - \beta} \tag{3.4.1}$$

where

$$\alpha = \frac{1+\sqrt{5}}{2} \text{ and } \beta = \frac{1-\sqrt{5}}{2}.$$

Proof It is easy to check that α and β are the solutions to the following equation:

$$x^2 = x + 1. \tag{3.4.2}$$

Thus we can use this equation as an identity for both α and β. We proceed by induction on n.

Base Case: If $n = 1$ or $n = 2$, the recurrence relation says that $F(1) = 1 = F(2)$. Formula 3.4.1 gives

$$F(1) = \frac{\alpha^1 - \beta^1}{\alpha - \beta} = \frac{\alpha - \beta}{\alpha - \beta} = 1$$

and

$$\begin{aligned} F(2) &= \frac{\alpha^2 - \beta^2}{\alpha - \beta} \\ &= \frac{(\alpha + 1) - (\beta + 1)}{\alpha - \beta}, \text{ using Equation 3.4.2} \\ &= \frac{\alpha - \beta}{\alpha - \beta} \\ &= 1, \end{aligned}$$

so the closed-form solution is correct for $n = 1$ and $n = 2$.

Inductive Hypothesis: Let $k > 2$. Suppose as inductive hypothesis that

$$F(i) = \frac{\alpha^i - \beta^i}{\alpha - \beta}$$

for all i such that $1 \le i < k$.

Inductive Step: Using the recurrence relation,

$$\begin{aligned} F(k) &= F(k-1) + F(k-2) \\ &= \frac{\alpha^{k-1} - \beta^{k-1}}{\alpha - \beta} + \frac{\alpha^{k-2} - \beta^{k-2}}{\alpha - \beta}, \text{ by inductive hypothesis} \\ &= \frac{\alpha^{k-2}(\alpha + 1) - \beta^{k-2}(\beta + 1)}{\alpha - \beta} \\ &= \frac{\alpha^{k-2}(\alpha^2) - \beta^{k-2}(\beta^2)}{\alpha - \beta}, \text{ using Equation 3.4.2} \\ &= \frac{\alpha^k - \beta^k}{\alpha - \beta} \end{aligned}$$

as required. □

Look back at this proof. The inductive step requires us to use two previous values of $F(n)$. That's why the inductive hypothesis needs to be stronger. Instead of assuming "Statement$(k-1)$, for some k," the inductive hypothesis has the form "Statement(i), for all $i < k$, for some k." This is known as *strong induction*; we state this as another principle.

The Second Principle of Mathematical Induction. To prove the statement

$$\text{“Statement}(n)\text{, for all } n \geq 1\text{,”}$$

it suffices to prove

1. Statement(1), and
2. Statement$(1) \wedge$ Statement$(2) \wedge \cdots \wedge$ Statement$(k-1) \Rightarrow$ Statement(k), for $k > 1$.

In other words, strong induction takes as its inductive hypothesis *all* previous cases, not just the immediate predecessor. As with simple induction, we are establishing a chain of implications that demonstrate Statement(n) for any value of n.

$$\begin{aligned}
\text{Statement}(1) &\Rightarrow \text{Statement}(2)\\
\text{Statement}(1) \wedge \text{Statement}(2) &\Rightarrow \text{Statement}(3)\\
\text{Statement}(1) \wedge \text{Statement}(2) \wedge \text{Statement}(3) &\Rightarrow \text{Statement}(4)\\
\cdots &\Rightarrow \cdots
\end{aligned}$$

As before, each implication is of the form $P \Rightarrow Q$, but this time P is the conjunction of *all* previously established cases. In other words, P is a stronger assumption. This can make proving the inductive step easier, because you have more to work with.

The form of an inductive proof is the same with strong induction as it was with simple induction: prove the base case, state the inductive hypothesis, and use the recursive definition to prove the next case. Only the inductive hypothesis is different.

Example 3.25 Theorem 2.9 states that a connected graph with no simple circuits is a tree. We can use strong induction to show that binary trees (as defined in Example 3.22) are connected and have no simple circuits.

Proof (Strong induction on the height of the tree.)

Base Case: A binary tree of height 0 consists of only a single vertex, so this graph is connected and has no simple circuits.

Inductive Hypothesis: Suppose as inductive hypothesis that any binary tree with height less than k is connected and has no simple circuits, for some $k > 1$.

Inductive Step: Suppose as given a binary tree T with height k. By the definition in Example 3.22, T consists of a root node r with edges going to two subtrees T_1 and T_2, each having height less than k. By inductive hypothesis, T_1 and T_2 are each connected and have no simple circuits. Since T_1 and T_2 are connected, there is a path from r to any vertex in T_1 and T_2, and therefore there is a path from any vertex in T to any other vertex. So T is connected, as required. Since T_1 and T_2 have no simple circuits, any simple circuit in T must pass through r. But in order for a circuit to begin and end in T_1 (or T_2), the circuit must pass through r twice—once to get out of T_1, and once to get back in. Such a circuit is not simple. So T has no simple circuits, as required. □

In the following example, notice that we must suppose that *all* previous primes are products of primes in the inductive step. If we made only the weaker hypothesis that the preceding prime was a product of primes—as with simple induction—the argument wouldn't work.

Theorem 3.7 *Every integer $n \geq 2$ is either prime or the product of primes.*

Proof (Strong induction on n.)

Base Case: The only factors of 2 are 1 and 2, so 2 is prime.

Inductive Hypothesis: Let $k > 2$ be given. Suppose as inductive hypothesis that every i is such that $2 \leq i < k$ is either prime or the product of primes.

Inductive Step: If k is prime, we are done. If k is not prime, then $k = pq$ for some $p \geq 2$ and $q \geq 2$. And since $k = pq$, p and q are both less than k. By inductive hypothesis, p and q are both either prime or products of primes, so $k = pq$ is the product of primes. □

3.4.4 Structural Induction

In all of the previous examples, induction has been "on" some discrete quantity: proofs involving formulas of n tend to use use induction on n; proofs about line maps use induction on the number of lines, etc. In some cases, however, it can be awkward to specify this quantity.

Example 3.26 Define a set $X \subseteq \mathbf{Z}$ recursively as

B. $4 \in X$.

$\mathbf{R_1}$. If $x \in X$ then $x - 12 \in X$.

$\mathbf{R_2}$. If $x \in X$ then $x^2 \in X$.

Prove that every element of X is divisible by 4.

Before looking at the proof, notice that the two recursive cases move in opposite directions: changing x to $x-12$ returns a smaller number, while changing x to x^2 returns a larger number. So it isn't natural to write the statement we are trying to prove in terms of n, for $n \geq 1$. The point of a structural induction proof is that the recursive case of the definition maintains the property of divisibility by 4.

Proof of Example 3.26 (Induction on the recursive definition of X.)

> Base Case: Since $4 = 1 \cdot 4$, $4 \mid 4$, so the claim holds for the base case of the definition.
>
> Inductive Hypothesis: Suppose as inductive hypothesis that some $x \in X$ is divisible by 4. Then $x = 4a$ for some integer a.
>
> Inductive Step: Now $x - 12 = 4a - 12 = 4(a-3)$, and $x^2 = (4a)^2 = 4(4a^2)$, so both $x - 12$ and x^2 are divisible by 4. Therefore cases $\mathbf{R_1}$ and $\mathbf{R_2}$ always produce integers that are divisible by 4 (given that $4 \mid x$), and the base case $\mathbf{B}$ gives an integer that is divisible by 4. So, by induction, all elements of X are divisible by 4. □

Pay particular attention to the inductive hypothesis in this proof: it is supposing that some element $x \in X$ has the desired property. In earlier proofs, we thought of x as an "object of level $k-1$," and we proved something about all "objects of level k." Here we avoid the issue of how many levels an object has, focusing on the base and recursive cases of the definition. We call this "induction on the recursive definition" because we are really doing induction on the number of times the recursive part of the definition is used to obtain an element of X. We show that zero uses of the recursive part of the definition produces an element with the desired property, and then we suppose as inductive hypothesis that $k-1$ uses of the recursive definition yields an element x with the desired property. Finally, we show that one more use of the recursive definition produces an element with the desired property. All this while, the variable k operates in the background, so it isn't really necessary to mention it in the proof.

This type of induction is also called "structural induction," because it uses the recursive structure of an object to guide the inductive argument.

Example 3.27 Theorem 3.5 states that any line map can be two-colored. Here is an alternate version of the proof of this theorem using structural induction.

Proof (Induction on the definition of a line map [Example 3.19].)

> Base Case: The base case of a line map is a blank rectangle, which can be two-colored because it can be one-colored.

Inductive Hypothesis: Suppose as inductive hypothesis that some line map M' can be two-colored. Make a two-coloring of M'.

Inductive Step: The recursive part of the definition says that a new line map M can be formed from M' by drawing some line l all the way across M'. Now reverse the colors of all the regions on one side of the line l. Each side will still be correctly two-colored, and any regions having l as a border will be opposite colors. Hence the map M is correctly two-colored. So by induction, the recursive definition of a line map always gives a two-colorable map. □

Remember that mathematical induction is a recursive technique; self-reference occurs in the inductive step of the proof. This is why many of the examples in this section make use of the recursive definitions from Section 3.3. Often the presence of a recursive definition indicates the need for an inductive proof, and the key to constructing an inductive argument lies in thinking recursively about the problem.

Exercises 3.4

1. Prove that the sum of the first n odd natural numbers is n^2.

2. You may already have a notion of what a *convex* region is, but here's a mathematical definition.

   Definition 3.2 A region R is *convex* if the line segment connecting any two points in R lies in R. A polygon is convex if it and its interior form a convex region.

   A consequence of this definition is that all the diagonals of a convex polygon lie inside the polygon. Use induction to prove that a convex n-gon has $n(n-3)/2$ diagonals. (Hint: Think of an n-gon as having an $(n-1)$-gon inside of it.)

3. Use induction to prove that the sum of the angles of a convex n-gon is $180(n-2)$ degrees.

4. Prove that any palindrome with an even number of letters can be constructed using the definition in Example 3.16. Use induction on n, where $2n$ is the length of the palindrome. So you need to prove that the string

$$a_n a_{n-1} \cdots a_2 a_1 a_1 a_2 \cdots a_{n-1} a_n$$

   can be constructed, for all $n \geq 1$.

5. Prove that any palindrome with an odd number of letters can be constructed using the definition in Example 3.16.

6. Prove by induction that all of the hexagonal numbers are odd. (See Example 3.4.)

7. Recall the definition of a line map (Example 3.19).

   (a) Prove by induction that a line map with n distinct lines has at least $n + 1$ regions.

   (b) Prove by induction that a line map with n distinct lines has at most 2^n regions.

   (c) Part (a) gives a lower bound on the number of regions in a line map. For example, a line map with five lines must have at least six regions. Give an example of a line map that achieves this lower bound, that is, draw a line map with five lines and six regions.

   (d) Part (b) says that a line map with three lines can have at most eight regions. Can you draw a line map with three lines that achieves this upper bound? Do so, or explain why you can't.

8. Let the following definitions be given, where s is a string.

   Definition 1. Define the number $l(s)$ as follows.

   B₁. $l(s) = 0$ if s is the empty string.

   B₂. $l(s) = 1$ if s is a single symbol.

   R. $l(s) = l(x) + l(y)$ if $s = xy$.

   Definition 2. Let n be a natural number. Define the string ns as follows:

   B. $1s = s$.

   R. $ns = (n - 1)s$ if $n > 1$.

   Use these definitions to prove that $l(ns) = nl(s)$ for all $n \geq 1$.

9. Use the recursive definition in Example 3.22 to prove that a binary tree with height n has less than 2^{n+1} nodes.

10. Refer to Exercise 15 of Section 3.3 for the definition of a full binary tree.

    (a) Use strong induction to prove that a full binary tree has an odd number of nodes.

    (b) Prove that a full binary tree has an even number of edges.

11. In Exercise 3 of Section 3.1, we defined the following recurrence relation:

$$H(n) = \begin{cases} 0 & \text{if } n \leq 0 \\ 1 & \text{if } n = 1 \text{ or } n = 2 \\ H(n-1) + H(n-2) - H(n-3) & \text{if } n > 2 \end{cases}.$$

    Prove that $H(2n) = H(2n-1) = n$ for all $n \geq 1$.

12. The Lucas numbers $L(n)$ are defined in Exercise 4 of Section 3.1, and the Fibonacci numbers $F(n)$ are given by Definition 3.1. Prove that $L(n) = F(n-1) + F(n+1)$ for all $n \geq 2$.

13. Prove that $B(n)$, the nth term in the sequence of shapes whose limit is the Badda-Bing fractal of Example 1.17, has $4 \cdot 3^{n-1}$ free vertices (i.e., vertices that lie on only one square) for all $n \geq 1$.

14. Prove that $B(n)$, the nth term in the sequence of shapes whose limit is the Badda-Bing fractal of Example 1.17, consists of $2 \cdot 3^{n-1} - 1$ squares, for $n \geq 1$.

15. Prove that $K(n)$, the nth term in the sequence of shapes whose limit is the Koch snowflake fractal of Example 3.23, has perimeter $3 \cdot (4/3)^{n-1}$, where the equilateral triangle in $K(1)$ has side length of 1 unit.

16. Find a formula for the area of (the black part of) $S(n)$, the nth term in the sequence of shapes whose limit is the Sierpinski gasket fractal in Figure 3.11 on page 174. Assume that $S(1)$ is a black equilateral triangle with area 1. Prove that your formula is correct.

17. Let X be the set defined in Example 3.21.

    (a) Prove, by induction on n, that $2n+1 \in X$ for all $n \geq 0$. (This shows that X contains all the odd natural numbers.)

    (b) Prove by induction that every element in X is odd. (This shows that the set of all odd natural numbers contains X.)

    (c) Together, what do (a) and (b) show?

18. Define a set X recursively as follows.

    B. $2 \in X$.

    R. If $x \in X$, so is $x + 10$.

    Use induction to prove that every element of X is even.

19. Define a set X recursively as follows.

    B. 3 and 7 are in X.

    R. If x and y are in X, so is $x + y$. (Here it is possible that $x = y$.)

    Prove that, for every $n \geq 12$, $n \in X$. (Hint: For the base case, show that 12, 13, and 14 are in X.)

20. In the game of chess, a knight moves by jumping to a square that is two units away in one direction and one unit away in another. For example, in Figure 3.18, the knight at K can move to any of the squares marked with an asterisk $*$. Prove by induction that a knight can move from any square to any other square on an $n \times n$ chessboard via a sequence of moves, for all $n \geq 4$.

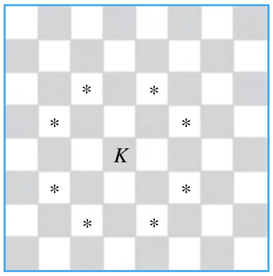


Figure 3.18 A knight can move to a square with a 2×1 L-shaped jump.

3.5 Recursive Data Structures

In this section we will see how thinking recursively about how to organize data can lead to elegant solutions. We will give recursive definitions for lists and trees, and we will use these definitions to define useful functions on these structures. Moreover, we will prove that these functions do what they are supposed to do. Since the definitions and functions are recursive, most of these proofs will use induction.

3.5.1 Lists

Almost every computer application uses some kind of list object. A *list* is a set of data elements in some sequential order

$$x_1, x_2, x_3, \ldots, x_n$$

where all of the x_i's are of the same type (e.g., integers, strings, etc.). You can make a list of n elements by adding an element to the end of a list of $n-1$ elements; this observation inspires the following recursive definition.

Definition 3.3 Let X be a set. A *list* of elements of X is:

B. x where $x \in X$.

R. L, x where $x \in X$ and L is a list of elements of X.

Notice that, unlike a set, a list may repeat the same element several times, and the order of the elements matters. Every symbol in this definition is important; the commas between the list elements are part of the structure of a list. For example, we can build up the list of strings

$$\texttt{cubs}, \texttt{bears}, \texttt{bulls}, \texttt{cubs}$$

in a bottom-up fashion using this definition.

$$\begin{array}{lllll} L_1 & = & \texttt{cubs} & & \text{by part } \mathbf{B} \\ L_2 & = & L_1, \texttt{bears} & = \texttt{cubs}, \texttt{bears} & \text{by part } \mathbf{R} \\ L_3 & = & L_2, \texttt{bulls} & = \texttt{cubs}, \texttt{bears}, \texttt{bulls} & \text{by part } \mathbf{R} \\ L_4 & = & L_3, \texttt{cubs} & = \texttt{cubs}, \texttt{bears}, \texttt{bulls}, \texttt{cubs} & \text{by part } \mathbf{R} \end{array}$$

One advantage to defining lists recursively is that it makes it possible to define recursive functions that tell us something about the data in the list. For example, we can use recursion to add up all the elements in a list of integers.

Definition 3.4 Let L be a list as defined by Definition 3.3, where $X = \mathbf{R}$, the real numbers. Define a function $\text{Sum}(L)$ recursively as follows.

B. If $L = x$, a single number, then $\text{Sum}(L) = x$.

R. If $L = L'$, x for some list L', then $\text{Sum}(L) = \text{Sum}(L') + x$.

Notice how the base and recursive cases of this definition match the base and recursive cases of Definition 3.3. The recursive structure of a list determines the way we write recursive functions.

Example 3.28 To evaluate the Sum function on the list $3, 1, 4, 2$, it is natural to take a top-down approach.

$$\begin{aligned} \text{Sum}(3,1,4,2) &= \text{Sum}(3,1,4) + 2 && \text{by part } \mathbf{R} \\ &= \text{Sum}(3,1) + 4 + 2 && \text{by part } \mathbf{R} \\ &= \text{Sum}(3) + 1 + 4 + 2 && \text{by part } \mathbf{R} \\ &= 3 + 1 + 4 + 2 && \text{by part } \mathbf{B} \\ &= 10. \end{aligned}$$

The Sum function returns the correct answer for the list $3, 1, 4, 2$, but will it always work? We can prove that it does, using induction.

Theorem 3.8 *Let L be the list $x_1, x_2, x_3, \dots, x_n$, where the x_i's are numbers. Then*

$$\text{Sum}(L) = x_1 + x_2 + x_3 + \cdots + x_n$$

for all $n \geq 1$.

Proof (Induction on the size of the list.)

Base Case: If L contains only a single number x, then the base case of the definition stipulates that $\text{Sum}(L) = x$, as required.

Inductive Hypothesis: Let $k > 1$. Suppose as inductive hypothesis that

$$\text{Sum}(L') = x_1 + x_2 + x_3 + \cdots + x_{k-1}$$

for any list L' containing $k - 1$ elements.

Inductive Step: Suppose as given a list

$$L = x_1, x_2, x_3, \dots, x_k$$

with k elements. Then, by Definition 3.3, $L = L', x_k$, where L' is a list of $k - 1$ elements. Therefore,

$$\begin{aligned}
\text{Sum}(L) &= \text{Sum}(L') + x_k && \text{by part } \mathbf{R} \\
&= (x_1 + x_2 + x_3 + \cdots + x_{k-1}) + x_k && \text{by inductive hypothesis} \\
&= x_1 + x_2 + x_3 + \cdots + x_k
\end{aligned}$$

as required. □

The next recursive definition is for educational purposes only; the object it defines is simple and somewhat limited. The point is to help us study ways to search for an element in a list.

Definition 3.5 An *SList* is

B. x where $x \in \mathbf{R}$, the real numbers.

R. (X, Y) where X and Y are SLists having the same number of elements, and the last number in X is less than the first number in Y.

For example, $(((1, 3), (8, 9)), ((12, 16), (25, 30)))$ is an SList. Notice that SLists always have 2^p elements, for some $p \geq 0$. (The proof of this fact is left as an exercise.) The number p counts the *depth* of parentheses of the SList, or more simply, the depth of the SList; every number in the list will be inside p pairs of parentheses. So the example SList above has depth 3 and contains 2^3 elements. Also notice that if $L = (X, Y)$ is an SList of depth p, then X and Y must have depth $p - 1$.

What's the "S" for? The elements of an SList must be *sorted* in order from left to right. That's another exercise.

Suppose we want to define a function that will tell us whether or not a given number is in the list. Since the list is defined recursively, it is natural to define the function recursively as well. Note that the base and recursive cases of the following function correspond to the base and recursive cases of Definition 3.5.

Definition 3.6 Define a true or false function $\text{Search}(t, L)$, where t is a number (the "target") and L is an SList, as follows.

B. Suppose $L = x$, a list of depth 0. Then

$$\text{Search}(t, L) = \begin{cases} \text{true} & \text{if } t = x \\ \text{false} & \text{if } t \neq x. \end{cases}$$

R. Suppose the depth of L is greater than 0, so $L = (X, Y)$. Then

$$\text{Search}(t, L) = \text{Search}(t, X) \vee \text{Search}(t, Y).$$

The Search function is supposed to tell if a given element is in the list. For example, let $L = (((1, 3), (8, 9)), ((12, 16), (25, 30)))$. The following calculation shows how to calculate the Search function on this list, with a target value of 8:

$$\begin{aligned}
\text{Search}[8, L] &= \text{Search}[8, ((1, 3), (8, 9))] \vee \text{Search}[8, ((12, 16), (25, 30))] \\
&= \text{Search}[8, (1, 3)] \vee \text{Search}[8, (8, 9)] \vee \text{Search}[8, (12, 16)] \\
&\quad \vee \text{Search}[8, (25, 30)] \\
&= \text{Search}[8, 1] \vee \text{Search}[8, 3] \vee \text{Search}[8, 8] \vee \text{Search}[8, 9] \\
&\quad \vee \text{Search}[8, 12] \vee \text{Search}[8, 16] \vee \text{Search}[8, 25] \vee \text{Search}[8, 30] \\
&= \text{false} \vee \text{false} \vee \text{true} \vee \text{false} \vee \text{false} \vee \text{false} \vee \text{false} \vee \text{false} \\
&= \text{true}.
\end{aligned}$$

Notice how to evaluate a recursive function: rewrite the function in terms of itself using the recursive step, until (hopefully) you are able to evaluate the function using the base case. Once the recursion "bottoms out," you can evaluate the function because there are no more recursive references to the function.

$\text{Search}[8, L]$ worked, because it returned "true" and 8 was indeed in the list. Here is a proof that the Search function works in general.

Theorem 3.9 $\text{Search}(t, L) \iff$ *t is in L.*

Proof (Induction on p, the depth of the SList L.)

Base Case: If $p = 0$, the list L contains only a single number, say $L = x$. The base case of the Search function will set $\text{Search}(t, L) = \text{true}$ if and only if $t = x$. Therefore

$$\text{Search}(t, L) \iff t = x \iff t \text{ is in } L$$

when the depth of the list is 0.

Inductive Hypothesis: Suppose as inductive hypothesis that the Search function works for any list L' of depth $k - 1$, for some $k > 0$. That is, if L' has depth $k - 1$, then we suppose that

$$\text{Search}(t, L') \iff t \text{ is in } L'.$$

Inductive Step: Given a list L of depth k, we know that $L = (X, Y)$ for some X and Y of depth $k - 1$. So

$$\text{Search}(t, L) = \text{Search}(t, X) \vee \text{Search}(t, Y)$$

by the recursive part of the function definition. Now

$$\begin{array}{rcll} t \text{ is in } L & \iff & (t \text{ is in } X) \vee (t \text{ is in } Y) & \text{by the definition of SList} \\ & \iff & \text{Search}(t, X) \vee \text{Search}(t, Y) & \text{by inductive hypothesis} \\ & \iff & \text{Search}(t, L) & \text{by the definition of Search.} \end{array}$$

So for any SList of any depth, the Search function will be true if and only if the target is in the list. □

This Search function isn't very clever: it does not use the fact that an SList has to be in order. Let's look at another function designed to test whether a given number is in an SList.

Definition 3.7 Define a true or false function $\text{BSearch}(t, L)$, where t is a number and L is an SList, as follows.

B. Suppose $L = x$, a list of depth 0. Then

$$\text{BSearch}(t, L) = \begin{cases} \text{true} & \text{if } t = x \\ \text{false} & \text{if } t \neq x. \end{cases}$$

R. Suppose L has depth $p > 0$, so $L = (X, Y)$. Let r be the last element of X. Then

$$\text{BSearch}(t, L) = \begin{cases} \text{BSearch}(t, Y) & \text{if } t > r \\ \text{BSearch}(t, X) & \text{if } t \not> r. \end{cases}$$

For example, let $L = (((1,3),(8,9)),((12,16),(25,30)))$, and try finding the number 8 using BSearch. Compare this calculation to the same search using the old (not smart) Search function:

$$\begin{aligned}
\text{BSearch}[8, L] &= \text{BSearch}[8, ((1,3),(8,9))] && \text{since } 8 \not> 9 \\
&= \text{BSearch}[8, (8,9)] && \text{since } 8 > 3 \\
&= \text{BSearch}[8, 8] && \text{since } 8 \not> 8 \\
&= \text{true} && \text{since } 8 = 8.
\end{aligned}$$

BSearch appears to be a more clever function. It looks like the BSearch function takes less work (and time) to evaluate than the Search function.

3.5.2 Efficiency

If these two search functions were implemented on a computer, we would probably expect BSearch to run more efficiently. How much better is it? To answer this question, we can try to count the number of operations each function does. In practice, instead of counting every single operation, computer scientists try to count the number of occurrences of the most time-consuming operation. For our functions, assume that these are the comparisons $t \overset{?}{=} x$ and $t \overset{?}{>} r$.

First consider the Search function. Let $C(p)$ be the number of times the comparison $t \overset{?}{=} x$ is done when searching a list of depth p. If $p = 0$, then the list contains only a single item, so the base case **B** of the definition gets used and one comparison is made. Therefore $C(0) = 1$.

Now suppose the Search function is evaluated on a list of depth p, for some $p > 0$. Then the recursive case **R** of the definition gets used, and the Search function is executed twice on lists of depth $p - 1$. Each of these two recursive calls uses $C(p - 1)$ comparisons. Therefore $C(p)$ must satisfy the following recurrence relation:

$$C(p) = \begin{cases} 1 & \text{if } p = 0 \\ 2C(p-1) & \text{if } p > 0. \end{cases} \tag{3.5.1}$$

This recurrence relation is easy to solve: $C(p) = 2^p$. Since an SList of depth p contains 2^p elements, the number of comparisons made by the Search function equals the number of elements in the list.

Similarly, we can derive a recurrence relation for the BSearch function. For a list containing a single item, the base case of the function requires one comparison. On a list of depth p, $p > 0$, the recursive part of the definition first makes one comparison, and then calls the BSearch function on a list of depth

$p-1$. If we use $D(p)$ to represent the number of comparisons needed, we get the following recurrence relation:

$$D(p) = \begin{cases} 1 & \text{if } p = 0 \\ 1 + D(p-1) & \text{if } p > 0. \end{cases} \tag{3.5.2}$$

This recurrence relation is even easier to solve: $D(p) = p+1$. So you only need $p+1$ comparisons to find an element in a list of size 2^p. In other words, to find a number in a list of N elements using BSearch, you need to make $\log_2 N + 1$ comparisons.

As the size of the list gets large, BSearch becomes a much better alternative to Search. For example, a list containing 1,048,576 elements requires 1,048,576 comparisons using Search, but only 21 comparisons using BSearch. Such considerations are important when writing computer programs that use these functions. We will study these issues in more depth in Chapter 5.

Here's one more recursive function that will be used in the exercises.

Example 3.29 Define a numerical function $\text{Sum}(L)$, where L is an SList, as follows.

B. If $L = n$, then $\text{Sum}(L) = n$.

R. If $L = (X, Y)$, then $\text{Sum}(L) = \text{Sum}(X) + \text{Sum}(Y)$.

This version of the Sum function differs from Definition 3.4 in exactly the ways that SLists differ from lists; the recursive definition of the object guides the recursive definition of the function.

3.5.3 Binary Search Trees Revisited

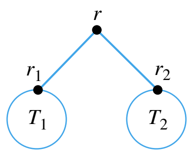
We have already considered recursive definitions for binary trees in general (Example 3.22). Now that we are used to thinking recursively about data structures, it is natural to write down the following recursive definition for a binary search tree. (Compare this definition with the discussion in Example 2.5 on page 68.)

Definition 3.8 Let S be a set that is totally ordered by $\leq$. A *binary search tree* on S is

B$_1$. The empty tree, or

B$_2$. a single vertex $r \in S$. In this case, r is the root of the tree.

R. If T_1 and T_2 are binary search trees with roots r_1 and r_2 respectively, and if $a \leq r$ for all nodes $a \in T_1$ and $r \leq b$ for all nodes $b \in T_2$, then the tree



is a binary search tree with root r.

You should convince yourself that the procedure outlined in Example 2.5 always produces a binary search tree. The key observation is that part **R** of the recursive definition is satisfied.

Given a binary search tree, we would like to be able to produce a listing of its nodes in order. We can define such a listing, separated by commas, as a recursive function.

Definition 3.9 Define a function InOrder(T), where T is a binary search tree,[3] as follows.

B_1. If T is the empty tree, InOrder(T) = “” (the empty listing).

B_2. If T is a single node r, then InOrder(T) = “r”.

R. If T has root r and subtrees T_1 and T_2, then

$$\text{InOrder}(T) = \text{“InOrder}(T_1), r, \text{InOrder}(T_2)\text{”}$$

where the commas are part of the listing unless T_1 or T_2 is empty.

Let’s step through this definition for the binary search tree in Figure 2.7 on page 69. Let T represent the whole tree, let L represent the subtree on the left containing complexify, clueless, and jazzed, and let R be the subtree on the right containing poset, phat, and sheafify. Then

$$\begin{aligned}\text{InOrder}(T) &= \text{InOrder}(L), \text{macchiato}, \text{InOrder}(R)\\ &= \text{InOrder}(\text{clueless}), \text{complexify}, \text{InOrder}(\text{jazzed}),\\ &\quad\ \text{macchiato},\\ &\quad\ \text{InOrder}(\text{phat}), \text{poset}, \text{InOrder}(\text{sheafify})\\ &= \text{clueless, complexify, jazzed, macchiato, phat, poset, sheafify.}\end{aligned}$$

So InOrder produces a listing of the words in alphabetical order. To see that InOrder always does this, observe that part **R** of Definition 3.9 always produces words that are in order, provided part **R** of Definition 3.8 holds.

3. Observe that this definition applies to a generic binary tree. We will discuss this and similar definitions further in Chapter 5.

Exercises 3.5

1. Let L be a list, as in Definition 3.3. Define a numerical function f as follows.

   B. If $L = x$, a single element, then $f(L) = 1$.

   R. If $L = L', x$ for some list L', then $f(L) = f(L') + 1$.

   (a) Show the steps of a "top-down" computation, as in Example 3.28, to find the value of $f(\texttt{veni}, \texttt{vidi}, \texttt{vici})$.

   (b) What does the value of $f(L)$ tell you about the list L, in general?

   (c) Prove your assertion in part (b), using induction.

2. Consider the following function p, where L is a list.

   B. If $L = x$, a single element, then $p(L) =$ "x".

   R. If $L = L', x$ for some list L', then $p(L) =$ "$x, p(L')$".

   (a) If $L = \texttt{john}, \texttt{paul}, \texttt{george}, \texttt{ringo}$, what is $p(L)$?

   (b) What does the function p do, in general?

   (c) Prove your assertion in part (b), using induction.

3. Define a function max: $\mathbf{R} \times \mathbf{R} \longrightarrow \mathbf{R}$ by

$$\max(a, b) = \begin{cases} a & \text{if } a > b \\ b & \text{if } b \geq a. \end{cases}$$

   (a) Use this function to write a recursive function LMax(L) that returns the greatest value in L, where L is a list of numbers.

   (b) Prove that your LMax function works. In other words, prove that LMax(L) returns the greatest value in the list L.

4. Prove that the number of elements in any SList of depth p is 2^p. (Use induction on p.)

5. Prove that the elements of an SList are in strictly increasing order from left to right. That is, if $x_1, x_2, x_3, \ldots, x_{2^p}$ are the elements of the list, show that

$$x_1 < x_2 < x_3 < \cdots < x_{2^p}.$$

   (Use induction on p.)

6. Prove that the BSearch function (Definition 3.7) works. In other words, prove that
$$\text{BSearch}(t, L) \iff t \text{ is in } L.$$

7. Prove that the Sum function of Example 3.29 works. In other words, prove, for any $p \geq 0$, that if $a_1, a_2, \ldots, a_{2^p}$ are the elements of L, then $\text{Sum}(L) = a_1 + a_2 + \cdots + a_{2^p}$.

8. Suppose L is an SList of depth p. Find a recurrence relation for $A(p)$, the number of times two numbers are added when evaluating Sum(L).

9. Write a recursive function $d(L)$ that returns the depth of an SList L.

10. Write a recursive function $a(L)$ that computes the average of an SList L. Use the fact that the average of a list is the average of the averages of each half.

11. Let $L = ((10, 20), (30, 40))$ be an SList.

    (a) Compute Search$(15, L)$, showing all steps.

    (b) Compute BSearch$(15, L)$, showing all steps.

12. Let $L = (((15, 25), (35, 45)), ((50, 60), (70, 80)))$ be an SList.

    (a) Compute Search$(15, L)$, showing all steps.

    (b) Compute BSearch$(15, L)$, showing all steps.

13. Modify the InOrder function (Definition 3.9) so that it lists the items in a binary search tree in reverse order.

*14. Write a recursive search function for finding an element in a binary search tree. You should use the notion of a left or right subtree of a node in your definition. (In Definition 3.8, the left and right subtrees are T_1 and T_2, respectively.) Make sure you account for empty nodes.

15. Define a *UList* as follows. A *UList* is

    B. a number x, or

    R. a pair (X, Y), where X and Y are ULists.

    (a) Use induction to prove that, for any $n \geq 1$, a UList containing n numbers can be constructed.

    (b) Write a definition of a recursive function Product(L) that inputs a UList and returns the product of all the numbers in the UList.

(c) Define a USearch function that tells whether a given number is in the list.

(d) Prove that your USearch function works.

16. Define a *NumberSquare* as

B. A single number x.

R. A diagram

$$\begin{bmatrix} S_1 & S_2 \\ S_3 & S_4 \end{bmatrix}$$

where S_1, S_2, S_3, S_4 are NumberSquares each containing the same amount of numbers.

Here are three examples of NumberSquares:

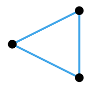
$$4, \qquad \begin{bmatrix} 4 & 17 \\ 13 & 1 \end{bmatrix}, \qquad \begin{bmatrix} \begin{bmatrix} 3 & 12 \\ 11 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 1 \\ 2 & 4 \end{bmatrix} \\ \begin{bmatrix} 6 & 10 \\ 7 & 3 \end{bmatrix} & \begin{bmatrix} 4 & 17 \\ 13 & 1 \end{bmatrix} \end{bmatrix}.$$

Define a recursive function Trace(S) that returns the sum of the upper-left/lower-right diagonal of the NumberSquare S. (For the examples above, the Trace function should return 4, 5, and 15, respectively.)

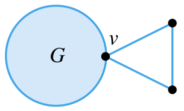
17. Refer to the definition of NumberSquare in Exercise 16. The *depth* of a NumberSquare is the number of pairs of brackets needed to write the square. (So the given examples have depth 0, 1, and 2, respectively.) Prove, using induction on p, that a NumberSquare of depth p has 4^p numbers in it.

18. Define a *TGraph* as follows.

B. This is a TGraph:



R. If G is a TGraph and v is a vertex of G, then this is also a TGraph:



Draw an example of a TGraph with seven vertices.

19. Refer to Exercise 18. Prove, by induction, that every vertex in a TGraph has even degree.

20. Refer to Exercise 18. Prove, by induction, that any TGraph can be three-colored.