

1991

Modern Algebra and Discrete Structures

R. F. Lax

Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: <https://digitalcommons.lsu.edu/etext>



Part of the [Algebra Commons](#)

Recommended Citation

Lax, R. F., "Modern Algebra and Discrete Structures" (1991). *E-Textbooks*. 3.
<https://digitalcommons.lsu.edu/etext/3>

This Book is brought to you for free and open access by LSU Digital Commons. It has been accepted for inclusion in E-Textbooks by an authorized administrator of LSU Digital Commons. For more information, please contact ir@lsu.edu.

MODERN ALGEBRA and DISCRETE STRUCTURES

R. F. Lax

Louisiana State University



ADDISON-WESLEY

An imprint of Addison Wesley Longman, Inc.

Reading, Massachusetts • Menlo Park, California • New York • Harlow, England
Don Mills, Ontario • Sydney • Mexico City • Madrid • Amsterdam

Sponsoring Editor: Peter Coveney/George Duda
Project Editor: David Nickol
Art Direction/Cover Coordinator: Heather A. Ziegler
Cover Design: Wanda Lebelska Design
Production: Kewal K. Sharma

Modern Algebra and Discrete Structures

Copyright © 1991 Addison-Wesley Educational Publishers Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

Library of Congress Cataloging-in-Publication Data

Lax, R. F.
Modern algebra and discrete structures / R.F. Lax.
p. cm.
Includes index.
ISBN 0-06-043878-9
1. Algebra. I. Title.
QA154.2.L39 1991
512—dc20

90-4710
CIP

To my family

Contents

Preface vii

CHAPTER I: Preliminaries 1

- 1. Sets and Counting 1
- 2. The Integers 8
- 3. Finite-State Machines 19

CHAPTER II: Relations 28

- 1. Relations and Mappings 28
- 2. Relations on Finite Sets 34
- 3. Equivalence Relations and Partitions 39
- 4. Minimization of Machines 44

CHAPTER III: Semigroups, Monoids, and Groups 55

- 1. Examples 55
- 2. The Group Axioms and More Examples 62
- 3. Subobjects 68
- 4. Cosets and Lagrange's Theorem 76
- 5. Congruences and Quotients 80
- 6. Homomorphisms 89
- 7. Semigroups and Machines 97

8. Dihedral Groups and Permutation Groups; Cayley's Theorem	106
9. Group Actions and Burnside's Theorem	117
10. Pólya Enumeration Theory	125
11. Conjugacy and the Class Equation	134
CHAPTER IV: Rings and Fields	140
1. Definitions	140
2. \mathbb{Z}_n	146
3. Ideals and Ring Homomorphisms	152
4. The Chinese Remainder Theorem	158
5. Public-Key Encryption	164
6. Rings of Polynomials	172
7. Creating Roots and Extending Fields	183
8. Finite Fields	192
9. Applications of Finite Fields to Combinatorial Design	199
CHAPTER V: Algebraic Coding Theory	209
1. An Example	210
2. Linear Codes	214
3. Error Detection and Correction	226
4. Coset Decoding	236
5. Hamming Codes	244
6. Polynomial and Cyclic Codes	251
CHAPTER VI: Boolean Algebras and Switching Functions	265
1. Axioms and Examples	265
2. Finite Boolean Algebras	274
3. Gating Networks and Switching Functions	280
4. Disjunctive Normal Form and Prime Implicants	289
5. Minimization	296
CHAPTER VII: Graph Theory	304
1. Graphs	305
2. Digraphs	314
3. Trees	322
4. Networks	332
References	345
Solutions or Suggestions for Selected Exercises	347
Index	381

Preface

This text is intended for either an applied algebra course or a modern algebra course that includes more applications than has been traditional. It is at an advanced undergraduate (junior-senior) level and is suitable for a one-semester or two-quarter course. We assume that students have already had a course in linear algebra (although we briefly review concepts from linear algebra when they are needed in the sections on fields and linear codes).

Our treatment is fairly rigorous, with almost every proof supplied. However, we have tried to concentrate on examples and applications. In an applied algebra course, which usually consists of majors in applied mathematics, computer science, and electrical engineering, the instructor will probably skip many of the proofs and concentrate on statements of results, examples, and algorithms. We also note that the last few exercises in most problem sets are usually of a more theoretical nature and can be omitted. There is too much material to cover in a single semester and the instructor will need to make some choices.

The instructor of a modern algebra course will likely want to include most proofs but will probably omit the material on switching functions and graph theory. The first five chapters provide ample material for a one-semester course in modern algebra, which would include many applications. We feel that present abstract algebra courses make algebra seem disjoint from the rest of the mathematics courses that a student takes. We have tried to exploit interactions between algebra and discrete mathematics and to include such contemporary applications as public-key cryptosystems and coding theory.

For an applied algebra course, one possible selection of material would be the first two chapters, III.1–III.9, a brief discussion of rings and fields (including finite fields), V.1–V.5, and Chapter VI. If there are few electrical engineering majors in the class, then the first half of Chapter IV could be covered in place of material on switching functions.

The first two chapters are rather brief. In Chapter I, we begin with a brief review of sets and simple counting arguments. Then we study the most important properties of the integers, including Induction, the Division and Euclidean Algorithms, and the Fundamental Theorem of Arithmetic. In the final section of the first chapter, we introduce finite-state machines (automata). The second chapter concentrates on relations. Throughout the text, we point out the importance and ubiquity of equivalence relations. The final section of Chapter II presents the application of equivalence relations to the problem of finding a minimal finite-state machine to perform a given task. Some of the proofs in this section are somewhat difficult for this point in the book, and the proofs, or the entire section, could certainly be omitted with no loss of continuity; but the main algorithm in this section is easy to understand, after doing a few examples, and this application helps students to understand the connection between equivalence relations and partitions.

In Chapter III, we study semigroups, monoids, and groups. We begin by studying the group of permutations of three elements (S_3), the group of symmetries of the square (D_4), and the semigroup of finite strings of elements from a finite alphabet. We hope that students will use these detailed examples to help them understand the abstract results that follow. We then present the main definitions and elementary results of the theory. In III.7, we show that semigroups and finite-state machines are very closely related. This section may be omitted, but we feel that it helps to make algebraic systems appear less abstract. In III.9 and III.10, we treat applications of groups to combinatorial problems by presenting Burnside's Theorem and Pólya Enumeration Theory. Section III.11 applies group actions to obtain theoretical results such as the class equation; this section may be omitted. We have left out some of the more difficult topics in abstract group theory, such as the Sylow Theorems. Some other difficult results, such as Cauchy's Theorem, appear as exercises at the end of some of the sections.

In Chapter IV, we study rings and fields. In IV.4, we present the Chinese Remainder Theorem, which has applications in computer arithmetic to so-called "fast adders." In IV.5, we study the application of the integers modulo n to cryptography. We then move on to polynomials and fields. More material on finite fields is presented than has been traditional in a modern algebra course. Again, we apply the theory to combinatorial problems by studying mutually orthogonal Latin squares and Steiner triple systems (a type of block design). Galois theory is treated only very briefly in a few exercises.

In Chapter V, we study algebraic coding theory. The first five sections of this chapter present basic topics such as linear codes, parity check matrices, error correction, coset decoding, and Hamming codes. These sections can be covered anytime after cosets have been studied, if the instructor provides a brief introduction to finite fields. The only finite fields that are used in the examples or exercises in these sections are \mathbf{Z}_2 and \mathbf{Z}_3 (although the theory is developed over any finite

field). In V.6, we treat the more advanced topic of cyclic codes. Here, one must have a good understanding of the material in IV.6–IV.8 on polynomials and the construction of finite fields.

Chapter VI presents Boolean algebras and their applications to the problem of finding a minimal logic circuit to perform a given task. The first two sections develop the theory and can prepare a student for more detailed study of lattices. The remaining sections of this chapter treat the Boolean algebra structure on equivalence classes of gating networks, and we present the Quine-McCluskey Algorithm for finding a minimal sum-of-products expression equivalent to a given Boolean expression (or network). This chapter may be covered anytime after Chapter II.

Since many applied algebra courses include material on graph theory, we have included a chapter that gives an introduction to the main concepts and algorithms of graph theory. Topics covered include graphs, directed graphs, trees, and networks. We present algorithms for finding the shortest distance between two vertices, a minimum spanning subtree, and a maximum flow in a network. This chapter may be covered anytime after Induction is studied.

We adhere to the following numbering conventions. Within a given chapter, we refer to results by section number and result number within the section; for example, if a reference to Theorem (3.6) occurs in Chapter IV, then it refers to a result in IV.3. Similarly, a reference to Exercise 2.5 within Chapter IV would refer to a problem at the end of IV.2. If we need to refer to a result from another chapter, then we will use the roman numeral of that chapter as well. So, for example, if we are in Chapter IV, then a reference to Example (III.7.4) would refer to an example in III.7 and a reference to Exercise (I.2.6) would refer to a problem at the end of I.2.

At the end of the text, we have included solutions or suggestions for selected exercises from each section. These problems are usually neither the easiest nor the hardest of the exercises. Most often, they are mid-range exercises that we feel will be assigned by most instructors and that should be understood by all students.

We would like to thank the reviewers and colleagues who made many suggestions that resulted in improvements in this book. Special thanks go to Bill Adkins, James Oxley, Neal Stoltzfus, Jorge Morales, and Michael Lacey. We also thank Peter Coveney of HarperCollins for his support.

Robert F. Lax
Baton Rouge

CHAPTER I

Preliminaries

In this chapter, we present some fundamental notions. The first section is a review of informal set theory and some basic counting arguments. In the second section, we study properties of the integers that are heavily used throughout the book. In particular, we consider Mathematical Induction, the Division and Euclidean Algorithms, and prime numbers. In the third section, we consider finite-state machines, which may be new to many readers. The basic concept of a finite-state machine is fairly simple. These machines can provide us with simple examples of many of the definitions in Chapters II and III. In Chapter III, we will see that finite-state machines are very closely related to algebraic objects known as semigroups.

1. Sets and Counting

We begin with the customary brief review of informal set theory. We consider a *set* as a collection of objects, which are called the *elements* of the set. If S is a set and x is an element of S , we write $x \in S$. We write $y \notin S$ to signify that y is not an element of S . We introduce the following special symbols for some sets that we will have occasion to use.

- \emptyset = the empty set
- \mathbf{B} = the set whose elements are 0 and 1
- \mathbf{P} = the set of all positive integers (natural numbers)
- \mathbf{Z} = the set of all integers
- \mathbf{Q} = the set of rational numbers
- \mathbf{R} = the set of real numbers
- \mathbf{R}_+ = the set of positive real numbers
- \mathbf{C} = the set of complex numbers

We will define new sets from old ones by using the “set builder” notation

$$\{x \in S : x \text{ satisfies a certain property}\},$$

read “the set of all x in S such that x satisfies a certain property.” For example, the even integers could be specified as $\{n \in \mathbf{Z} : n = 2m \text{ for some } m \in \mathbf{Z}\}$.

Two sets are called *equal* if they have exactly the same elements.

If S is a finite set, then we denote the number of elements in S , or *cardinality* of S , by $|S|$. In computer science applications, a finite set is often called an *alphabet* and its elements are called *letters*.

We say that S is a *subset* of T , and write $S \subseteq T$ if every element of S is an element of T . We leave it as an exercise to see that $S = T$ if and only if $S \subseteq T$ and $T \subseteq S$.

Given two sets S and T , we define their *union* $S \cup T$ by

$$S \cup T = \{x : x \in S \text{ or } x \in T \text{ (or both)}\}$$

and their *intersection* $S \cap T$ by

$$S \cap T = \{x : x \in S \text{ and } x \in T\}.$$

Similarly, if S_1, S_2, \dots, S_n are sets, then we define

$$\begin{aligned}\cup_{i=1}^n S_i &= \{x : x \in S_i \text{ for some } i = 1, 2, \dots, n\} \\ \cap_{i=1}^n S_i &= \{x : x \in S_i \text{ for all } i = 1, 2, \dots, n\}.\end{aligned}$$

A simple counting argument gives the following result.

(1.1) PROPOSITION. Suppose S and T are finite sets. Then

$$|S \cup T| = |S| + |T| - |S \cap T|.$$

PROOF. When we add $|S|$ and $|T|$, we count the elements in $S \cap T$ twice. By subtracting $|S \cap T|$, we eliminate this duplication. ■

To get into the spirit of writing proofs, we demonstrate the following proposition.

(1.2) PROPOSITION. $S \subseteq T$ if and only if $S \cap T = S$.

PROOF. There are two “directions” to show. First, suppose that $S \subseteq T$. To show that $S \cap T = S$, we will see that each set is a subset of the other. It is clear from the definitions that $S \cap T \subseteq S$. Now suppose $x \in S$. Since we are supposing $S \subseteq T$, it follows that $x \in T$. Hence $x \in S \cap T$ and we have shown that $S \subseteq S \cap T$. This completes one direction.

Conversely, suppose $S \cap T = S$. Then $S \subseteq T$, since it is clear from the definitions that $S \cap T \subseteq T$. ■

We define the *Cartesian product* $S \times T$ of two sets S and T to be the set of all ordered pairs in which the first member, or coordinate, of the pair comes from S and the second member, or coordinate, of the pair comes from T ; i.e.,

$$S \times T = \{(s, t) : s \in S, t \in T\}.$$

(Recall that the *ordered pairs* (a, b) and (b, a) are different, while the *sets* $\{a, b\}$ and $\{b, a\}$ are identical.) Similarly, given sets S_1, S_2, \dots, S_n , we define their Cartesian product $S_1 \times S_2 \times \dots \times S_n$ by

$$S_1 \times S_2 \times \dots \times S_n = \{(x_1, x_2, \dots, x_n) : x_i \in S_i \text{ for } i = 1, 2, \dots, n\},$$

where (x_1, x_2, \dots, x_n) denotes an ordered n -tuple. We put

$$S^n = \underbrace{S \times \dots \times S}_{n \text{ times}}.$$

Suppose $|S_i| = m_i$ for $i = 1, 2, \dots, n$. Then in forming an element of $S_1 \times S_2 \times \dots \times S_n$, we have m_i choices for the i^{th} coordinate, for $i = 1, 2, \dots, n$. By the so-called “product rule” for counting, we then have the following result.

(1.3) PROPOSITION. If $|S_i| = m_i$ for $i = 1, 2, \dots, n$, then

$$|S_1 \times S_2 \times \dots \times S_n| = m_1 m_2 \dots m_n.$$

Given a set S , the *power set* of S , denoted $\mathcal{P}(S)$, is the set of all subsets of S . In forming a subset of S , we can either include or not include each element of S ; that is, we have two choices corresponding to each element of S . This gives the following result.

(1.4) **PROPOSITION.** If $|S| = n$, then $|\mathcal{P}(S)| = 2^n$.

► (1.5) **EXAMPLE.** Suppose $S = \{0, 1, 2, 3\}$ and $T = \{0, 2, 4\}$. Then

$$\begin{aligned} B &\subseteq S \\ S \cup T &= \{0, 1, 2, 3, 4\} \\ S \cap T &= \{0, 2\} \\ S \times T &= \{(0, 0), (0, 2), (0, 4), (1, 0), (1, 2), (1, 4), \\ &\quad (2, 0), (2, 2), (2, 4), (3, 0), (3, 2), (3, 4)\} \\ \mathcal{P}(T) &= \{\emptyset, \{0\}, \{2\}, \{4\}, \{0, 2\}, \{0, 4\}, \{2, 4\}, \{0, 2, 4\}\}. \end{aligned}$$

Note that $5 = |S \cup T| = |S| + |T| - |S \cap T|$, that $12 = |S \times T| = |S| \cdot |T|$, and that $8 = |\mathcal{P}(T)| = 2^{|T|}$. ◀

Given a set S with n elements, how many different ordered r -tuples can we form using elements from S such that no two coordinates in any r -tuple are identical? That is, what is the cardinality of the subset P_r of S^r defined by

$$P_r = \{(s_1, s_2, \dots, s_r) : s_i \in S \text{ and } s_i \neq s_j \text{ if } i \neq j, \text{ for } i, j = 1, 2, \dots, r\}.$$

The set P_r is sometimes called the set of all r -permutations of the set S . There are n possibilities for the first coordinate of an r -tuple in P_r . Then there are $n - 1$ possibilities for the second coordinate, since the second coordinate must be different from the first. Continuing, we see that there are $n - k + 1$ possibilities for the k^{th} coordinate. The number of r -tuples in P_r is then

$$n(n-1) \cdots (n-r+1) = n!/(n-r)!.$$

(Recall that $n!$, read “ n factorial,” is defined by $n! = n(n-1)(n-2) \cdots 2 \cdot 1$. We also adopt the convention that $0! = 1$.)

Now consider the question of how many r -element (unordered) subsets there are of an n -element set S . That is, what is the cardinality of the subset C_r of $\mathcal{P}(S)$ consisting of all subsets of S that contain exactly r elements? C_r is sometimes called the set of all r -combinations of S . The cardinality of C_r is also the number of different ways we can choose r elements (without replacement) from S . We can obtain all r -permutations of S by considering all possible r -permutations of each of the r -element subsets in C_r . Notice that there are $r!$ r -permutations of an r -element set. Therefore, we have $n!/(n-r)! = r! \cdot |C_r|$. Consequently, we may state the following result.

(1.6) PROPOSITION. The number of ways to choose r elements (without replacement) from an n -element set is

$$\frac{n!}{r!(n-r)!}.$$

(1.7) DEFINITION. The integer $n!/r!(n-r)!$ is denoted by $\binom{n}{r}$ (read “ n choose r ”) and is called a *binomial coefficient*.

The reason that this integer is called a “binomial coefficient” is that the Binomial Theorem from elementary algebra states that if a and b are real numbers and if n is a positive integer, then

$$(a+b)^n = a^n + \binom{n}{1}a^{n-1}b + \cdots + \binom{n}{r}a^{n-r}b^r + \cdots + b^n.$$

► **(1.8) EXAMPLE.** Consider the set $S = \{1, 2, 3, 4\}$. The number of two-element subsets of S is $\binom{4}{2} = 4!/2!2! = (4 \cdot 3)/(2 \cdot 1) = 6$. The six 2-element subsets of S are

$$\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \text{ and } \{3, 4\}. \blacktriangleleft$$

Finally, we state another obvious, but useful, counting principle.

(1.9) THE PIGEONHOLE PRINCIPLE. Suppose n pigeons are assigned to m pigeonholes.

- 1) If $n > m$, then at least one pigeonhole contains two or more pigeons.
- 2) If $n < m$, then at least one pigeonhole contains no pigeons.
- 3) Suppose $n = m$. Then the following statements are equivalent.
 - a) Every pigeonhole contains a pigeon.
 - b) No pigeonhole contains more than one pigeon.
 - c) Every pigeonhole contains exactly one pigeon.

One could also think of this principle as referring to letters and mailboxes, but of course we will be interested in applying it to elements and sets.

► (1.10) EXAMPLES

1) If someone rolls a die seven times, then one of the numbers from the set $S = \{1, 2, 3, 4, 5, 6\}$ must occur (at least) twice. If the die were rolled only five times, then (at least) one of the numbers in S does not occur. If the die were rolled six times, then each number in S occurs if and only if no number in S occurs twice. Here, we can view the elements in S as the "pigeonholes" and the outcomes of the rolls of the die as the "pigeons."

2) Suppose at a party (with at least two people) that no two people shook each other's hands more than once. We claim that there were (at least) two people at the party who shook the same number of hands. Let n be the number of people at the party who shook at least one hand. If $n = 0$, then everyone shook the same number of hands (namely, none) and we are done. So we may assume that n is positive and since $n = 1$ is impossible (it takes two people to shake hands), we may assume $n \geq 2$. Let the pigeonholes be the integers 1 to $n - 1$ and assign each of the n people who shook at least one hand to the pigeonhole that corresponds to the number of hands they shook. Notice that since nobody shook his own hand and no two people shook each other's hands more than once, each of these n people shook at most $n - 1$ hands. Since there are more people than pigeonholes, two people must be assigned to the same pigeonhole; that is, two people shook the same number of hands. ◀

EXERCISES

* The symbol † means that a solution or a suggestion for that exercise appears at the back of the book.

† 1.1. Let $S = \{n \in \mathbf{P} : n^2 < 100\}$ and let $T = \{n \in \mathbf{P} : n = 3m \text{ for some } m \in S\}$.

a) Find $S \cap T$.

b) Find $|S \cup T|$ in two ways.

† 1.2. Show that $|\mathbf{B}^2| = |\mathcal{P}(\mathbf{B})|$ by finding the elements of these two sets.

† 1.3. Prove that $S = T$ if and only if $S \subseteq T$ and $T \subseteq S$.

† 1.4. Prove that $S \subseteq T$ if and only if $S \cup T = T$.

1.5. Prove that $(A \cap C) \cap (B \cap C) = A \cap B \cap C$.

† 1.6. Prove that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

1.7. Prove that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

- † 1.8. a) Prove that $(A \times B) \cap (A \times C) = A \times (B \cap C)$.
 b) Suppose A , B , and C are finite sets. Prove that $|(A \times B) \cap (A \times C)| = |A| \cdot |B \cap C|$.
- 1.9. a) Prove that $A \times (B \cup C) = (A \times B) \cup (A \times C)$.
 b) Suppose A , B , and C are finite sets. Then it follows from part (a) that

$$|A \times (B \cup C)| = |(A \times B) \cup (A \times C)|.$$

Give another proof of this result using Proposition (1.1) and Exercise 1.8(b).

- 1.10. In how many different ways can one choose three elements (without replacement) from a set with five elements?

- 1.11. Show that $\binom{n}{r} = \binom{n}{n-r}$.

- 1.12. Prove that

$$\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}.$$

- † 1.13. If A , B , and C are finite sets, prove that

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|.$$

- † 1.14. Prove that among any 13 people, at least two have their birthday in the same month.

- 1.15. Prove that if four distinct numbers are chosen from the set $\{1, 2, 3, 4, 5, 6\}$, then two of the four must have a sum of seven. (Hint: Take the pigeonholes to be the sets $\{1, 6\}$, $\{2, 5\}$, and $\{3, 4\}$.)

- † 1.16. Prove that if n pigeons are assigned to m pigeonholes, then one of the pigeonholes must contain at least $\lfloor (n-1)/m \rfloor + 1$ pigeons, where $\lfloor x \rfloor$ denotes the greatest integer in x (i.e., the integer s such that $s \leq x < s+1$).

- 1.17. Suppose that S is a set consisting of nine positive integers, each of which is less than 50. Show that there exists (at least) two different subsets of S such that the sum of the integers in one subset equals the sum of the integers in the other subset. (Hint: Use the Pigeonhole Principle with the subsets of S as the pigeons and the integers from 0 to 405 as the pigeonholes.)

2. The Integers

In this section, we collect some of the basic properties of the integers that will be used in later chapters. One of the fundamental properties (actually, the last of Peano's five axioms) of the natural numbers, or positive integers, is the familiar Principle of Mathematical Induction.

(2.1) PRINCIPLE OF MATHEMATICAL INDUCTION. Suppose S is a subset of \mathbf{P} such that the following two properties hold:

- (i) $1 \in S$
- (ii) For all $k \in \mathbf{P}$, if $k \in S$, then $k + 1 \in S$.

Then $S = \mathbf{P}$.

► (2.2) EXAMPLES

- 1) We will use induction to prove that the formula

$$1 + 2 + \cdots + n = n(n+1)/2$$

is valid for all positive integers n . The idea is to let S denote the set of all positive integers for which this formula is valid and then to show that $S = \mathbf{P}$; however, we do not usually explicitly introduce S , but rather proceed as follows.

First, we have $1 = 1(2)/2$, which establishes the formula for $n = 1$. (The $n = 1$ case is usually not difficult.) Now we *assume* the formula true for $n = k$ (the "induction hypothesis") and *prove* the formula true for $n = k + 1$. The idea is to utilize the induction hypothesis in proving the $n = k + 1$ case. We have

$$\begin{aligned} 1 + 2 + \cdots + k + (k + 1) &= [1 + 2 + \cdots + k] + (k + 1) \\ &= k(k + 1)/2 + (k + 1) \\ &= (k + 1)(k + 2)/2, \end{aligned}$$

where we used the induction hypothesis in the next-to-last step. Thus the formula is true for $n = k + 1$ and is therefore true for all n by the Principle of Induction.

- 2) Suppose A, B_1, B_2, \dots, B_n are sets. We use induction to prove that

$$A \cap (\cup_{i=1}^n B_i) = \cup_{i=1}^n (A \cap B_i).$$

The case $n = 1$ is trivial, since $A \cap B = A \cap B$. We now assume the statement true for $n = k$ and prove the statement true for $n = k + 1$. We have

$$\begin{aligned} A \cap (\cup_{i=1}^{k+1} B_i) &= A \cap ((B_1 \cup \cdots \cup B_k) \cup B_{k+1}) \\ &= (A \cap (B_1 \cup \cdots \cup B_k)) \cup (A \cap B_{k+1}) \quad \text{by Exercise 1.6} \\ &= ((A \cap B_1) \cup \cdots \cup (A \cap B_k)) \cup (A \cap B_{k+1}) \quad 18 \\ &= \cup_{i=1}^{k+1} (A \cap B_i), \end{aligned}$$

where we used the induction hypothesis in the next-to-last step. Note that the above argument does not prove that the validity of the statement for $n = 2$ follows from the fact that the statement is true when $n = 1$; we needed to establish the $n = 2$ case separately in Exercise 1.6. ◀

The Principle of Induction is the theoretical basis for recursive definitions, by which we mean the definition of a sequence of numbers by specifying the first term of the sequence and then specifying the k^{th} term of the sequence under the assumption that the $(k - 1)^{\text{st}}$ term is known.

We will also have occasion to use an alternative form of Induction. The statement below is equivalent to (2.1) in the sense that it can be shown that each statement implies the other.

(2.3) PRINCIPLE OF MATHEMATICAL INDUCTION (ALTERNATIVE FORM). Suppose S is a subset of \mathbf{P} such that the following two properties hold:

- (i) $1 \in S$
- (ii) For all $k \in \mathbf{P}$, if $1, 2, \dots, k - 1$ are all in S , then $k \in S$.

Then $S = \mathbf{P}$.

We can use Induction to derive another basic fact about the natural numbers that is intuitively clear.

(2.4) THE WELL-ORDERING PRINCIPLE. Every nonempty subset of \mathbf{P} has a least element.

PROOF. Suppose T is a nonempty subset of \mathbf{P} and put $S = \{n \in \mathbf{P} : n \notin T\}$. By way of contradiction, we assume that T has no least element. Then $1 \notin T$, since if it were in T , then it obviously would be the least element in T . Thus $1 \in S$. Now suppose that $1, 2, \dots, k - 1$ are all in S , and thus not in T . Then $k \notin T$, or else it would be the least element in T . Hence $k \in S$ and we can conclude by (2.3) that $S = \mathbf{P}$, which implies that T is empty. This contradiction shows that T must indeed have a least element. ■

(2.5) COROLLARY. Suppose $n_0 \in \mathbf{Z}$ and suppose T is a nonempty subset of $\{n \in \mathbf{Z} : n \geq n_0\}$. Then T has a least element.

PROOF. If $n_0 \in \mathbf{P}$, then the corollary follows immediately from the Well-Ordering Principle. If $n_0 \notin \mathbf{P}$, then consider $T \cap \{n_0, n_0 + 1, \dots, 0\}$. If this intersection is empty, then again T is a subset of \mathbf{P} and the corollary follows from (2.4). If this intersection is nonempty, then the least element in this intersection is the least element in T . (There is no problem in determining the least element in this intersection since it is a finite set.) ■

The next basic fact about the integers that we will consider is the Division Algorithm, which we first learn in about the third grade.

(2.6) THE DIVISION ALGORITHM. If $a \in \mathbf{Z}$ and $b \in \mathbf{P}$, then there exist unique integers q and r such that

$$a = qb + r \text{ with } 0 \leq r < b.$$

(Here q is the "quotient" and r is the "remainder.")

PROOF. We first establish the existence of q and r as above, and then we will show the required uniqueness. Put $T = \{a - nb : n \in \mathbf{Z} \text{ and } a - nb \geq 0\}$. We claim that T is nonempty. Indeed, if $a \geq 0$, then $a = a - 0b$ is in T , while if $a < 0$, then $-(b-1)a = a - ab$ is in T . Therefore, by Corollary (2.5), T has a least element, call it r , and suppose $r = a - qb$. By the definition of T , we have $r \geq 0$. We need to see that $r < b$. Suppose $r \geq b$. Then $r - b \geq 0$ and $r - b = a - (q+1)b$, so $r - b$ would be in T , contradicting the minimality of r .

Now for the uniqueness. In general, to prove uniqueness of something, we assume there exist two such objects and then show they must be equal. So suppose there exist q' and r' such that

$$a = q'b + r' \text{ with } 0 \leq r' < b.$$

Then $r - r' = (q' - q)b$; i.e., $r - r'$ is an integral multiple of b . But since $0 \leq r, r' < b$, we have $-b < r - r' < b$. Hence the only way $r - r'$ can be an integral multiple of b is for both $q' - q$ and $r - r'$ to be 0. Therefore, we have $q = q'$ and $r = r'$, establishing the required uniqueness. ■

(2.7) DEFINITION. Suppose a and b are integers with $b \neq 0$. We say that b divides a and write $b|a$ if there exists an integer c such that $bc = a$. If $b|a$, then we say that b is a *divisor* of a and a is a *multiple* of b .

So, for example, $2|6$ since $2 \cdot 3 = 6$. Note that if b is positive, then b divides a exactly when the remainder r in the Division Algorithm is 0. Also note that if $b|a$, then b divides any integral multiple of a ; indeed, if $bc = a$, then $b(cd) = ad$ for any $d \in \mathbf{Z}$.

(2.8) DEFINITION. Suppose $a, b \in \mathbf{Z}$, with not both a and b equal to 0. A positive integer d is called a *greatest common divisor (GCD)* of a and b if the following two properties are satisfied:

- (i) $d|a$ and $d|b$.
- (ii) If $c \in \mathbf{Z}$ and if $c|a$ and $c|b$, then $c|d$.

We could have defined the greatest common divisor of a and b to be the largest positive integer in the finite set of common divisors of a and b , but the above definition has the advantage that it will generalize to allow us to define the greatest common divisor of two polynomials. Unfortunately, it also has some disadvantages. Indeed, we now need to establish that a greatest common divisor of two integers exists — just giving a definition does not guarantee existence, and it is not obvious that the largest integer in the set of common divisors of a and b has the second property in Definition (2.8). We also want to know that two integers have a unique GCD according to our definition. Let's handle the uniqueness first.

(2.9) LEMMA. If a greatest common divisor of a and b exists, then it is unique.

PROOF. Suppose d and d' are greatest common divisors of a and b . Then from part (ii) of the definition of GCD we must have $d|d'$ and $d'|d$. We leave it as an exercise to show that this implies $d = \pm d'$. Since d and d' are both positive (from the definition of GCD), we can conclude that $d = d'$. ■

Now for the existence of GCDs. Instead of giving a theoretical existence proof, we will consider the Euclidean Algorithm, which actually produces the GCD of two positive integers. First, though, we prove the following lemma.

(2.10) LEMMA. Suppose $a, b \in \mathbf{Z}$ with $0 < b \leq a$. Apply the Division Algorithm to obtain q and r such that $a = qb + r$ with $0 \leq r < b$. Then the GCD of a and b equals the GCD of b and r . (Here we assume the existence of GCDs, which will be proved below.)

PROOF. Let d denote the GCD of a and b . Since $r = a - qb$ and since $d|a$ and $d|b$, we have that d divides r by Exercise 2.7. Hence d is a common divisor of b and r . Now suppose $d'|b$ and $d'|r$. Then $d'|qb$ and hence $d'|a$, again by Exercise 2.7. Hence $d'|d$ since d is the GCD of a and b . This proves that d is the GCD of b and r . ■

Now for the Euclidean Algorithm.

(2.11) EUCLIDEAN ALGORITHM FOR FINDING GCD. Suppose a and b are two integers with $0 < b \leq a$.

Step 1. Apply the Division Algorithm to obtain q_1 and r_1 such that

$$a = q_1b + r_1 \text{ with } 0 \leq r_1 < b.$$

If $r_1 = 0$, then b is the GCD of a and b .

Step 2. If $r_1 \neq 0$, then apply the Division Algorithm to obtain q_2 and r_2 such that

$$b = q_2 r_1 + r_2 \text{ with } 0 \leq r_2 < r_1.$$

If $r_2 = 0$, then the GCD of a and b is r_1 .

Step k . If $r_{k-1} \neq 0$, then divide r_{k-2} by r_{k-1} to obtain q_k and r_k such that

$$r_{k-2} = q_k r_{k-1} + r_k \text{ with } 0 \leq r_k < r_{k-1}.$$

Since $r_1 > r_2 > \cdots > r_k \geq 0$, we must eventually reach a remainder of 0 (in at most r_1 steps). If r_n is the first zero remainder, then the GCD of a and b is r_{n-1} (i.e., the last positive remainder).

PROOF. Suppose r_n is the first zero remainder in the above algorithm. From the definition of GCD, it is easy to see that the GCD of r_{n-1} and 0 is r_{n-1} . It then follows from Lemma (2.10) and the fact that $r_{n-2} = q_n r_{n-1}$ that r_{n-1} is the GCD of r_{n-2} and r_{n-1} . Since $r_{n-3} = q_{n-1} r_{n-2} + r_{n-1}$, another application of Lemma (2.10) shows that r_{n-1} is the GCD of r_{n-2} and r_{n-3} . Continuing to work our way back up the steps of the algorithm and applying Lemma (2.10) at each step, it follows that r_{n-1} is the GCD of a and b . ■

(2.12) NOTATION. Suppose $a, b \in \mathbf{Z}$, not both of which equal 0. It is customary to denote the GCD of a and b by (a, b) .

It is easy to see from the definition of GCD that $(a, b) = (-a, b) = (-a, -b)$. Also, if $b \neq 0$, then $(0, b) = |b|$. It then follows from (2.11) that we have established the existence of the GCD of any two integers such that not both are zero.

► **(2.13) EXAMPLE.** Let's find $(2244, 3276)$. The Euclidean Algorithm then becomes

$$3276 = 1 \cdot 2244 + 1032$$

$$2244 = 2 \cdot 1032 + 180$$

$$1032 = 5 \cdot 180 + 132$$

$$180 = 1 \cdot 132 + 48$$

$$132 = 2 \cdot 48 + 36$$

$$48 = 1 \cdot 36 + 12$$

$$36 = 3 \cdot 12 + 0$$

Therefore, $(2244, 3276) = 12$. ◀

(2.14) PROPOSITION. Suppose $a, b \in \mathbf{Z}$, not both of which equal 0. Then there exist integers s and t such that

$$(a, b) = sa + tb.$$

PROOF. Suppose $0 < b \leq a$. Then we can actually compute s and t from the Euclidean Algorithm. With notation as above, suppose $(a, b) = r_{n-1}$. Then the next to last equation in the algorithm may be written

$$r_{n-1} = r_{n-3} - q_{n-1}r_{n-2}. \quad (*)$$

Now, by the previous step in the algorithm, we can replace r_{n-2} in $(*)$ by $r_{n-4} - q_{n-2}r_{n-3}$. This results in an expression for r_{n-1} as the sum of an integral multiple of r_{n-3} and an integral multiple of r_{n-4} . Going up one more step in the algorithm, we can replace r_{n-3} by an expression involving r_{n-4} and r_{n-5} . Continuing to work up the steps of the algorithm in this manner, we eventually write r_{n-1} as the sum of an integral multiple of a and an integral multiple of b .

We leave the case when at least one of a or b is negative as an exercise at the end of this section (Exercise 2.11). ■

► **(2.15) EXAMPLE.** We will use (2.13) to find s and t such that $12 = s \cdot 2244 + t \cdot 3276$. Following the procedure described in the proof of (2.14), we obtain

$$\begin{aligned} 12 &= 48 - 1 \cdot 36 \\ &= 48 - (132 - 2 \cdot 48) = 3 \cdot 48 - 1 \cdot 132 \\ &= 3(180 - 1 \cdot 132) - 1 \cdot 132 = 3 \cdot 180 - 4 \cdot 132 \\ &= 3 \cdot 180 - 4(1032 - 5 \cdot 180) = 23 \cdot 180 - 4 \cdot 1032 \\ &= 23(2244 - 2 \cdot 1032) - 4 \cdot 132 = 23 \cdot 2244 - 50 \cdot 1032 \\ &= 23 \cdot 2244 - 50(3276 - 1 \cdot 2244) \\ &= 73 \cdot 2244 - 50 \cdot 3276. \blacktriangleleft \end{aligned}$$

(2.16) DEFINITION. An integer $p > 1$ is called *prime* if the only divisors of p are $\pm 1, \pm p$. Two integers a and b are called *relatively prime* if $(a, b) = 1$.

(2.17) LEMMA. Suppose p is a prime and $a \in \mathbf{Z}$. Then either $p|a$ or $(a, p) = 1$.

PROOF. Exercise 2.13.

(2.18) PROPOSITION. Suppose a, b , and c are integers such that $a|bc$ and $(a, b) = 1$. Then $a|c$.

PROOF. By Proposition (2.14), there exist integers s and t such that $1 = sa + tb$. Hence, we have $c = sac + tbc$. Since $a|sac$ and $a|tbc$ (since $a|bc$), we know that $a|c$ by Exercise 2.7. ■

(2.19) COROLLARY. If p is a prime and if $p|ab$, then either $p|a$ or $p|b$.

PROOF. Suppose p does not divide a . Then by Lemma (2.17), we have $(a, p) = 1$. It then follows from Proposition (2.18) that $p|b$. ■

(2.20) COROLLARY. Suppose p is a prime and $p|a_1a_2 \cdots a_n$. Then $p|a_i$ for some $i = 1, 2, \dots, n$.

PROOF. Exercise 2.14.

We now come to the Fundamental Theorem of Arithmetic, which says that any integer greater than 1 has a unique prime factorization.

(2.21) THE FUNDAMENTAL THEOREM OF ARITHMETIC. Let n be any integer greater than 1.

- 1) There exist primes $p_1 < p_2 < \cdots < p_r$ such that

$$n = p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r}$$

for some positive integers s_1, s_2, \dots, s_r .

- 2) The factorization in (1) is unique in the sense that if we also have

$$n = q_1^{t_1} q_2^{t_2} \cdots q_l^{t_l}$$

for primes $q_1 < q_2 < \cdots < q_l$ and positive integers t_1, t_2, \dots, t_l , then $r = l$ and for each $i = 1, \dots, r$, we have $p_i = q_i$ and $s_i = t_i$.

PROOF. We first show the existence of a prime factorization for every integer greater than 1 by using Induction (in the form (2.3), but starting the induction with $n = 2$). Since 2 is prime, such a factorization exists for $n = 2$.

Now we assume that every integer m with $2 \leq m < k$ can be written as a product of primes. Under this hypothesis, we must show that k can be written as a product of primes. If k is itself prime, then we are done. If k is not prime, then we can write $k = m_1 m_2$ with $2 \leq m_1, m_2 < k$. By our induction hypothesis, each of m_1 and m_2 can be written as a product of primes. Thus k is the product of all the primes appearing in these two factorizations. This completes the existence portion of the proof.

We use a similar inductive argument for the uniqueness part of the proof. Clearly, the only way to write 2 as a product of primes is $2 = 2$. Now assume that every integer m with $2 \leq m < k$ has a unique prime factorization as specified in the theorem. We will suppose we have two prime factorizations for k and then show that these factorizations are identical. So suppose

$$k = p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r} = q_1^{t_1} q_2^{t_2} \cdots q_l^{t_l}, \quad (*)$$

where $p_1 < p_2 < \cdots < p_r$ and $q_1 < q_2 < \cdots < q_l$ are primes. Since $p_1 | k$, we have that $p_1 | q_1^{t_1} \cdots q_l^{t_l}$. It follows then from Corollary (2.20) that $p_1 | q_j$ for some j . Since q_j is prime, this implies that $p_1 = q_j$. Note that $q_1 \leq q_j = p_1$. The same reasoning shows that q_1 must equal one of the p_i and hence that $p_1 \leq p_i = q_1$. It follows then that $p_1 = q_1$. We claim that s_1 must then equal t_1 . Indeed, if $s_1 < t_1$, then by dividing each side of (*) by $p_1^{s_1}$, we would have that $p_1 | p_2^{s_2} \cdots p_r^{s_r}$. But this would imply that $p_1 = p_i$ for some $i > 1$, which is a contradiction. Similarly, we could not have $s_1 > t_1$, so we can conclude that $s_1 = t_1$. But then we have

$$u = \frac{k}{p_1^{s_1}} = p_2^{s_2} \cdots p_r^{s_r} = q_2^{t_1} \cdots q_l^{t_l}.$$

If $u = 1$, then $k = p_1^{s_1}$ is the unique prime factorization of k . If $u > 1$, then, since $u < k$, we may apply our induction hypothesis to conclude that $r = l$, $p_i = q_i$ and $s_i = t_i$ for $i = 2, \dots, r$. This completes the induction and the proof. ■

We note that there is no really effective way to find the prime factorization of a very large integer. In fact, this difficulty is the basis for very good encryption schemes ("unbreakable" codes). We will discuss this topic in IV.5.

Our final result in this section shows how to find the GCD of two positive integers if we know their prime factorizations. Again, since prime factorizations are difficult to find, this result does not really give an effective way to find GCDs and so the Euclidean Algorithm should usually be used.

(2.22) PROPOSITION. Suppose p_1, p_2, \dots, p_r are all the primes occurring in the prime factorizations of two positive integers a and b . Suppose

$$\begin{aligned} a &= p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r} \\ b &= p_1^{t_1} p_2^{t_2} \cdots p_r^{t_r} \end{aligned}$$

where $s_i, t_i \geq 0$ for $i = 1, \dots, r$. Then

$$(a, b) = p_1^{\min\{s_1, t_1\}} p_2^{\min\{s_2, t_2\}} \cdots p_r^{\min\{s_r, t_r\}}.$$

PROOF. Exercise 2.15.

- **(2.23) EXAMPLE.** By dividing 2244 and 3276 by small primes, one sees that the prime factorizations of these two integers are

$$2244 = 2^2 \cdot 3 \cdot 11 \cdot 17 \quad \text{and}$$

$$3276 = 2^2 \cdot 3^2 \cdot 7 \cdot 13.$$

Thus the GCD of 2244 and 3276 is $2^2 \cdot 3 = 12$, as was shown in Example (2.13). ◀

EXERCISES

- † 2.1. Prove that the formula

$$1^3 + 2^3 + \cdots + n^3 = [n(n+1)/2]^2$$

is valid for all $n \in \mathbf{P}$.

- 2.2. Prove that the formula

$$2 + 4 + \cdots + 2n = n(n+1)$$

is valid for all $n \in \mathbf{P}$.

- 2.3. Suppose A, B_1, B_2, \dots, B_n are sets. Prove that

$$A \cup (\cap_{i=1}^n B_i) = \cap_{i=1}^n (A \cup B_i).$$

- † 2.4. The Fibonacci sequence $\{F_n\} = \{1, 1, 2, 3, 5, 8, 13, 21, \dots\}$ is defined recursively by $F_1 = 1, F_2 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 3$.

- a) Prove that $(F_{n+1})^2 - F_n F_{n+2} = (-1)^n$ for all positive integers n .
b) Prove that F_{3n-2} and F_{3n-1} are odd and F_{3n} is even for all positive integers n .

- 2.5. Prove that $6 \mid n^3 - n$ for all positive integers n .

- † 2.6. Find the quotient q and the remainder r as in the Division Algorithm if:

- a) $a = 37, b = 5$
b) $a = -37, b = 5$ (Remember that r must be ≥ 0 .)
c) $a = 42, b = 7$

d) $a = 0, b = 5$

† 2.7. Suppose $c = a + b$ and suppose $d|a$ and $d|b$. Prove that $d|c$.

2.8. Show that if $a|b$ and $b|a$, then either $a = b$ or $a = -b$.

† 2.9. If $\{F_n\}$ is the Fibonacci sequence of Exercise 2.4, then prove that $5|F_{5n}$ for all positive integers n .

2.10. Suppose a, b, s, t are integers and $1 = sa + tb$. Prove that $(a, b) = 1$.

2.11. Prove Proposition (2.14) in the case when at least one of a or b is negative. (Hint: First apply the proposition to find s and t such that $(a, b) = s|a| + t|b|$.)

† 2.12. In each part, find the GCD of a and b and find integers s and t such that $(a, b) = sa + tb$.

a) $a = 12, b = 138$

b) $a = -12, b = 138$

c) $a = 210, b = 130$

d) $a = 17, b = 64$

e) $a = 1211, b = 6203$

† 2.13. Prove Lemma (2.17).

2.14. Use Mathematical Induction and Corollary (2.19) to prove Corollary (2.20).

† 2.15. Prove Proposition (2.22).

2.16. In each part, find the prime factorization of a and b and use these factorizations to find (a, b) .

a) $a = 42, b = 28$

b) $a = 360, b = 84$

c) $a = 561, b = 2093$

† 2.17. Suppose m_1, m_2, \dots, m_k are integers that are pairwise relatively prime; i.e., $(m_i, m_j) = 1$ if $i \neq j$ and $1 \leq i, j \leq k$. Prove that $m_1 m_2 \cdots m_k | n$ if and only if $m_i | n$ for all $i = 1, 2, \dots, k$.

† 2.18. Suppose $a, b \in \mathbf{Z}$. One defines the *least common multiple* of a and b , denoted $[a, b]$, as follows. If $a = 0$ or $b = 0$, then $[a, b] = 0$. If a and b are both nonzero, then a positive integer e is called a least common multiple of a and b if (1) $a|e$ and $b|e$ and (2) whenever $a|n$ and $b|n$, then $e|n$.

- a) Prove that a and b can only have one least common multiple.
 b) Prove that $[a, b]$ is the smallest positive integer that is a multiple of both a and b .
 c) Suppose a and b are positive and

$$\begin{aligned} a &= p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r} \\ b &= p_1^{t_1} p_2^{t_2} \cdots p_r^{t_r} \end{aligned}$$

where the p_i are all the primes occurring in the prime factorizations of a and b and $s_i, t_i \geq 0$ for $i = 1, \dots, r$. Prove that

$$[a, b] = p_1^{\max\{s_1, t_1\}} p_2^{\max\{s_2, t_2\}} \cdots p_r^{\max\{s_r, t_r\}}.$$

- d) Suppose a and b are positive. Prove that $ab = (a, b)[a, b]$.

2.19. Find $[a, b]$ in each part of Exercise 2.16.

2.20. Use Mathematical Induction, Proposition (1.2), and Example (2.2.2) to establish the “Principle of Inclusion-Exclusion”: Suppose A_1, A_2, \dots, A_n are finite sets. Then

$$\begin{aligned} \left| \bigcup_{i=1}^n A_i \right| &= \sum_{i=1}^n |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| \\ &\quad - \cdots + (-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap A_n|. \end{aligned}$$

- 2.21.** Prove that there are an infinite number of primes in \mathbf{P} . (Hint: Suppose that the only primes were p_1, p_2, \dots, p_N . Consider the integer $p_1 p_2 \cdots p_N + 1$.)
- 2.22.** a) Prove that (2.1) implies (2.3).
 b) Prove that (2.3) implies (2.1).
- 2.23.** Prove that (2.4) implies (2.1). (Hint: Let $T = \{n \in \mathbf{P} : n \notin S\}$ and apply (2.4) if T is nonempty.)

3. Finite-State Machines

In this section we present the definition and examples of finite-state machines. We will see in Chapter III that these machines are very closely related to algebraic objects known as semigroups.

We first need to review the notion of “function,” which we do in the following informal definition.

(3.1) DEFINITIONS. A *function* or *mapping* $f : A \rightarrow B$ from a set A to a set B is a rule that assigns to each element $a \in A$ precisely one element $f(a) \in B$, called the *image* of a under f . The set A is called the *domain* of the function f . The set B is sometimes called the *codomain* or *target* of f . The *range* of f is $\{b \in B : b = f(a) \text{ for some } a \in A\}$.

We will usually use the word “function” if the elements of the codomain are numbers; otherwise, we will usually use “mapping.” We called this definition “informal” because the term “rule” has not been precisely defined. We will give a rigorous definition of “function” in the next chapter.

A finite-state machine is an object that accepts an input and gives an output. In processing the input and producing the output, the internal condition, or state, of the machine may change. A computer, of course, is an example of a finite-state machine. For a more elementary example, we will consider a simple type of vending machine.

- **(3.2) EXAMPLE.** Consider a vending machine that sells packs of gum that cost 25¢ each. Suppose that the machine accepts nickels, dimes, and quarters as inputs. This machine can be constructed with five states, depending on whether the amount that has been input so far is 0, 5, 10, 15, or 20¢. For example, if a person begins by putting a dime in the machine, then the machine will shift into the state corresponding to 10¢. If another dime is inserted, then the machine will shift from the 10¢ state to the 20¢ state. If a nickel is now inserted, then the machine will output a pack of gum and will shift back to the 0¢ state. We can represent this machine pictorially as in Figure 3.1. In this figure, the arrows show the possible shifts of states and each arrow is labeled according to the amount of money input and also either with a 0, meaning no gum should be dispensed, or with a 1, meaning that (at least) 25¢ has been received and a pack of gum may be dispensed. Of course, this is a very simple example; we have not concerned ourselves with the coin return, or with the problem of giving change if too much money is input, or with the actual output routine. ◀

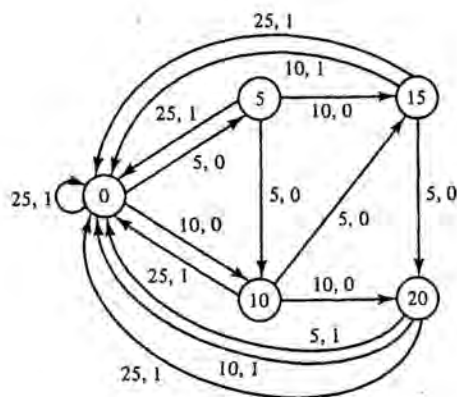


Figure 3.1

We can formalize the notion of a finite-state machine with the following definition.

(3.3) DEFINITION. A *finite-state machine* $M = (S, X, Z, \tau, \omega)$ is an ordered 5-tuple consisting of:

- 1) a finite set S , called the *state set*
- 2) a finite set X , called the *input alphabet*
- 3) a finite set Z , called the *output alphabet*
- 4) a mapping $\tau : S \times X \rightarrow S$, called the *next state* or *transition mapping*
- 5) a function $\omega : S \times X \rightarrow Z$, called the *output function*

A finite-state machine is also called a *finite automaton*.

We will assume that the input alphabet X and the output alphabet Z are both the set $\{0, 1\}$ unless we state otherwise. In our vending machine example above, $X = \{5, 10, 25\}$ and $Z = \{0, 1\}$. We will also assume throughout that a machine is in the lowest numbered state (usually σ_0 , but σ_1 if there is no σ_0) when the first input occurs.

A finite-state machine as defined above is sometimes also called a *Mealy machine*. At times we will be interested only in the state transitions within a machine, and we will forget about the outputs; in these circumstances, we will refer to the triple (S, X, τ) as a machine. (Such a machine is sometimes called a *semiautomaton*.) There is another type of machine, called a *Moore machine*, which is slightly different from a Mealy machine. In a Moore machine, the output function has S as its domain; that is, the output mapping looks like $\omega : S \rightarrow Z$, and thus the output depends only on the state, not on the input. Moore machines will play the principal role in II.4. ³⁰

(3.4) DEFINITIONS. Suppose X is an input alphabet and Z is an output alphabet. We let X^+ denote the set of all possible finite input strings $\hat{x} = x_1x_2 \cdots x_n$, where $n \in \mathbf{P}$ and $x_i \in X$ for $i = 1, 2, \dots, n$. Similarly, Z^+ will denote the set of all finite output strings. Let $\epsilon = "$ " denote the "empty" string. Put $X^* = X^+ \cup \{\epsilon\}$ and $Z^* = Z^+ \cup \{\epsilon\}$.

We can extend the output function $\omega : S \times X \rightarrow Z$ of a Mealy machine to a mapping $\omega^* : S \times X^* \rightarrow Z^*$ as follows. Put $\omega^*(\sigma, \epsilon) = \epsilon$ for all $\sigma \in S$. If $\hat{x} = x_1x_2 \cdots x_n \in X^+$ and $\sigma_1 \in S$, then we put

$$\omega^*(\sigma_1, \hat{x}) = \omega(\sigma_1, x_1)\omega(\sigma_2, x_2) \cdots \omega(\sigma_n, x_n),$$

where $\sigma_{i+1} = \tau(\sigma_i, x_i)$, for $i = 1, 2, \dots, n-1$. (When we write an input string $x_1x_2 \cdots x_n$, we mean that the first input is x_1 , the next input is x_2 , and so on. Similarly, an output string $z_1z_2 \cdots z_n$ means that z_1 is the first output and z_n is the last output.)

Similarly, if $\omega : S \rightarrow Z$ is the output function of a Moore machine, then we extend this to a mapping $\omega^* : S \times X^* \rightarrow Z^+$ as follows. Put $\omega^*(\sigma, \epsilon) = \omega(\sigma)$ for all $\sigma \in S$. If $\hat{x} = x_1x_2 \cdots x_n \in X^+$ and $\sigma_1 \in S$, then we put

$$\omega^*(\sigma_1, \hat{x}) = \omega(\sigma_2)\omega(\sigma_3) \cdots \omega(\sigma_{n+1}),$$

where $\sigma_{i+1} = \tau(\sigma_i, x_i)$ for $i = 1, 2, \dots, n$.

There are two standard ways to represent finite-state machines visually. One way is by a defining table that has one row for each state and whose columns describe the next state and, in the case of a Mealy machine, the output corresponding to a given input. In the case of a Moore machine, the output column in the defining table lists the output associated to the given state.

The other way is by a directed graph* (as in Figure 3.1) whose vertices correspond to the states and whose arcs (arrows) connect a state to the next state and are labeled according to the input and the output (in the case of a Mealy machine). We will call such a directed graph the *digraph of the machine*. When we give a digraph for a Moore machine, we either designate each vertex with the output associated to that state, or we specify the output associated to a state σ_i below a horizontal line within the circle representing that state; for example,



* Terms from graph theory are defined in Chapter VII.

means that the output 0 is associated to the state σ_0 . In the digraph of a Moore machine, an arc from σ_i to σ_j is labeled with the input that takes the machine from σ_i to σ_j .

When we say “construct a machine,” we will mean to give either the defining table or the digraph of the machine. We’ll illustrate these concepts with several basic examples.

► (3.5) **EXAMPLE:** Parity Check Machine.

Suppose a sequence of binary digits (i.e., 0’s and 1’s) is sent from one device to another device (say from one computer to another or from a disk drive to a computer). One calls a string of binary digits *even* if it contains an even number of 1’s and *odd* if it contains an odd number of 1’s. Two strings of binary digits are said to have the same *parity* if they are either both odd or both even. If the string that is received and the string that was sent do not have the same parity, then there was definitely an error in the transmission. This motivates a very simple machine called the parity check machine. The task of this machine is to receive a string of binary digits and to output a 0 if the string is even and a 1 if the string is odd. For example, if the input string 1011101 is fed into the machine (remember, start at the left), then the output string will be 1101001. This machine requires two states, which we will denote by σ_0 and σ_1 . The digraph of this machine is shown in Figure 3.2.

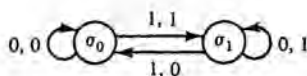


Figure 3.2

Here the machine is in the state σ_0 if the string received so far is even and in the state σ_1 if the string received so far is odd. If the next digit received is a 0, then the parity is unchanged, and the machine continues in its present state and repeats the present output. If the next digit received is a 1, then the parity changes, so the machine changes state and output.

We can also represent the parity check machine by the following table in which the first column lists the present state, the second and third columns list the “next state” (i.e., $\tau(\sigma_i, x_j)$) corresponding to a given input and the fourth and fifth columns list the output (i.e., $\omega(\sigma_i, x_j)$) corresponding to a given input. So, for example, if the present state is σ_0 and the machine receives the input 1, then the machine shifts into σ_1 and gives an output of 1.

(3.6) TABLE

Present State	Next State		Output	
	0	1	0	1
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_0	1	0

It is easy to think of the parity check machine as a Moore machine. The output function simply takes σ_0 to 0 and σ_1 to 1. We could then represent the machine by the digraph shown in Figure 3.3, in which we have named each state with the associated output.

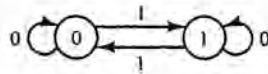


Figure 3.3

The defining table for this Moore machine would be

(3.7) TABLE

Present State	Next State		Output
	0	1	
0	0	1	0
1	1	0	1

Here, if the machine is in state 0 and an input of 1 is received, then the machine shifts into state 1 and the machine gives the output, namely 1, that is associated to the state into which the machine has shifted. ◀

▶ (3.8) EXAMPLE: The Mod 3 Binary Counter.

The next machine we consider is very similar to the parity check machine. This machine counts the 1's in the input sequence and outputs as follows. Let n denote the number of 1's received and apply the division algorithm to write $n = 3q + r$ with $r = 0, 1$, or 2 . The output of the machine is then the binary representation of r ; i.e., 00 if $r = 0$, 01 if $r = 1$, and 10 if $r = 2$ (so that the output alphabet for this machine is $Z = \{00, 01, 10\}$). The digraph of this machine is shown in Figure 3.4. If we view this machine as a Moore machine, we would draw the digraph shown in Figure 3.5. ◀

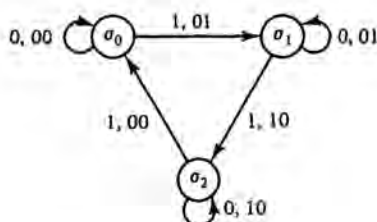


Figure 3.4

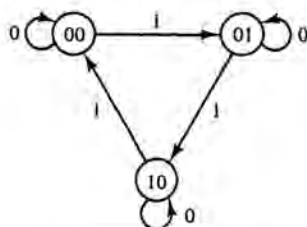


Figure 3.5

► (3.9) **EXAMPLE:** A Sequence Recognizer.

An important type of finite-state machine is a sequence recognizer. This machine outputs a 1 precisely when a certain designated sequence has been input. As an example, we will construct a machine that outputs a 1 precisely when the input string ends in 010. We will say that such a machine “recognizes” the string 010.

In constructing such a machine, we need to specify whether or not we allow overlapping. (For sequences such as 011, for example, the question of overlapping does not arise.) In the case when overlapping is allowed, we would allow the last zero of a triple 010 to count as the first zero of another such triple. Thus if the input string is 0101010, then the output string should be 0010101. The digraph of a Mealy machine constructed to perform this task is shown in Figure 3.6.

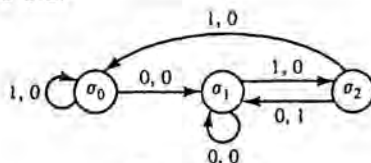


Figure 3.6

Here, the machine stays in σ_0 until a 0 is input, in which case it shifts into σ_1 . If a 0 is received while the machine is in σ_1 , it stays in that state, while if a 1 is received (meaning that the sequence of the last two inputs is 01), then the machine shifts into σ_2 . If the machine is in σ_2 and a 0 is input, then the triple 010 has been received and the machine outputs a 1 and shifts

back to σ_1 (to await a 1). If a 1 is input while the machine is in σ_2 , then the sequence of the past three outputs is 011, and the machine shifts to σ_0 to await a 0.

If we view this machine as a Moore machine, then the digraph of the machine would be the somewhat different digraph shown in Figure 3.7. Note that one more state would be needed.

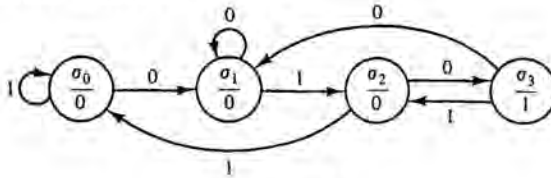


Figure 3.7

Now suppose that we do not allow overlapping. This would mean that we want to recognize “disjoint” occurrences of our sequence 010. Thus if the input string were 0101010, then the output string should be 0010001. To get a digraph for this machine, one would just need to change one of the arcs in each of the machines above so that after receipt of the sequence 010 (represented by the last state in the machine) one shifted back to the first state (“looking for a 0 to begin the sequence”) rather than the second state (“looking for a 1 to follow a 0”).

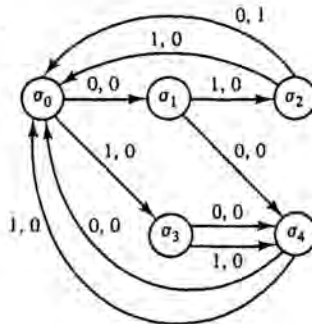


Figure 3.8

There is yet another type of machine we might want to construct in connection with the sequence 010. We could want to output a 1 precisely when a string of length $3n$ for some $n \in \mathbf{P}$ has been received that ends in 010. This means that we are looking at each sequence of three inputs and outputting a 1 only when such a sequence of three is 010. In this case, if, for example, the input string were 0101010, then the output string would be 0010000. The digraph of a Mealy machine to perform this task is shown in Figure 3.8. Here, the machine must reset to the initial state after every

third input. The digraph of a Moore machine to perform this task is shown in Figure 3.9. ◀

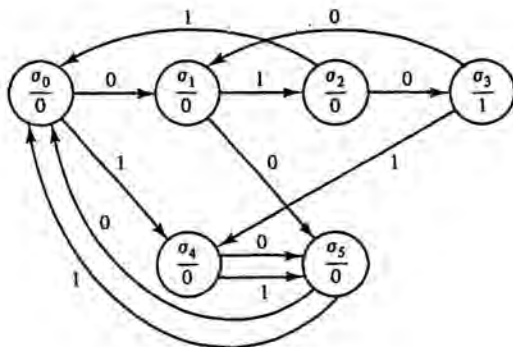
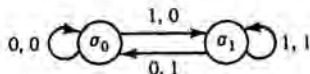


Figure 3.9

EXERCISES

† 3.1. The machine



is called a unit delay machine. Assume the machine starts in σ_0 .

- What is the output sequence corresponding to the input sequence 010101?
 - What is the output sequence corresponding to the input sequence 110110?
 - Describe the function of this machine.
- † 3.2. Construct a Moore machine that outputs 1 whenever the number of 1's received in the input string is a positive integral multiple of 3 and outputs 0 otherwise. For example, if the input string were 010101111, then the output should be 000001001.
- † 3.3. Construct a Moore machine with output as follows. Let n denote the number of 1's that has been received and write $n = 4q + r$ with $0 \leq r < 4$. Then the output of the machine should be the binary representation of r . (So this machine is a mod 4 binary counter.)
- † 3.4. Construct Mealy and Moore machines that recognize the triple 101 if:
- overlapping is allowed

b) overlapping is not allowed

- † 3.5. Construct Mealy and Moore machines that output 1 precisely when a sequence of length $3n$, for some $n \in \mathbf{P}$, ending in 101 has been received. For example, if the input string were 001101010101, then the output should be 000001000001.
- † 3.6. Construct a Mealy machine that recognizes the string 1001 if:
- a) overlapping is allowed
 - b) overlapping is not allowed
- † 3.7. Construct a Mealy machine that outputs 1 precisely when a string of length $4n$, for some $n \in \mathbf{P}$, ending in 1001 has been received.
- † 3.8. Construct Mealy and Moore machines that recognize the string 011. (So, for example, if the input string were 10111011, then the output should be 00010001.)
- † 3.9. Construct a Mealy machine that outputs 1 if the number of 1's is greater than the number of 0's in the preceding three inputs and outputs 0 otherwise. (If the first two inputs are 1 and 1, then the first two outputs should be 0 and 1; otherwise the first two outputs should both be 0.) For example, if the input string were 1100110, then the output string should be 0110011.

CHAPTER II

Relations

A relation is a very general object. Examples of relations include equality, set inclusion, partial orders, functions, and operations. In this chapter, we study relations and special properties that a relation may satisfy. The second section is devoted to the study of relations on finite sets. In this case, a relation may be represented by a matrix. In the third section, we consider the very important type of relation known as an equivalence relation. Equivalence relations play a central role in the remainder of this book and are very important throughout all of mathematics. Our hope in this chapter is that the reader will become familiar with relations in general and will gain a good comprehension of equivalence relations in particular. In the final section, we consider an application of equivalence relations to the problem of finding a minimal finite-state machine to perform a given job. Some of the proofs in the last section are a little complicated, but the reader can certainly apply the algorithm to some simple examples to gain an understanding of the main results and, hopefully, a better understanding of equivalence relations and partitions.

1. Relations and Mappings

A relation is one of the most general concepts in mathematics.

(1.1) DEFINITION. A *relation* from a set A to a set B is a subset R of $A \times B$. If $(a, b) \in R$, then we say that a is related to b by R , and we also write aRb . A relation from A to A is called a *relation on A* .

► (1.2) EXAMPLES

1) Most often we do not give a relation by specifying a collection of ordered pairs; rather, we give a “rule” describing when two elements are related. For example, we define the order relation $<$ on \mathbf{R} by $x < y$ if $y - x \in \mathbf{R}_+$. More formally, we have $<$ is the subset $\{(x, y) : y - x \in \mathbf{R}_+\}$ of $\mathbf{R} \times \mathbf{R}$.

2) On \mathbf{Z} , we have the “divides” relation; namely, $b|a$ if $a = bc$ for some $c \in \mathbf{Z}$.

3) On any set A , we may define the relation “=” to be the subset $\{(a, a) : a \in A\}$ of $A \times A$.

4) A function, such as $f(x) = \lfloor x \rfloor$ (the greatest integer function), may be thought of as a relation. The greatest integer function is a relation from \mathbf{R} to \mathbf{Z} and consists of the subset $\{(x, \lfloor x \rfloor) : x \in \mathbf{R}\}$ of $\mathbf{R} \times \mathbf{Z}$.

5) Suppose A is a set. We can define the usual inclusion relation on subsets of A by: if S and T are subsets of A , then $S \subseteq T$ if every element of S is an element of T . This defines a relation on $\mathcal{P}(A)$, the power set of A .

6) On a finite set A , we can specify a relation R by explicitly giving a subset of $A \times A$. For example, if $A = \{0, 1, 2\}$, then one relation on A would be

$$R = \{(0, 1), (1, 0), (0, 2), (1, 2)\}.$$

7) Suppose $M = (S, X, Z, \tau, \omega)$ is a finite-state machine. We can define a “connection relation” C on S by $\sigma_i C \sigma_j$ if there exists a (finite) sequence of inputs that carries the machine from state σ_i to state σ_j . We can define a “simple connection” relation C_1 on S by $\sigma_i C_1 \sigma_j$ if there exists $x \in X$ such that $\tau(\sigma_i, x) = \sigma_j$; that is, $\sigma_i C_1 \sigma_j$ if there is a single input that carries the machine from state σ_i to state σ_j . ◀

The reader is no doubt familiar with the notion of composition of functions. This concept may be easily generalized to define composition of relations.

(1.3) DEFINITION. If R_1 is a relation from A to B and R_2 is a relation from B to C , then the *composition of R_2 with R_1* , denoted $R_2 \circ R_1$, is defined by $R_2 \circ R_1 = \{(a, c) \in A \times C : \text{there exists } b \in B \text{ such that } (a, b) \in R_1 \text{ and } (b, c) \in R_2\}$.

We give names to special properties that certain relations possess.

(1.4) DEFINITIONS. Let R denote a relation on a set A . Then R is called:

- (i) *reflexive* if aRa for all $a \in A$.
- (ii) *symmetric* if whenever aRb , then bRa .

- (iii) *antisymmetric* if whenever aRb and bRa , then $a = b$.
 (iv) *transitive* if whenever aRb and bRc , then aRc .

► (1.5) EXAMPLES

1) The order relation $<$ on \mathbf{R} is transitive but is neither reflexive nor symmetric. (It is actually antisymmetric for a trivial reason — it is never the case that $a < b$ and $b < a$.)

2) One defines \leq on \mathbf{R} by $x \leq y$ if $x < y$ or $x = y$. This relation is reflexive, antisymmetric, and transitive. Such a relation is called a *partial order*. A set together with a partial order on the set is called a partially ordered set, or *poset*.

3) The “divides” relation on \mathbf{P} gives another example of a partial order. Note that this relation is not antisymmetric on \mathbf{Z} , since if $a|b$ and $b|a$, we can conclude only that $a = \pm b$.

4) It is not difficult to construct finite-state machines such that the simple connection relation on the set of states has certain desired properties. For example, the simple connection relation on the Moore machine shown in Figure 1.1 is symmetric, but neither reflexive nor transitive (since $\sigma_0 C_1 \sigma_1$ and $\sigma_1 C_1 \sigma_0$, but σ_0 is not related to σ_0). ◀

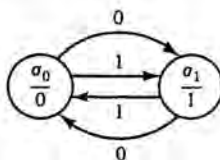


Figure 1.1

In II.3, we will study a very important special class of relations called equivalence relations. These relations are reflexive, symmetric, and transitive.

A function or mapping is a special kind of relation. In I.3, we gave an informal definition of a mapping. A precise definition is:

(1.6) DEFINITION. A *mapping*, or *function*, $f : A \rightarrow B$ from a set A to a set B is a relation R from A to B such that for every $a \in A$ there is precisely one element $f(a) \in B$ such that $(a, f(a)) \in R$.

Two important properties that a mapping may have are the properties of being one-to-one or onto.

(1.7) DEFINITIONS. Suppose $f : A \rightarrow B$ is a mapping. We say that f is *one-to-one* (1-1), or *injective*, if whenever $f(a_1) = f(a_2)$, then $a_1 = a_2$.

(Equivalently, f is 1-1 if $a_1 \neq a_2$ implies $f(a_1) \neq f(a_2)$.) We say that f is *onto*, or *surjective*, if for each $b \in B$ there exists $a \in A$ such that $f(a) = b$. If f is both 1-1 and onto, then f is called a *bijection* or a *1-1 correspondence*.

► (1.8) EXAMPLES

1) Consider the mapping $f : \mathbf{Z} \rightarrow \mathbf{Z}$ defined by $f(n) = 2n$. We will show that f is 1-1. Suppose $f(a) = f(b)$. This means that $2a = 2b$, hence we can conclude that $a = b$. This is all that is needed to show that f is a 1-1 mapping. Note that f is not onto since, for example, there is no $n \in \mathbf{Z}$ such that $f(n) = 1$.

2) Let $2\mathbf{Z}$ denote the set of all even integers and consider the mapping $f : \mathbf{Z} \rightarrow 2\mathbf{Z}$ given by $f(n) = 2n$. Then f is onto, for if $m \in 2\mathbf{Z}$, then $m/2 \in \mathbf{Z}$ and $f(m/2) = m$. Of course, f is also 1-1 by the same argument we gave in (1) above. Thus there is a 1-1 correspondence between the set of all integers and the set of all even integers. ◀

A mapping $f : A \rightarrow A$ may also be called a *unary operation on A*. In Chapter III, we will be especially interested in *binary operations* on a set.

(1.9) DEFINITION. A *binary operation* \circ on a set A is a mapping $\circ : A \times A \rightarrow A$. If $\circ(a, b) = c$, we usually write $a \circ b = c$.

The word “binary” in Definition (1.9) means that we operate on two elements at a time.

► (1.10) EXAMPLES

1) The usual arithmetic operations are simple examples of binary operations. Addition is an operation on \mathbf{P} , but note that subtraction is not an operation on \mathbf{P} , since the difference of two positive integers is not always positive. Of course, subtraction is an operation on \mathbf{Z} . Similarly, division is not an operation on the nonzero integers but is an operation on the nonzero rational numbers.

2) Union and intersection are two binary operations on the power set of a set.

3) Suppose S is a set and let S^S denote the set of all mappings from S to S . For f and g in S^S , define $f \circ g$ to be the composition of f with g , so that $(f \circ g)(s) = f(g(s))$. Then this defines a binary operation on S^S . ◀

Two important special properties that an operation may possess are associativity and commutativity.

(1.11) DEFINITIONS. Suppose \circ is a binary operation on a set A . We say that \circ is *associative* if $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in A$. We say that \circ is *commutative* if $a \circ b = b \circ a$ for all $a, b \in A$.

If \circ is associative, then it follows by an easy induction argument that

$$a_1 \circ a_2 \circ \cdots \circ a_n$$

is well-defined, in the sense that it is independent of how we group (put parentheses around) these elements. (Note that we must group these elements since a binary operation can only be performed on two elements at a time.) However, we must not change the order in which two elements appear in this "product" unless we know that the operation is commutative as well.

► (1.12) EXAMPLES

1) Addition and multiplication are associative and commutative operations on \mathbf{Z} . However, subtraction is neither associative, nor commutative. ($1 - (2 - 3) \neq (1 - 2) - 3$ and $1 - 2 \neq 2 - 1$.)

2) Union and intersection of two subsets of a given set are associative and commutative operations.

3) Composition of mappings, the operation defined in Example (1.10.3) on S^S , is associative, but not commutative if S has at least two elements. For if x, y are two distinct elements of S , then there is a mapping f that takes everything in S to x and a mapping g that takes everything in S to y . Then we have $f \circ g = f$ and $g \circ f = g$. ◀

This is a good place to make a few remarks concerning proofs and counterexamples. When you are asked to prove that a relation satisfies a certain property, you must give a *general* proof that is valid in all cases. It is not enough simply to verify the property in a few instances. For example, in Exercise 1.11, you are asked to prove that multiplication of 2×2 real matrices is an associative operation. To do this, you should consider three arbitrary 2×2 real matrices, say

$$A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, B = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}, \text{ and } C = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix},$$

and then show that $A(BC) = (AB)C$.

On the other hand, when you are asked to show that a relation fails to have a certain property, you should give a *specific* counterexample. For example, in Exercise 1.11, you are also asked to show that multiplication of 2×2 real matrices is not commutative. To do this, you should produce two specific matrices (i.e., with specific real numbers as entries) A and B such that $AB \neq BA$.

EXERCISES

- 1.1. Suppose A is a set. Show that the relation \subseteq is a partial order (i.e., reflexive, antisymmetric, and transitive) on the power set of A .
- † 1.2. a) Give a relation on the set $\{1, 2, 3\}$ that is reflexive and symmetric but not transitive.
b) Give a relation on $\{1, 2, 3\}$ that is reflexive and transitive but not symmetric.
- † 1.3. Define a relation R on \mathbf{Z} by nRm if and only if $n - m$ is even. Prove that R is reflexive, symmetric, and transitive.
- † 1.4. Define a relation R on \mathbf{Z} by nRm if and only if $n - m$ is odd. Is R reflexive? symmetric? transitive?
- † 1.5. a) Construct a finite-state machine such that the simple connection relation is reflexive and transitive but not symmetric.
b) Construct a finite-state machine such that the simple connection relation is transitive and symmetric but not reflexive.
- † 1.6. Prove that the mapping $f : \mathbf{R} \rightarrow \mathbf{R}$ given by $f(x) = x^3$ is 1-1.
- † 1.7. Prove that the mapping $f : \mathbf{Z} \rightarrow \mathbf{Z}$ given by $f(n) = n + 1$ is onto.
- † 1.8. Suppose A and B are sets with n elements and $f : A \rightarrow B$ is a mapping. Prove that f is 1-1 if and only if f is onto. (See the Pigeonhole Principle in I.1).
- † 1.9. Suppose $f : A \rightarrow B$ is a 1-1, onto mapping. Define a relation R from B to A by $(b, a) \in R$ if and only if $b = f(a)$. Prove that R is a mapping. (R is called the *inverse* of f and is denoted f^{-1} .)
- † 1.10. Prove that composition of mappings is an associative operation on S^S .
- † 1.11. Let $M(2, \mathbf{R})$ denote the set of all 2×2 real matrices. Prove that matrix multiplication is an associative, but not commutative, operation on $M(2, \mathbf{R})$.
- † 1.12. Define an operation \circ on \mathbf{P} by $n \circ m = (n, m)$, the GCD of n and m . Is \circ associative?
- † 1.13. Define an operation \circ on \mathbf{Z} by $n \circ m = n + m - nm$. Is \circ associative?

† 1.14. Define a binary operation \circ on the set $S = \{e, a, b\}$ by the following table:

\circ	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

(Here, we enter $x \circ y$ in the row corresponding to x and the column corresponding to y .) Is \circ associative? Is \circ commutative?

2. Relations on Finite Sets

If R is a relation from a finite set A to a finite set B , then we may list the ordered pairs in R (viewing R as a subset of $A \times B$). A more convenient way to represent a relation from one finite set to another finite set is by a matrix. Suppose $A = \{a_1, a_2, \dots, a_m\}$ and $B = \{b_1, b_2, \dots, b_n\}$ and suppose R is a relation from A to B .

(2.1) DEFINITION. The matrix $M_R = [m_{ij}]$ of R is the $m \times n$ matrix with entries from $\mathbf{B} = \{0, 1\}$ defined by

$$m_{ij} = \begin{cases} 1 & \text{if } a_i R b_j \\ 0 & \text{if } a_i \not R b_j \end{cases}$$

for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. (We write $a \not R b$ if a is not related to b by R .)

► (2.2) EXAMPLES

1) If $|A| = n$, then the matrix of the relation “=” on A is the $n \times n$ identity matrix.

2) If $A = \{0, 1, 2\}$ and R is the relation $\{(0, 1), (1, 0), (0, 2), (1, 2)\}$ on A , then

$$M_R = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \blacktriangleleft$$

In linear algebra, one represents linear transformations by matrices, and a key fact is that the composition of two linear transformations is represented by the product of their respective matrices. We have a similar result for

relations and their matrices if we multiply the matrices in a special way. We define the operations \vee (join) and \wedge (meet) on \mathbf{B} by

$$\begin{aligned} 0 \vee 0 &= 0, 0 \vee 1 = 1 \vee 0 = 1, 1 \vee 1 = 1 \\ 0 \wedge 0 &= 0, 0 \wedge 1 = 1 \wedge 0 = 0, 1 \wedge 1 = 1. \end{aligned}$$

(These operations go a long way toward making \mathbf{B} into a Boolean algebra — see VI.1.) Now suppose $A = [a_{ik}]$ is an $m \times l$ matrix and $B = [b_{kj}]$ is an $l \times n$ matrix with the entries in both matrices from \mathbf{B} .

(2.3) DEFINITION. The *Boolean product* of A and B , denoted $A \odot B$, is the $m \times n$ matrix $C = [c_{ij}]$ with entries from \mathbf{B} defined by

$$c_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee \cdots \vee (a_{il} \wedge b_{lj}).$$

It is not hard to see from the definition of \vee and \wedge that we have

$$c_{ij} = \begin{cases} 1 & \text{if } a_{ik} \text{ and } b_{kj} \text{ both equal 1 for some } k, 1 \leq k \leq l \\ 0 & \text{otherwise.} \end{cases}$$

► **(2.4) EXAMPLE.** Suppose

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then

$$A \odot B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}. \blacktriangleleft$$

(2.5) PROPOSITION. Suppose $A = \{a_1, a_2, \dots, a_m\}$, $B = \{b_1, b_2, \dots, b_l\}$, and $C = \{c_1, c_2, \dots, c_n\}$. If R is a relation from A to B and S is a relation from B to C , then

$$M_{S \circ R} = M_R \odot M_S.$$

PROOF. By definition of the Boolean product, the entry in the i^{th} row and j^{th} column of $M_R \odot M_S$ is 1 precisely when there exists $b_k \in B$ such that $a_i R b_k$ and $b_k R c_j$. But this is exactly when a_i and c_j are related by $S \circ R$. ■

We can define a partial order \leq on \mathbf{B} by specifying $0 \leq 0, 0 \leq 1$, and $1 \leq 1$ (and $1 \not\leq 0$). This can be extended to a partial order on $m \times n$ matrices with entries from \mathbf{B} as follows.

(2.6) DEFINITION. Suppose $A = [a_{ij}]$ and $B = [b_{ij}]$ are two $m \times n$ matrices with entries from \mathbf{B} . We say $A \leq B$ if and only if $a_{ij} \leq b_{ij}$ for all i and j .

► **(2.7) EXAMPLE.** Suppose

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Then $A \leq C$ and $B \leq C$, but $A \not\leq B$. ◀

Special properties of a relation on a finite set can now be characterized in terms of the matrix representing the relation.

(2.8) PROPOSITION. Suppose $A = \{a_1, a_2, \dots, a_n\}$ and R is a relation on A . Let $M_R = [m_{ij}]$ denote the matrix representing R . Then

- 1) R is reflexive if and only if $I \leq M_R$, where I is the $n \times n$ identity matrix.
- 2) R is symmetric if and only if M_R is symmetric (i.e., $M_R = M_R^t$, its transpose).
- 3) R is transitive if and only if $M_{R^2} \leq M_R$.

PROOF. Statements (1) and (2) follow immediately from the definitions.

Assume R is transitive and suppose the (i, j) entry of M_{R^2} (i.e., the entry in the i^{th} row and j^{th} column) is a 1. We must show that the (i, j) entry of M_R is also a 1. Since the (i, j) entry of $M_{R^2} = M_R \odot M_R$ is 1, there exists $k, 1 \leq k \leq n$, such that $a_i R a_k$ and $a_k R a_j$. Since R is transitive, we can conclude that $a_i R a_j$, which means that the (i, j) entry of M_R is a 1.

Conversely, assume $M_{R^2} \leq M_R$ and suppose $a_i R a_k$ and $a_k R a_j$. Then the (i, j) entry of M_{R^2} is a 1. Hence the (i, j) entry of M_R must also be a 1, which implies that $a_i R a_j$. Thus R is transitive. ■

► **(2.9) EXAMPLE.** Let R be the relation on $A = \{a_1, a_2, a_3\}$ whose matrix is

$$M_R = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \quad 46$$

Then R is not reflexive since the $(3, 3)$ entry is not a 1. Since $M_R^t = M_R$, we see that R is symmetric. To determine if R is transitive, we may compute

$$M_{R^2} = M_R \odot M_R = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Since $M_{R^2} \not\leq M_R$, we conclude that R is not transitive. (Note that a_3Ra_1 and a_1Ra_3 , but a_3 is not related to itself.) ◀

Another way to represent a relation R on a finite set A is by a directed graph (digraph). The vertices of the digraph correspond to the elements in A and we draw an arc from a_i to a_j if and only if a_iRa_j . The matrix M_R of Definition (2.1) is the adjacency matrix (see VII.2) of the digraph of R .

- **(2.10) EXAMPLE.** If R is the relation on $A = \{a_1, a_2, a_3\}$ given by the matrix M_R in Example (2.9), then the digraph of R is as shown in Figure 2.1. ◀

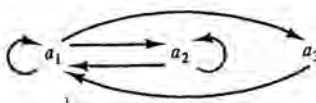


Figure 2.1

Suppose A is a finite set and R is a partial order on A (so that R is reflexive, transitive, and antisymmetric). Then instead of considering the digraph of R , one often considers the *Hasse diagram* of R . The Hasse diagram is a graph whose vertices correspond to the elements of A and in which an edge connects a_i to a_j if and only if a_iRa_j , $i \neq j$, and there is no element a_k , $k \neq i, j$, of A such that a_iRa_k and a_kRa_j . We draw the Hasse diagram vertically, with a_j above a_i if a_iRa_j (and $i \neq j$). This gives the graph a natural “direction,” and in doing this we think of an element a_j as being “greater” or “higher” than an element a_i if a_iRa_j (and $i \neq j$).

- **(2.11) EXAMPLE.** Let $A = \{1, 2, 3, 4, 6, 9, 12, 18, 36\}$ denote the positive divisors of 36. The relation “divides” defines a partial order on A . The Hasse diagram of this partial order is shown in Figure 2.2. ◀

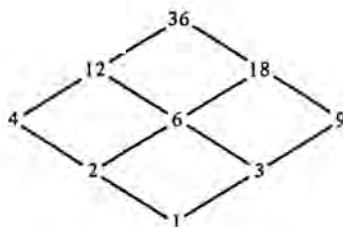


Figure 2.2

EXERCISES

- † 2.1. Let $A = \{1, 2, 3, 4\}$. For each of the following relations R on A , find the matrix of R and the digraph of R . Also, determine if R is reflexive, symmetric, or transitive.
- $R = \{(1, 1), (1, 2), (1, 3), (1, 4)\}$.
 - $R = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), (3, 2), (3, 3), (3, 4), (4, 3), (4, 4)\}$.
 - $R = A \times A$.
 - $R = \{(1, 1), (2, 2), (3, 3), (4, 4)\}$.
 - $R = \{(1, 2), (2, 1), (1, 3), (3, 1), (1, 4), (4, 1), (2, 3), (3, 2), (2, 4), (4, 2), (3, 4), (4, 3)\}$.
- † 2.2. a) Show that there are 2^{n^2} distinct relations on a set with n elements.
 b) Show that there are $2^{n(n-1)}$ distinct reflexive relations on a set with n elements. (Hint: Use the fact that the matrix of a reflexive relation must have 1's down the main diagonal.)
 c) Show that there are $2^{n(n+1)/2}$ distinct symmetric relations on a set with n elements. (Hint: Note that $n + (n-1) + \cdots + 1 = n(n+1)/2$.)
- 2.3. Suppose R is a relation on a finite set. Show that R is antisymmetric if and only if $M_R = [m_{ij}]$ satisfies the following property: if $i \neq j$, then $m_{ij} = 0$ or $m_{ji} = 0$.
- 2.4. Draw the Hasse diagram associated with each of the following partially ordered sets (posets).

- a) $A = \{1, 2, 3, 6, 9, 18\}$ with the relation “divides.”
 b) $A = \{1, 2, 3, 4\}$ with the relation whose matrix is

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- c) $A = \{1, 2, 3, 5, 6, 10, 15, 30\}$ with the relation “divides.” (This Hasse diagram should be drawn in 3-space.)
 d) $A =$ the set of all 2×2 matrices with entries from \mathbf{B} with the relation \leq from Definition (2.6).
- 2.5. How many partial orders are there on a set with three elements? Try to answer this by determining the Hasse diagram associated to each such partial order. If we call “equivalent” two Hasse diagrams that “look the same” but just have differently labeled vertices, then how many Hasse diagrams would there be in this case such that no two distinct Hasse diagrams were equivalent?

3. Equivalence Relations and Partitions

We now come to one of the most important topics in this book — the notion of an equivalence relation. Equivalence relations will appear often in the following chapters and, in fact, they appear often throughout all of mathematics.

(3.1) DEFINITION. A relation R on a set A is called an *equivalence relation* on A if R is reflexive, symmetric, and transitive.

If R is an equivalence relation on A and if aRb , then we say that a and b are *equivalent* (under R).

► (3.2) EXAMPLES

- 1) A simple example of an equivalence relation is the relation “=.”
- 2) Consider the relation \equiv_2 on \mathbf{Z} defined by $a \equiv_2 b$ if and only if $2|b-a$. Another way to define this relation is to say $a \equiv_2 b$ if and only if a and b are both even or both odd. Let’s verify that \equiv_2 satisfies the three properties necessary to make it an equivalence relation. (This was Exercise 1.3.)

Suppose $a \in \mathbf{Z}$. Then obviously a and a are both even or both odd, so we have $a \equiv_2 a$. This shows that \equiv_2 is reflexive. Now suppose $a \equiv_2 b$. Then we also have $b \equiv_2 a$ since b and a are both even or both odd. Thus \equiv_2 is symmetric. Finally, to show that \equiv_2 is transitive, we suppose $a \equiv_2 b$ and $b \equiv_2 c$ and we must see that $a \equiv_2 c$. But if a and b are both even or both odd, and if b and c are both even or both odd, then a and c are both even or both odd. Hence we have $a \equiv_2 c$ and we are done.

3) Consider the relation R on the set $\{1, 2, 3\}$ given by

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

It is not hard to verify that R is reflexive, symmetric, and transitive. Notice that “=” (i.e., the relation $\{(1, 1), (2, 2), (3, 3)\}$) is another equivalence relation on the set $\{1, 2, 3\}$. In general, as will become clear below, there are many equivalence relations that one may put on a given set.

4) Let $M = (S, X, Z, \tau, \omega)$ be a Mealy machine. Define a relation R on the set of states S by $\sigma_i R \sigma_j$ if $\omega^*(\sigma_i, \hat{x}) = \omega^*(\sigma_j, \hat{x})$ for every finite sequence $\hat{x} = x_1 x_2 \cdots x_n$ of inputs; i.e., given an input string \hat{x} , the machine will produce exactly the same output string whether it was in state σ_i or in state σ_j when the first input in the string was fed into the machine. Then it is not hard to see that R is an equivalence relation on S . ◀

An equivalence relation on a set breaks up that set into subsets, called equivalence classes, such that two elements are equivalent if and only if they are in the same equivalence class. Formally, we have:

(3.3) DEFINITIONS. Suppose R is an equivalence relation on a set A and let $a \in A$. The *equivalence class of a* (with respect to R), denoted $[a]_R$, is defined by

$$[a]_R = \{x \in A : aRx\}.$$

A member of an equivalence class is called a *representative* of that class.

If it is clear to which equivalence relation we are referring, we will write $[a]$ instead of $[a]_R$.

► (3.4) EXAMPLES

1) The equivalence classes of “=” on a set A consist of all singleton subsets of A since each element of A is equal only to itself.

2) If R is the equivalence relation \equiv_2 of Example (3.2.2), then R has two distinct equivalence classes. One of these consists of all the even integers (note that $[0]_R = [2]_R = [2k]_R$, where $k \in \mathbf{Z}$), while the other consists of all the odd integers.

3) If R is the equivalence relation in Example (3.2.3); then we have $[1]_R = \{1\}$, $[2]_R = \{2, 3\} = [3]_R$. ◀

The crucial fact about equivalence classes is that they either coincide or are disjoint (i.e., have empty intersection). This fact follows from the next result.

(3.5) PROPOSITION. Suppose R is an equivalence relation on a set A and suppose $a, b \in A$.

- (1) $[a] = [b]$ if and only if aRb .
- (2) If $[a] \neq [b]$, then $[a] \cap [b] = \emptyset$.

PROOF. (1): There are two directions to prove. First, suppose $[a] = [b]$. Since R is reflexive, we have bRb , so $b \in [b]$. Then since $[a] = [b]$, we have $b \in [a]$, which means that aRb .

Conversely, suppose aRb . We must show that the sets $[a]$ and $[b]$ are identical. To do this, we will show that each is a subset of the other. First, suppose c is any element in $[a]$. Then we have aRc . Since aRb and R is symmetric, we have bRa . Now, since R is transitive, it follows from bRa and aRc that we have bRc . Thus $c \in [b]$ and so $[a] \subseteq [b]$. Similarly, if $d \in [b]$, then bRd and aRb , so we have aRd and $d \in [a]$. Thus $[b] \subseteq [a]$ and we conclude that $[a] = [b]$.

(2): Here, we are assuming that $[a]$ and $[b]$ are not identical, and we must then show that they have no elements in common. By way of contradiction, assume $c \in [a] \cap [b]$. Then aRc and bRc , hence cRb . By transitivity of R , we then have aRb . But the first statement in the proposition then shows that $[a]$ and $[b]$ are identical. This contradiction shows that if $[a]$ and $[b]$ are not identical, then they are disjoint. ■

(3.6) DEFINITION. A *partition* of a set A is a collection $\{S_\lambda : \lambda \in \Lambda\}$ of nonempty subsets of A (where Λ is some index set) such that:

- (1) $\cup_{\lambda \in \Lambda} S_\lambda = A$ and
- (2) $S_\lambda \cap S_\mu = \emptyset$ if $\lambda \neq \mu$.

Equivalently, the collection $\{S_\lambda\}$ of nonempty subsets of A is a partition of A if each element of A is in exactly one of the S_λ .

*If R is an equivalence relation on a set A ,
then the equivalence classes of R partition A .*

Indeed, since R is reflexive, each element of A is in some equivalence class, and Proposition (3.5) shows that no element can be in two distinct equivalence classes.

The next important theorem says that equivalence classes and partitions are really the same things.

(3.7) THEOREM. Let A be a set. There is a one-to-one correspondence between equivalence relations on A and partitions of A .

PROOF. Suppose R is an equivalence relation on A . Then, as we noted above, the equivalence classes of R give a partition of A .

Conversely, suppose $\{S_\lambda : \lambda \in \Lambda\}$ is a partition of A . Define a relation R on A by aRb if and only if a and b are in the same S_λ . It is not hard to see that R is reflexive, symmetric, and transitive, hence an equivalence relation.

Finally, to see that this correspondence between equivalence relations and partitions is one-to-one, suppose R_1 and R_2 are two equivalence relations that give rise to the same partition of A . This means that the equivalence classes of R_1 and R_2 are exactly the same. By Proposition (3.5), this implies that aR_1b if and only if aR_2b . Hence $R_1 = R_2$ (i.e., R_1 and R_2 are the same subset of $A \times A$). ■

► (3.8) EXAMPLES

1) Let's find all distinct equivalence relations on the set $A = \{1, 2, 3\}$. By Theorem (3.7), this amounts to finding all possible partitions of A . The possible partitions of A are:

$$\begin{aligned} &\{1, 2, 3\} \\ &\{1\}, \{2, 3\} \\ &\{2\}, \{1, 3\} \\ &\{3\}, \{1, 2\} \\ &\{1\}, \{2\}, \{3\}. \end{aligned}$$

Thus there are five distinct equivalence relations on A . Note that the last partition in our list above corresponds to the relation “=.”

2) When we divide an integer by 3 (as in the Division Algorithm), the remainder may be 0, 1, or 2. This suggests the following partition of \mathbf{Z} into the three subsets:

$$\begin{aligned} S_0 &= \{3k : k \in \mathbf{Z}\} \\ S_1 &= \{3k + 1 : k \in \mathbf{Z}\} \\ S_2 &= \{3k + 2 : k \in \mathbf{Z}\}. \end{aligned}$$

This partition defines an equivalence relation called “congruence mod 3,” which is usually denoted \equiv_3 . Another way to describe this relation is by saying $a \equiv_3 b$ if and only if $3|b - a$. ◀

EXERCISES

† 3.1. In each part, determine if the given relation is an equivalence relation. If it is, find the equivalence classes. If it is not, tell which property or properties fail.

a) The relation \leq on \mathbf{Z} .

b) The relation R on \mathbf{Z} defined by aRb if and only if $|a| = |b|$.

c) The relation R on \mathbf{Z} defined by aRb if and only if $a^2 + a = b^2 + b$.

d) The relation R on the set $\{1, 2, 3\}$, where

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

e) The relation R on the set $\{1, 2, 3, 4\}$ whose matrix is

$$M_R = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

f) The relation R on the set $\{1, 2, 3, 4\}$ whose matrix is

$$M_R = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

g) The relation R on the set $\{1, 2, 3, 4\}$ whose matrix is

$$M_R = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

† 3.2. In each part, write the ordered pairs in the equivalence relation associated to the given partition of the set $\{1, 2, 3, 4, 5\}$.

a) $\{\{1, 4, 5\}, \{2, 3\}\}$.

b) $\{\{1\}, \{2\}, \{3, 4\}, \{5\}\}$.

- 3.3. Define a relation R on $\mathbf{Z} \times \mathbf{P}$ by $(a, b)R(c, d)$ if and only if $ad = bc$. Prove that R is an equivalence relation. (The set of equivalence classes of R may be identified with the rational numbers.)
- † 3.4. Define \equiv_n on \mathbf{Z} by $a \equiv_n b$ if and only if $n|(b - a)$. Prove that \equiv_n is an equivalence relation. Show that there are n distinct equivalence classes for this equivalence relation and describe these classes.
- 3.5. Define a relation R on \mathbf{Q} by xRy if and only if $y - x \in \mathbf{Z}$. Prove that R is an equivalence relation.
- † 3.6. How many distinct equivalence relations may be put on a set with four elements? (The generalization of this question to a set with n elements is not an easy one to solve. See the discussion of "Bell numbers" on p. 92 in [1].)
- 3.7. If R_1 and R_2 are equivalence relations on a set A , must $R_1 \circ R_2$ be an equivalence relation on A ?
- 3.8. Suppose $f : X \rightarrow Y$ is a mapping of sets. Define a relation R on X by $x_1 R x_2$ if and only if $f(x_1) = f(x_2)$. Prove that R is an equivalence relation on X . (The equivalence classes of R are called the *fibers* of f .)

4. Minimization of Machines

In this section, we will consider an application of equivalence relations to the problem of finding a minimal finite-state machine to do a given task. If we construct a finite-state machine to perform a certain job, there is no guarantee that we have done the most efficient construction. There may be another machine with fewer states that can perform the same task. If there is no machine with fewer states that does the same job as ours, then we would call our machine *minimal*. There is an obvious cost benefit in being able to find a minimal machine for a given job. Our goal here is to present an algorithm for finding a minimal machine equivalent to a given machine. First we need to formalize some of these concepts.

Suppose $M = (S, X, Z, \tau, \omega)$ and $M_0 = (S_0, X, Z, \tau_0, \omega_0)$ are two finite-state machines with the same input and output alphabets (usually $\{0, 1\}$). We will be dealing with Moore machines below, but the definitions we will give apply, with minor modifications, to other types of finite-state machines as well.

(4.1) DEFINITION. We say that M_0 covers, or *simulates*, M , denoted $M_0 \geq M$, if there exists a mapping $\varphi : S \rightarrow S_0$ such that for every input string $\hat{x} \in X^*$ and for every $\sigma \in S$, we have

$$\omega^*(\sigma, \hat{x}) = \omega_0^*(\varphi(\sigma), \hat{x}).$$

This means that if M is in state σ and M_0 is in state $\varphi(\sigma)$ and if both machines are then fed the same input string, then they will both produce the same output string. Intuitively, M_0 covers M if M_0 can do anything that M can do. We leave it as an exercise to show that the relation \geq of (4.1) is reflexive and transitive. However, it is not the case that \geq is antisymmetric.

► **(4.2) EXAMPLE.** Consider the Moore machines $M = (S, X, Z, \tau, \omega)$ with the digraph shown in Figure 4.1 and $M' = (S', X, Z, \tau', \omega')$ with the digraph shown in Figure 4.2.

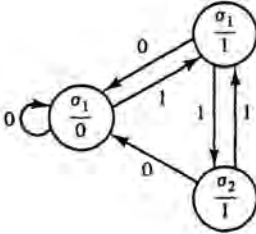


Figure 4.1

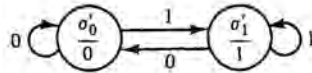


Figure 4.2

These machines are both very simple. They both give an output of 0 when the input is 0 and an output of 1 when the input is 1. Let's see that each machine simulates the other according to Definition (4.1). To show $M \geq M'$, we can define $\varphi : S' \rightarrow S$ by $\varphi(\sigma'_0) = \sigma_0$ and $\varphi(\sigma'_1) = \sigma_1$. Then it is easy to see (because of the simple description of the operation of these machines) that the condition in Definition (4.1) is satisfied. To show that $M' \geq M$, we can define $\psi : S \rightarrow S'$ by $\psi(\sigma_0) = \sigma'_0$, $\psi(\sigma_1) = \psi(\sigma_2) = \sigma'_1$. Again, it is easy to see that the condition in Definition (4.1) is satisfied. Thus $M \geq M'$ and $M' \geq M$, but $M \neq M'$ (proving that \geq is not antisymmetric). Note that in this case, M' is clearly a Moore machine with the minimal

number of states that can simulate M (i.e., that can output 0 on an input of 0 and output 1 on an input of 1), since any Moore machine to perform this job must have at least two states. ◀

(4.3) DEFINITION. Suppose M and M' are finite-state machines. If $M \geq M'$ and $M' \geq M$, then we say that M and M' are *equivalent* and we write $M \equiv M'$.

(4.4) DEFINITION. A finite-state machine $\widetilde{M} = (\widetilde{S}, X, Z, \widetilde{\tau}, \widetilde{\omega})$ is called *minimal* if whenever a machine $M_0 = (S_0, X, Z, \tau_0, \omega_0)$ covers \widetilde{M} , then $|\widetilde{S}| \leq |S_0|$.

Informally, a machine is minimal if there does not exist a machine with fewer states that can perform the same task. The machine M' of Example (4.2) is an example of a minimal machine.

The problem to be addressed in this section can now be phrased as follows: Given a finite-state machine M , find a minimal machine \widetilde{M} that is equivalent to M . To solve this problem, we will introduce a family of equivalence relations $\{E_k\}$, $k = 0, 1, \dots$, on the states of a machine. We define these relations recursively as follows.

(4.5) DEFINITION. Suppose $M = (S, X, Z, \tau, \omega)$ is a Moore machine and suppose $\sigma_i, \sigma_j \in S$.

- 1) σ_i and σ_j are called *0-equivalent*, denoted $\sigma_i E_0 \sigma_j$, if $\omega(\sigma_i) = \omega(\sigma_j)$.
- 2) For $k \geq 1$, we say σ_i and σ_j are *k-equivalent*, denoted $\sigma_i E_k \sigma_j$, if $\sigma_i E_{k-1} \sigma_j$ and if for all $x \in X$, we have $\tau(\sigma_i, x) E_{k-1} \tau(\sigma_j, x)$.
- 3) σ_i is said to be *equivalent* to σ_j , denoted $\sigma_i E \sigma_j$, if $\sigma_i E_k \sigma_j$ for all $k \geq 0$.

While Definition (4.5) will make some of the proofs to come easier, the reader may want to think of k -equivalence in terms of the following lemma.

(4.6) LEMMA. Two states σ_i and σ_j are k -equivalent if and only if for every input string \hat{x} of length at most k , the machine produces the same output string whether it was in σ_i or in σ_j when the first input in \hat{x} was received (i.e., $\omega^*(\sigma_i, \hat{x}) = \omega^*(\sigma_j, \hat{x})$).

PROOF. We use induction. The lemma is clearly true for $k = 0$. Assume it is true for $k = n - 1$. Suppose σ_i and σ_j are n -equivalent. Then the first input x of a string of length at most n takes σ_i and σ_j to states that are $(n - 1)$ -equivalent, so these states, $\tau(\sigma_i, x)$ and $\tau(\sigma_j, x)$, in particular are 0-equivalent and have the same output associated to each of them. By induction, the remaining inputs in the string produce the same output whether the machine is in $\tau(\sigma_i, x)$ or in $\tau(\sigma_j, x)$ since these states are $(n - 1)$ -equivalent. 56

It remains to show the converse; namely, that if for every input string of length at most n the machine produces the same output string whether it was in σ_i or in σ_j when the first input was received, then $\sigma_i E_n \sigma_j$. If \hat{y} is any string of length at most $n - 1$, then our assumption says that $\omega^*(\sigma_i, \hat{y}) = \omega^*(\sigma_j, \hat{y})$. Hence, by induction, we may conclude that σ_i and σ_j are $(n - 1)$ -equivalent. Now let $x \in X$. Then our assumption says that $\omega^*(\sigma_i, x\hat{y}) = \omega^*(\sigma_j, x\hat{y})$. Hence, we have $\omega^*(\tau(\sigma_i, x), \hat{y}) = \omega^*(\tau(\sigma_j, x), \hat{y})$. By induction, again, we conclude that $\tau(\sigma_i, x) E_{n-1} \tau(\sigma_j, x)$. Thus $\sigma_i E_n \sigma_j$ and the proof is complete. ■

It follows then that σ_i and σ_j are equivalent if for any possible input string, the machine produces the same output string whether it was in σ_i or in σ_j when the first input in the string was received. We leave it as an exercise to see that the relations in Definition (4.5) are indeed equivalence relations.

Recalling that the equivalence classes of an equivalence relation on S partition S , we let Π_k denote the partition of S into the equivalence classes of E_k , and we let Π denote the partition of S into the equivalence classes of E .

To find a minimal machine \widetilde{M} equivalent to a given machine M , we will take the equivalence classes of E as our states in \widetilde{M} . We must then prove that the resulting machine has a well-defined transition function, is equivalent to M , and is minimal.

► (4.7) EXAMPLES

1) Consider the machine M of Example (4.2). Obviously, we have $\Pi_0 = \{\{\sigma_0\}, \{\sigma_1, \sigma_2\}\}$. In this case, to find the equivalence classes of the relation E_1 , we simply must see if σ_1 and σ_2 are 1-equivalent. Therefore, we must see if $\tau(\sigma_1, 0)$ and $\tau(\sigma_2, 0)$ are 0-equivalent and if $\tau(\sigma_1, 1)$ and $\tau(\sigma_2, 1)$ are also 0-equivalent. We have

$$\begin{aligned}\tau(\sigma_1, 0) &= \tau(\sigma_2, 0) = \sigma_0 \\ \tau(\sigma_1, 1) &= \sigma_2, \quad \tau(\sigma_2, 1) = \sigma_1\end{aligned}$$

and since $\sigma_2 E_0 \sigma_1$, we conclude that $\sigma_1 E_1 \sigma_2$. Hence $\Pi_1 = \Pi_0$. It is not hard to see that, for this (rather trivial) machine, $\Pi_k = \Pi_0$ for all $k \geq 0$.

2) Suppose M is the Moore machine whose digraph is shown in Figure 4.3.

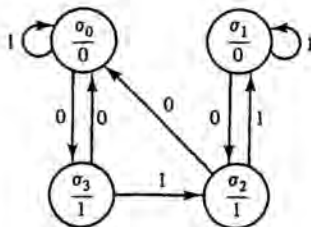


Figure 4.3

Obviously, $\Pi_0 = \{\{\sigma_0, \sigma_1\}, \{\sigma_2, \sigma_3\}\}$. Now, we have

$$\tau(\sigma_0, 0) = \sigma_3, \tau(\sigma_1, 0) = \sigma_2, \tau(\sigma_0, 1) = \sigma_0, \tau(\sigma_1) = \sigma_1.$$

Since $\sigma_3 E_0 \sigma_2$ and $\sigma_0 E_0 \sigma_1$, we conclude that $\sigma_0 E_1 \sigma_1$. We also have

$$\tau(\sigma_2, 0) = \sigma_0, \tau(\sigma_3, 0) = \sigma_0, \tau(\sigma_2, 1) = \sigma_1, \tau(\sigma_3, 1) = \sigma_2.$$

Now, $\sigma_0 E_0 \sigma_0$, but σ_1 and σ_2 are not 0-equivalent. Therefore, σ_2 and σ_3 are not 1-equivalent. Clearly, σ_2 and σ_3 are also not 1-equivalent to either σ_0 or σ_1 (since neither σ_2 nor σ_3 were 0-equivalent to σ_0 or σ_1). Hence $\Pi_1 = \{\{\sigma_0, \sigma_1\}, \{\sigma_2\}, \{\sigma_3\}\}$. We leave it to the reader to see that $\Pi_2 = \{\{\sigma_0\}, \{\sigma_1\}, \{\sigma_2\}, \{\sigma_3\}\}$ (the key is that σ_2 and σ_3 are not 1-equivalent) and that $\Pi_k = \Pi_2$ for all $k \geq 2$. ◀

(4.8) LEMMA. Each of the equivalence classes in Π_{k+1} is a subset of an equivalence class in Π_k .

PROOF. It was part of Definition (4.5) that if $\sigma_i E_{k+1} \sigma_j$, then it is necessary that $\sigma_i E_k \sigma_j$. The lemma follows immediately from this. ■

It follows from Lemma (4.8) that if $|S| = n$, then there can be at most $n - 1$ distinct equivalence relations among the E_k 's; and the only way there could be actually $n - 1$ of them is if all but one of the states were 0-equivalent, precisely $n - 2$ of these 0-equivalent states were k_1 -equivalent, for some $k_1 > 0$, precisely $n - 3$ of these $n - 2$ states were k_2 -equivalent for some $k_2 > k_1$, and so on until we reached some k_{n-2} such that each state was k_{n-2} -equivalent only to itself. (So, in this scenario, the distinct equivalence relations would be $E_0, E_{k_1}, \dots, E_{k_{n-2}}$. Note then that we would have $E = E_{k_{n-2}}$.) However, we can say even more. The next lemma says that if $E_k = E_{k+1}$, then $E_r = E_k$ for all $r \geq k$. This means that in the above scenario in which we had $n - 1$ distinct equivalence relations, we would have to have $k_1 = 1, k_2 = 2, \dots, k_{n-2} = n - 2$.

(4.9) LEMMA. If $E_k = E_{k+1}$, then $E_r = E_k$ for all $r \geq k$ and hence $E = E_k$.

PROOF. The proof is by induction (the induction here starts with the case $r = k$). The lemma is obviously true for $r = k$. Assume $E_r = E_k$ for all r such that $k \leq r \leq n$, where n is an integer greater than k . We must show that $E_{n+1} = E_k$. It suffices to show that if $\sigma_i E_k \sigma_j$, then $\sigma_i E_{n+1} \sigma_j$. By our induction hypothesis, if $\sigma_i E_k \sigma_j$, then $\sigma_i E_n \sigma_j$. Hence, we are reduced to showing that if $x \in X$, then $\tau(\sigma_i, x) E_n \tau(\sigma_j, x)$. Now, since $\sigma_i E_n \sigma_j$, we know that $\tau(\sigma_i, x) E_{n-1} \tau(\sigma_j, x)$ for all $x \in X$. But by our induction hypothesis, $E_{n-1} = E_n$. Hence we have $\tau(\sigma_i, x) E_n \tau(\sigma_j, x)$. ■

It is now a fairly simple matter to give an algorithm for computing the partition Π .

(4.10) ALGORITHM FOR COMPUTING Π . Suppose M is a Moore machine.

Step 1. Find Π_0 by partitioning S into subsets according to which output is associated to each state. (So if $Z = \{0, 1\}$ and if not every state produces the same output, then there will be two subsets in Π_0 .) Set $r = 1$.

Step 2. We find Π_r , for $r > 0$, as follows. For each subset in Π_{r-1} , consider every possible pair of states σ_i, σ_j in that subset. These states will belong to the same subset in Π_r if and only if for every $x \in X$, the states $\tau(\sigma_i, x)$ and $\tau(\sigma_j, x)$ belong to the same subset in Π_{r-1} (note that this latter subset need not be the same one to which σ_i and σ_j belong).

Step 3. If the partition Π_r is exactly the same as the partition Π_{r-1} , then we know $\Pi = \Pi_{r-1}$ and the algorithm stops (by Lemma (4.9)). If the subsets of Π_r are all singletons, then $\Pi = \Pi_r$ and the algorithm stops. If some subset in Π_r is a proper subset of a subset in Π_{r-1} and some subset in Π_r contains at least two states, then replace r by $r + 1$ and go to Step 2.

By our remarks above, the algorithm must terminate in at most $|S| - 1$ steps.

► **(4.11) EXAMPLE.** Let M be the Moore machine whose defining table is shown in Table (4.12). From the table, it is clear that Π_0 is the partition of S into the two subsets

$$\{\sigma_1, \sigma_2, \sigma_5\} \text{ and } \{\sigma_0, \sigma_3, \sigma_4\}.$$

Now we find Π_1 . We have $\sigma_1 E_1 \sigma_2$ since $\tau(\sigma_1, 0) = \sigma_4$ is 0-equivalent to $\tau(\sigma_2, 0) = \sigma_4$ and $\tau(\sigma_1, 1) = \sigma_2$ is 0-equivalent to $\tau(\sigma_2, 1) = \sigma_1$. We have $\sigma_1 E_1 \sigma_5$ since $\tau(\sigma_1, 0) = \sigma_4$ is 0-equivalent to $\tau(\sigma_5, 0) = \sigma_3$ and $\tau(\sigma_1, 1) = \sigma_2$

is 0-equivalent to $\tau(\sigma_5, 1) = \sigma_5$. Note that since E_1 is an equivalence relation, we must then have $\sigma_2 E_1 \sigma_5$. Therefore, the set $\{\sigma_1, \sigma_2, \sigma_5\}$ is also one of the subsets in the partition Π_1 .

(4.12) TABLE

Present State	Next State		Output
	0	1	
σ_0	σ_3	σ_5	1
σ_1	σ_4	σ_2	0
σ_2	σ_4	σ_1	0
σ_3	σ_2	σ_0	1
σ_4	σ_5	σ_0	1
σ_5	σ_3	σ_5	0

Continuing to the second subset in Π_0 , we see that σ_0 is not 1-equivalent to either σ_3 or σ_4 since $\sigma_3 = \tau(\sigma_0, 0)$ is not 0-equivalent to either $\sigma_2 = \tau(\sigma_3, 0)$ or $\sigma_5 = \tau(\sigma_4, 0)$. Finally, it is not hard to check that we have $\sigma_3 E_1 \sigma_4$. Hence

$$\Pi_1 = \{\{\sigma_1, \sigma_2, \sigma_5\}, \{\sigma_0\}, \{\sigma_3, \sigma_4\}\}.$$

Now we must compute Π_2 . It is not hard to see that $\{\sigma_1, \sigma_2, \sigma_5\}$ is still one of the subsets in Π_2 ; the key point is that $\sigma_4 = \tau(\sigma_1, 0)$ and $\sigma_3 = \tau(\sigma_5, 0)$ are 1-equivalent. It is easy to check that $\sigma_3 E_2 \sigma_4$. It follows that $\Pi_2 = \Pi_1$, so the algorithm stops and

$$\Pi = \{\{\sigma_1, \sigma_2, \sigma_5\}, \{\sigma_0\}, \{\sigma_3, \sigma_4\}\}. \blacktriangleleft$$

We now form a new Moore machine $\widetilde{M} = (\Pi, X, Z, \tilde{\tau}, \tilde{\omega})$, called the quotient of \widetilde{M} by the equivalence relation E , by taking the subsets in Π as the states in \widetilde{M} . Since any two states in a given subset of Π are 0-equivalent, we may define the output function $\tilde{\omega}$ for \widetilde{M} by associating to each subset of Π the output associated to any state in that subset. We define the transition mapping for \widetilde{M} as follows. Suppose $A \in \Pi$ and let $\sigma \in A$. If $x \in X$, define $\tilde{\tau}(A, x)$ to be the subset in Π that contains $\tau(\sigma, x)$. We must check that this mapping is well-defined in the sense that it does not depend on our choice of $\sigma \in A$. Indeed, if σ' were another state in A , then $\tau(\sigma, x)$ and $\tau(\sigma', x)$ must be equivalent, so they must lie in the same subset in Π . This completes the construction of \widetilde{M} .

Notice that if the partition Π consists entirely of singleton sets (i.e., if the equivalence relation E is " $=$ "), then the machine \widetilde{M} will look just

like the original machine M . When this happens, it means that the original machine M was a minimal machine.

To complete the program outlined above, we still need to see that \widetilde{M} is equivalent to the original machine M and that \widetilde{M} is a minimal machine. But first, let's do a couple of examples.

► (4.13) EXAMPLES

1) If M is the machine in Example (4.2), then we saw in Example (4.6.1) that Π consists of the two sets $A = \{\sigma_0\}$ and $B = \{\sigma_1, \sigma_2\}$. Using the digraph from (4.2), it is easy to construct the digraph of \widetilde{M} as in Figure 4.4. Note that \widetilde{M} looks exactly like the machine M' of Example (4.2).

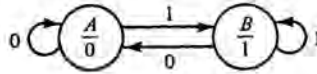


Figure 4.4

2) Let M be the machine of Example (4.11). Then the partition Π consists of the three sets $A = \{\sigma_0\}$, $B = \{\sigma_1, \sigma_2, \sigma_5\}$, and $C = \{\sigma_3, \sigma_4\}$. Using Table (4.12), it is not hard to see that the digraph for the quotient machine \widetilde{M} is as shown in Figure 4.5. ◀

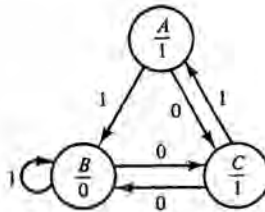


Figure 4.5

Below, we will prove that this machine \widetilde{M} is minimal and is equivalent to M . This means, in particular, that if M starts in σ_0 and if \widetilde{M} starts in A , then the two machines will produce identical outputs when fed the same input string. (Try an example to convince yourself.) Thus we have constructed a machine with three states that can perform the same job

as the machine M , which has six states. Furthermore, the minimality of \widetilde{M} means that no machine with only two states can possibly have this property.

Now we will establish the two results needed to complete our program.

(4.14) PROPOSITION. $\widetilde{M} = (\Pi, X, Z, \tilde{\tau}, \tilde{\omega})$, the quotient of the machine $M = (S, X, Z, \tau, \omega)$ by E , is equivalent to M .

PROOF. We must show $M \geq \widetilde{M}$ and $\widetilde{M} \geq M$. Suppose $\Pi = \{[\sigma_1], [\sigma_2], \dots, [\sigma_n]\}$, where $\sigma_i \in S$ for $i = 1, 2, \dots, n$ and where $[\sigma]$ denotes the equivalence class of σ under E . To show $M \geq \widetilde{M}$ define $\varphi : \Pi \rightarrow S$ by $\varphi([\sigma_i]) = \sigma_i$. (Here, we make a choice of representatives for the equivalence classes of Π . Different choices will change φ , but the rest of the proof will be unchanged.) We will use induction on the length of an input string to see that, for all $i = 1, 2, \dots, n$, we have $\tilde{\omega}^*([\sigma_i], \hat{x}) = \omega^*(\sigma_i, \hat{x})$ for all $\hat{x} \in X^*$. From the definition of $\tilde{\omega}$, it follows that $\tilde{\omega}([\sigma_i]) = \omega(\sigma_i)$. Now suppose that for all $i = 1, 2, \dots, n$ and for all input sequences \hat{x} of length k , we have that $\tilde{\omega}^*([\sigma_i], \hat{x}) = \omega^*(\sigma_i, \hat{x})$ and suppose $\hat{x} = x_1 x_2 \cdots x_{k+1}$. Then

$$\begin{aligned}\tilde{\omega}^*([\sigma_i], \hat{x}) &= \tilde{\omega}^*(\tilde{\tau}([\sigma_i], x_1))\tilde{\omega}^*(\tilde{\tau}([\sigma_i], x_1), x_2 \cdots x_{k+1}) \\ &= \omega^*(\tau(\sigma_i, x_1))\omega^*(\tau(\sigma_i, x_1), x_2 \cdots x_{k+1}) \\ &= \omega^*(\sigma_i, \hat{x}),\end{aligned}$$

where we have applied our induction hypothesis to the state $\tau([\sigma_i], x_1) (= [\sigma_j]$ for some $j, 1 \leq j \leq n$) and the input sequence $x_2 \cdots x_{k+1}$.

To prove $\widetilde{M} \geq M$, define $\psi : S \rightarrow \Pi$ by $\psi(\sigma) = [\sigma]$. The remainder of the proof is similar to the first part, and we leave the details to the reader as an exercise. ■

Finally, we prove

(4.15) PROPOSITION. With notation as in Proposition (4.14), \widetilde{M} is a minimal finite-state machine.

PROOF. We must show that \widetilde{M} satisfies the property in Definition (4.4). So suppose $M_0 \geq \widetilde{M}$. Then there exists a mapping $\varphi : \Pi \rightarrow S_0$ such that $\tilde{\omega}^*([\sigma_i], \hat{x}) = \omega_0^*(\varphi([\sigma_i]), \hat{x})$ for every $i = 1, 2, \dots, n$ and every $\hat{x} \in X^*$. To show that M_0 has at least as many states as \widetilde{M} , it suffices to show that φ is a one-to-one mapping. Suppose $\varphi([\sigma_i]) = \varphi([\sigma_j])$. Then, if $\hat{x} \in X^*$ we have

$$\begin{aligned}\omega^*(\sigma_i, \hat{x}) &= \tilde{\omega}^*([\sigma_i], \hat{x}) = \omega_0^*(\varphi([\sigma_i]), \hat{x}) \\ &= \omega_0^*(\varphi([\sigma_j]), \hat{x}) = \tilde{\omega}^*([\sigma_j], \hat{x}) = \omega^*(\sigma_j, \hat{x}).\end{aligned}\tag{62}$$

But this implies $\sigma_i E \sigma_j$, so we have $[\sigma_i] = [\sigma_j]$. Therefore, φ is one-to-one and the proof is complete. ■

EXERCISES

- 4.1. Show that the relation \geq of Definition (4.1) is reflexive and transitive.
- 4.2. Show that the relations E_k and E of Definition (4.5) are equivalence relations.
- † 4.3. If the same output is associated to every state of a Moore machine M , what is a minimal machine equivalent to M ?
- † 4.4. Construct a minimal Moore machine that has three states and is different from the machine in Example (4.13.2).
- † 4.5. In each part, find the minimal quotient machine \widetilde{M} that is equivalent to the Moore machine M with the given defining table.

a)

Present State	Next State		Output
	0	1	
σ_0	σ_3	σ_0	0
σ_1	σ_2	σ_2	1
σ_2	σ_0	σ_3	1
σ_3	σ_0	σ_2	0

b)

Present State	Next State		Output
	0	1	
σ_0	σ_3	σ_1	0
σ_1	σ_3	σ_2	1
σ_2	σ_0	σ_1	1
σ_3	σ_0	σ_2	0

c)

Present State	Next State		Output
	0	1	
σ_0	σ_3	σ_0	0
σ_1	σ_2	σ_2	1
σ_2	σ_0	σ_3	1
σ_3	σ_0	σ_3	0

fe

d)

Present State	Next State		Output
	0	1	
σ_0	σ_3	σ_6	0
σ_1	σ_4	σ_2	1
σ_2	σ_4	σ_1	1
σ_3	σ_2	σ_0	0
σ_4	σ_2	σ_0	0
σ_5	σ_3	σ_5	1
σ_6	σ_4	σ_2	0

e)

Present State	Next State		Output
	0	1	
σ_0	σ_3	σ_6	0
σ_1	σ_4	σ_2	1
σ_2	σ_6	σ_1	1
σ_3	σ_2	σ_0	0
σ_4	σ_2	σ_0	0
σ_5	σ_3	σ_5	1
σ_6	σ_4	σ_2	0

4.6. Complete the proof of Proposition (4.14).

- † 4.7. Suppose M_1 and M_2 are minimal and equivalent. What can you conclude?
- † 4.8. Suppose M_1 is equivalent to a minimal machine with two states and M_2 is equivalent to a minimal machine with three states. Can M_1 be equivalent to M_2 ?

CHAPTER III

Semigroups, Monoids, and Groups

In this chapter, we will study binary algebras, which consist of a set together with a binary operation on that set. Semigroups, monoids, and groups are binary algebras that satisfy certain axioms. We will see that semigroups are very closely related to finite-state machines. We will also see that groups arise in many contexts, and we will study an application of groups to an area of combinatorics called Pólya Enumeration Theory.

We begin this chapter with three examples that will motivate and illustrate the definitions in the second section. These examples may be viewed as “laboratories” in which one can do experiments to help to understand the concepts that arise later in the chapter.

1. Examples

- (1.1) **EXAMPLE:** S_3 , the Permutations of the Set $\{1, 2, 3\}$.

(1.2) **DEFINITION.** Let X be a set. A *permutation* of X is a 1-1 mapping from X onto itself. The set of all permutations of X will be denoted $\text{Perm}(X)$. The set of all permutations of a set of n objects (which are usually taken to be the integers $1, 2, \dots, n$) is denoted S_n .

How many elements are there in S_n ? If $\sigma \in S_n$, then there are n possible choices for $\sigma(1)$, but then only $n-1$ choices for $\sigma(2)$ since σ is a 1-1 mapping. Then there are $n-2$ choices for $\sigma(3)$ and so on until there is only one possibility for $\sigma(n)$. Hence we have

(1.3) PROPOSITION. The set S_n contains exactly $n!$ elements.

There is an obvious binary operation that one may consider on the set $\text{Perm}(X)$, namely composition of mappings. It is easy to see that if σ and τ are two 1-1 mappings from X onto itself, then the composition $\sigma \circ \tau$ is again a 1-1 mapping from X onto itself.

We will consider in detail here the set S_3 together with the operation composition. First, we introduce a shorthand notation for a permutation of the set $\{1, 2, 3\}$.

(1.4) NOTATION. We will let

$$\begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{pmatrix}$$

denote the permutation $\sigma \in S_n$.

Let

$$\begin{aligned} \epsilon &= \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \\ \sigma &= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \\ \tau &= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}. \end{aligned}$$

Then, recalling that $\sigma \circ \tau(i) = \sigma(\tau(i))$, we have

$$\sigma \circ \tau = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}.$$

Note that composition is not a commutative operation on S_3 . In particular,

$$\tau \circ \sigma = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}.$$

We will now write $\sigma\tau$ instead of $\sigma \circ \tau$, and we will speak of the operation in S_3 as “product” or “multiplication.” (We note that some authors prefer to define $\sigma\tau$ to mean the permutation one gets by first performing σ and then performing τ ; but remember that our operation is composition of mappings in the usual way one learns in more elementary courses.) We’ve seen above

five of the six elements in S_3 , namely $\epsilon, \sigma, \tau, \sigma\tau$, and $\tau\sigma$. The sixth element of S_3 is $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$, and this is equal to $\sigma^2 (= \sigma\sigma)$. It is then not hard to compute the entries in the following “multiplication table” for S_3 . (In this table, the product $\mu\nu$ occurs in the row corresponding to μ and the column corresponding to ν .)

(1.5) TABLE

	ϵ	σ	τ	σ^2	$\sigma\tau$	$\tau\sigma$
ϵ	ϵ	σ	τ	σ^2	$\sigma\tau$	$\tau\sigma$
σ	σ	σ^2	$\sigma\tau$	ϵ	$\tau\sigma$	τ
τ	τ	$\tau\sigma$	ϵ	$\sigma\tau$	σ^2	σ
σ^2	σ^2	ϵ	$\tau\sigma$	σ	τ	$\sigma\tau$
$\sigma\tau$	$\sigma\tau$	τ	σ	$\tau\sigma$	ϵ	σ^2
$\tau\sigma$	$\tau\sigma$	$\sigma\tau$	σ^2	τ	σ	ϵ

We note that the binary algebra (S_3, \circ) satisfies the following properties:

- 1) $\mu\nu \in S_n$ for all $\mu, \nu \in S_3$. (This just says that composition is a binary operation on S_3 .)
- 2) $\mu(\nu\rho) = (\mu\nu)\rho$ for all $\mu, \nu, \rho \in S_3$. (This says that composition is an associative operation.)
- 3) $\mu\epsilon = \epsilon\mu = \mu$ for all $\mu \in S_3$.
- 4) For each $\mu \in S_3$, there exists an element $\theta \in S_3$ such that $\mu\theta = \theta\mu = \epsilon$.

Indeed, θ is just the inverse map μ^{-1} , which takes each element of $\{1, 2, 3\}$ “back to where it came from” under the map μ . For example, the inverse of $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ is $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$. (The inverse of μ can also be obtained by switching the two rows in the shorthand notation for μ . For example, the inverse of $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ can also be written as $\begin{pmatrix} 2 & 3 & 1 \\ 1 & 2 & 3 \end{pmatrix}$ — this notation still represents the permutation that takes 2 to 1, 3 to 2, and 1 to 3.)

These four properties exactly say that S_3 with the operation \circ is a group. We will see later in this chapter that the permutation groups S_n , $n = 2, 3, \dots$, play an important role in the theory and applications of groups. The permutation group S_n is also called the *symmetric group on n letters*.

► (1.6) **EXAMPLE:** D_4 , the Symmetries of a Square.

Although the study of groups usually seems quite abstract at first, much of the motivation for groups comes from geometric problems, especially problems concerning symmetries of geometric figures. By a symmetry of a square we mean, roughly speaking, a rigid motion of the square in 3-space so that the square covers the same region at the end of the motion as it did at the start. More precisely, if we take axes for 3-space with the origin at the center of the square, then a symmetry is a linear transformation of 3-space that carries the square onto itself. Such a linear transformation must be a rotation, or a reflection about some line, or a composition of rotations and reflections. *We will assume all rotations below to be in the counterclockwise direction.*

(1.7) **DEFINITION.** Let D_4 denote the set of symmetries of a square. More generally, for $n \geq 3$, let D_n denote the set of symmetries of a regular plane n -gon (i.e., a polygon in the plane with n equal sides and each exterior angle equal to $360^\circ/n$).

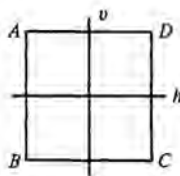


Figure 1.1

For the square in Figure 1.1, let α denote the symmetry given by rotating the square (about its center) by 90° and let β denote the symmetry given by reflection about the line h . Again, composition of mappings is an obvious operation to be considered on D_4 and again we will write the operation as “multiplication.” The set D_4 then has the following eight elements:

- ϵ = the identity mapping
- α = rotation by 90°
- α^2 = rotation by 180°
- α^3 = rotation by 270°
- β = reflection about the line h
- $\alpha\beta$ = reflection about the diagonal BD
- $\alpha^2\beta$ = reflection about the line v
- $\alpha^3\beta$ = reflection about the diagonal AC .

It is not too hard to determine the following multiplication table for D_4 .

(1.8) TABLE

	ϵ	α	α^2	α^3	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$
ϵ	ϵ	α	α^2	α^3	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$
α	α	α^2	α^3	ϵ	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$	β
α^2	α^2	α^3	ϵ	α	$\alpha^2\beta$	$\alpha^3\beta$	β	$\alpha\beta$
α^3	α^3	ϵ	α	α^2	$\alpha^3\beta$	β	$\alpha\beta$	$\alpha^2\beta$
β	β	$\alpha^3\beta$	$\alpha^2\beta$	$\alpha\beta$	ϵ	α^3	α^2	α
$\alpha\beta$	$\alpha\beta$	β	$\alpha^3\beta$	$\alpha^2\beta$	α	ϵ	α^3	α^2
$\alpha^2\beta$	$\alpha^2\beta$	$\alpha\beta$	β	$\alpha^3\beta$	α^2	α	ϵ	α^3
$\alpha^3\beta$	$\alpha^3\beta$	$\alpha^2\beta$	$\alpha\beta$	β	α^3	α^2	α	ϵ

One may check that D_4 with composition satisfies the same four properties as S_3 did, hence that D_4 is a group. It is called the *dihedral group of order 8*. We note that each reflection is its own inverse, as is rotation by 180° ; rotation by 90° and rotation by 270° are inverse to each other. We will study D_n , the dihedral group of order $2n$, further in §8. ◀

Note that if we label the four vertices of our square 1, 2, 3, and 4, as in Figure 1.2, then we may represent a symmetry of the square by an element of S_4 by considering the final position of each vertex after applying the symmetry. For example, (counterclockwise) rotation by 90° will take vertex 1 to position 2, vertex 2 to position 3, etc. Hence we may represent α by the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$. The reader may check that under this correspondence the product of two symmetries will be represented by the product of the two corresponding permutations. It is often useful to represent the elements of a group by permutations on some set and, as we will see, this is always possible.

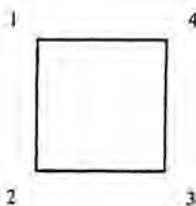


Figure 1.2

Note here that not every permutation in S_4 represents a symmetry of the square; indeed, S_4 has $4! = 24$ elements and D_4 has only 8 elements. For example, the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 3 & 1 \end{pmatrix}$ does not represent a symmetry of the square.

► (1.9) **EXAMPLE:** X^+ , Finite Input Strings.

Let X denote a finite set; for example, X might be the input alphabet of a finite-state machine $M = (S, X, Z, \tau, \omega)$. As in I.3, we make the following definition.

(1.10) **DEFINITION.** Let X^+ denote the set of all possible finite strings (or “words”) $\hat{x} = x_1x_2 \cdots x_n$, where $n \in \mathbf{P}$ and $x_i \in X$ for $i = 1, 2, \dots, n$. Let $\epsilon = “ ”$ denote the empty string and put $X^* = X^+ \cup \{\epsilon\}$.

The operation we wish to consider on X^+ and on X^* is “concatenation” (linking) of strings. For example, suppose $X = \{0, 1\}$ and $\hat{x} = 011$ and $\hat{y} = 1010$ are two input strings. Then $\hat{x}\hat{y}$ is the string 0111010.

The set X^+ with the operation concatenation satisfies the two properties:

- 1) If $\hat{x}, \hat{y} \in X^+$, then $\hat{x}\hat{y} \in X^+$.
- 2) If $\hat{x}, \hat{y}, \hat{z} \in X^+$, then $\hat{x}(\hat{y}\hat{z}) = (\hat{x}\hat{y})\hat{z}$.

These properties are the same first two properties that were satisfied by S_3 and D_4 . However, X^+ does not satisfy the third and fourth properties that were satisfied by S_3 and D_4 . Indeed, to satisfy the “identity” property (i.e., there exists ϵ such that $\hat{x}\epsilon = \epsilon\hat{x} = \hat{x}$), we must include the empty string. Thus the set X^* with the operation concatenation satisfies the same first three properties that were satisfied by S_3 and D_4 . But certainly there are no “inverse” strings in X^* — there is no way to link any string to a nonempty string and wind up with the empty string. Thus X^* does not satisfy the fourth property that was satisfied by S_3 and D_4 . Note that we cannot give a multiplication table for X^+ since it is an infinite set (each string is of finite length, but the strings may be arbitrarily long). ◀

EXERCISES

† 1.1. In S_3 , for each permutation μ find the least positive integer r such that $\mu^r = \epsilon$.

70

† 1.2. a) In S_3 , show that the set $\{\epsilon, \sigma, \sigma^2\}$ is closed under multiplication

- (i.e., that the product of any two elements in this set is again in this set).
- b) Find three subsets of S_3 , each containing two elements, that are closed under multiplication.
 - c) Show that the subset $\{\epsilon, \sigma\tau, \tau\sigma\}$ is not closed under multiplication.
- 1.3. Verify that $\mu(\nu\theta) = (\mu\nu)\theta$ for various choices of μ, ν , and $\theta \in S_3$.
- † 1.4. In D_4 , for each symmetry γ , find the least positive integer r such that $\gamma^r = \epsilon$.
- † 1.5. a) In D_4 , show that the set $\{\epsilon, \alpha^2, \beta, \alpha^2\beta\}$ is closed under multiplication.
- b) Find another subset of D_4 containing four elements that is closed under multiplication.
- 1.6. a) Find the permutation in S_4 that corresponds to each symmetry in D_4 .
- b) Verify that the inverse of a symmetry γ in D_4 is represented by the inverse of the permutation in S_4 corresponding to γ .
- 1.7. D_3 , the group of symmetries of an equilateral triangle consists of six elements – the identity, rotations by 120° and 240° , and reflections about each of the three medians. Set up a 1-1 correspondence between the symmetries of an equilateral triangle and the elements of S_3 such that the permutation representing the product of two symmetries γ and δ is the product of the permutations representing γ and δ .
- † 1.8. If X is an input alphabet containing n “letters,” how many “words” will there be of length r ?
- † 1.9. Suppose M is a finite-state machine with input alphabet $X = \{0, 1\}$. Call two input strings in X^+ “equivalent” if they both contain an even number of 1’s or if they both contain an odd number of 1’s (i.e., if they have the same parity).
- a) Show that this is an equivalence relation on X^+ .
 - b) Show that if the input strings \hat{x}_1 and \hat{x}_2 are equivalent and the input strings \hat{y}_1 and \hat{y}_2 are equivalent, then $\hat{x}_1\hat{y}_1$ and $\hat{x}_2\hat{y}_2$ are equivalent.
 - c) If M is the parity check machine of I.3.3, is it true that two equivalent input strings give rise to equivalent output strings, where “equivalence” for output strings means the same as above?

2. The Group Axioms and More Examples

We now present the main definitions of this chapter.

(2.1) DEFINITIONS. A set A with a binary operation \circ is called a *semigroup* if:

- 1) (Associativity) $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in A$.

A semigroup (A, \circ) is called a *monoid* if it further satisfies

- 2) (Identity) There exists an element $e \in A$ such that $a \circ e = e \circ a = a$ for all $a \in A$. The element e is called an *identity* for \circ .

A monoid (A, \circ) is called a *group* if it further satisfies

- 3) (Inverse) For every $a \in A$ there exists an element $a' \in A$ such that $a \circ a' = a' \circ a = e$. The element a' is called an *inverse* of a (with respect to \circ).

If the operation \circ is commutative, then we call A a *commutative* semigroup, monoid, or group (whichever is appropriate). A commutative group is also called an *abelian* group, after the Norwegian mathematician Niels Henrik Abel (1802–1829).

By an argument using mathematical induction and the associative property, it can be shown that a finite “product” in a semigroup may be written unambiguously as

$$a_1 \circ a_2 \circ \cdots \circ a_n$$

(i.e., no parentheses are necessary).

We refer to (1)–(3) above as the *group axioms*. Often, the fact that A is closed under the operation \circ (meaning that if $a, b \in A$, then $a \circ b \in A$) is also considered a group axiom, but this closure property is included in our definition of what it means for \circ to be a “binary operation on A .” One of our goals is to derive elementary properties of groups from these axioms. The power of the “axiomatic method” is that these properties will then hold in any particular example of a group. But first, we look at some more examples.

► (2.2) EXAMPLES

a) $(\mathbf{Z}, +)$ is a group.

b) (\mathbf{Z}, \cdot) is a monoid but not a group. Indeed, the only integers that have a multiplicative inverse in \mathbf{Z} are 1 and -1 .

c) The binary algebra $(X^+, \text{concatenation})$ of Example (1.9) is a semigroup but not a monoid. The binary algebra $(X^*, \text{concatenation})$ is a monoid but not a group.

d) $(\mathbf{P}, +)$ is a semigroup but not a monoid.

e) (\mathbf{P}, \cdot) is a monoid but not a group.

f) Put $\mathbf{Z}_n = \{0, 1, 2, \dots, n-1\}$, where n is an integer greater than 1. Define addition on \mathbf{Z}_n by

$$a +_n b = \begin{cases} a + b & \text{if } a + b < n \\ (a + b) - n & \text{if } a + b \geq n. \end{cases}$$

Then $(\mathbf{Z}_n, +_n)$ is a group. (The associativity is not so easy to verify, but the theory to be developed in §5 will show that this is a group — cf. Example (5.15).) This group is called the group of integers modulo n . Another way to define $+_n$ is to say $a +_n b$ is the remainder when $a + b$ (the sum in \mathbf{Z}) is divided by n .

g) Define multiplication on \mathbf{Z}_n by $a \cdot_n b =$ the remainder when $a \cdot b$ (the product in \mathbf{Z}) is divided by n . Then (\mathbf{Z}_n, \cdot_n) is a monoid, and we will see in Chapter IV that $(\mathbf{Z}_n - \{0\}, \cdot_n)$ is a group if n is prime.

h) $(\mathbf{Q} - \{0\}, \cdot)$ is a group.

i) Let $M(2, \mathbf{R})$ denote the set of all 2×2 real matrices. If “ \cdot ” denotes matrix multiplication, then $(M(2, \mathbf{R}), \cdot)$ is a monoid but not a group. The subset $GL(2, \mathbf{R})$ consisting of all nonsingular (determinant $\neq 0$) 2×2 real matrices does form a group under matrix multiplication.

j) Suppose Y is a set. Let Y^Y denote the set of all mappings $f: Y \rightarrow Y$ and let \circ denote composition of mappings. Then (Y^Y, \circ) is a monoid (the identity is the identity map), but it is not a group (a mapping that is not 1-1 cannot have an inverse). ◀

More examples will be examined in the exercises.

(2.3) NOTATION. As is customary, we will generally say “let A be a semigroup” (or monoid or group) instead of referring to both the set A and the binary operation \circ . Unless otherwise specified, we will treat the binary operation on A as “product” or “multiplication” and we will write, for example, ab instead of $a \circ b$.

Notice that in the Identity Axiom we did not assume that there is a *unique* identity element. We will now prove that this is in fact true. One might then say, “Why not assume there is a unique identity in the axiom?” The reason is that we want to assume as little as possible in our axioms, and anything that follows from the axioms should be proved and not assumed in the axioms. In fact, if we required in the Identity Axiom that there should be a unique identity, then every time we wanted to verify that a binary algebra was a group, we would have to produce an identity element *and* prove that it was the unique identity element.

(2.4) LEMMA. Suppose that A is a monoid (or group, of course). Then there is a unique identity element in A .

PROOF. To prove uniqueness of an object with a certain property, we invariably use the following strategy: Assume there are two objects having the given property and then show that these objects are equal. So, suppose that e and e' are two identities for the binary operation on A . Then

$$e = ee' = e',$$

where the first equality holds because e' is an identity and the second equality holds because e is an identity. ■

Similarly, if an element has an inverse then it has a unique inverse.

(2.5) LEMMA. Suppose A is a monoid. If an element $a \in A$ has an inverse with respect to the operation on A , then it has a unique inverse.

PROOF. Exercise 2.4.

(2.6) NOTATION. Suppose A is a monoid. If an element $a \in A$ has an inverse, then we will denote that inverse by a^{-1} , except when the operation on A is addition, in which case the inverse of a will be denoted $-a$.

An important consequence of the existence of an inverse for each element in a group is:

(2.7) PROPOSITION. (Cancellation laws) Suppose G is a group and $a, b, c \in G$. Then

- 1) If $ab = ac$, then $b = c$.
- 2) If $ac = bc$, then $a = b$.

PROOF. To prove (1), multiply both sides on the left by a^{-1} . To prove (2), multiply both sides on the right by c^{-1} . ■

We note that if G is not a commutative group, then we could have $ab = ca$ without having $b = c$; indeed, in D_4 we have $\alpha(\alpha^2\beta) = \beta\alpha$, but $\alpha^2\beta \neq \beta$.

It follows from these cancellation laws that in the multiplication table for a finite group no element can occur more than once in any row or in any column. For example, if the same element occurred in the columns corresponding to g' and g'' in the row corresponding to g , then we would have

$gg' = gg''$, which implies $g' = g''$. Since the number of rows and the number of columns equals the number of elements in the group, we have

(2.8) PROPOSITION. If G is a finite group, then in the multiplication table for G , each element occurs exactly once in every row and every column.

We note that the above cancellation laws can still hold in a semigroup that is not a monoid (so one cannot even speak of “inverses”). Indeed, the semigroup X^+ of Example (1.9) serves as an example. However, in most finite semigroups that are not monoids that we will study, one or both cancellation laws will fail, and therefore elements of the semigroup may appear more than once in a given row or column of the multiplication table for that semigroup.

We define exponents in a (multiplicative) binary algebra as follows.

(2.9) DEFINITION. Suppose A is a semigroup and $a \in A$.

- 1) For $n \in \mathbf{P}$ we define a^n to be the product of a with itself n times.
- 2) If A is a monoid with identity e , then we define $a^0 = e$.
- 3) If a^{-1} exists (so, for example, if A is a group), then for $n \in \mathbf{P}$, we define a^{-n} to be $(a^{-1})^n$.

For example, in D_4 we have $\alpha^{-2} = (\alpha^{-1})^2 = (\alpha^3)^2 = \alpha^2$. It is not too hard to show that we have the familiar laws of exponents

(2.10) PROPOSITION

$$a^m a^n = a^{m+n} \text{ and } (a^m)^n = a^{mn}$$

for all m and n such that these expressions are defined.

(2.11) PROPOSITION. Suppose that a^{-1} and b^{-1} exist. Then

- 1) $(ab)^{-1} = b^{-1}a^{-1}$.
- 2) $(a^{-1})^{-1} = a$.

PROOF. To prove that one element is the inverse of another element we just need to take the product of the elements (in both orders if the operation on A is not commutative) and see that we get the identity. In (1),

$$(ab)(b^{-1}a^{-1}) = a(bb^{-1})a^{-1} = aea^{-1} = aa^{-1} = e$$

and

$$(b^{-1}a^{-1})(ab) = b^{-1}(a^{-1}a)b = b^{-1}eb = b^{-1}b = e$$

and we are done.

For (2), we just must note that the equalities

$$aa^{-1} = a^{-1}a = e$$

not only show that a^{-1} is the inverse of a , but also that a is the inverse of a^{-1} ; that is, $a = (a^{-1})^{-1}$. ■

Unless the operation on A is commutative, it may *not* be true that $(ab)^n = a^n b^n$ for all n and all $a, b \in A$. For example, $(ab)^2 = (ab)(ab)$, while $a^2 b^2 = aabb$, and these will not be equal for some $a, b \in A$ if A is a group and the operation on A is not commutative (see Exercise 2.7). Of course, if the operation on A is commutative, then $(ab)^n$ will equal $a^n b^n$, whenever these expressions are defined.

We note that in an additive binary algebra, the expression a^n will be replaced by na . So in an additive binary algebra, the "laws of exponents" (Proposition (2.10)) become the "laws of coefficients": $ma + na = (m+n)a$ and $(mn)a = n(ma)$.

EXERCISES

- † 2.1. Determine if each of the following binary algebras is a semigroup, monoid, or group.
- The set of even integers under addition.
 - The set of even integers under multiplication.
 - The set of odd integers under multiplication.
 - The set of 2×2 real matrices with first row consisting of two zeros, under matrix multiplication.
 - The set of integers under subtraction.

- 2.2. Here is the multiplication table for a semigroup consisting of two elements:

	a_1	a_2
a_1	a_1	a_2
a_2	a_1	a_2

Produce at least three different multiplication tables for semigroups with two elements, including one that is the multiplication table for a group with two elements.

† 2.3. How can you decide whether a finite group is abelian from the multiplication table for the group?

† 2.4. Prove Lemma 2.5.

† 2.5. Suppose $G = \{e, a, b\}$ is a group. Construct a multiplication table for this group. Is this an abelian group? What is a^2 ? a^3 ? a^{-1} ?

2.6. Determine a multiplication table for a group with four elements e, a, b, c such that the square of each element is e . Is this an abelian group? This group is called the *Klein four-group*.

† 2.7. Suppose that G is a group. Show that if $(ab)^2 = a^2b^2$ for all $a, b \in G$, then G is abelian.

† 2.8. Determine the addition table for the group $(\mathbb{Z}_4, +_4)$. What is -1 in this group?

† 2.9. Determine the multiplication table for the monoid (\mathbb{Z}_4, \cdot_4) . What is 2^2 in this monoid?

2.10. Let S denote the set of all mappings from the set $\{1, 2\}$ to itself. (S then consists of four elements.) Determine the multiplication table for (S, \circ) , where \circ denotes composition.

2.11. In the group S_3 of Example (1.1), find $(\sigma\tau)^{-1}$, $\sigma^{-1}\tau^{-1}$, $\tau^{-1}\sigma^{-1}$, $(\sigma\tau)^2$, $\sigma^2\tau^2$.

2.12. In the group D_4 of Example (1.6), find $(\alpha\beta)^{-1}$, $\alpha^{-1}\beta^{-1}$, $\beta^{-1}\alpha^{-1}$, $(\alpha\beta)^2$, $\alpha^2\beta^2$.

2.13. Let G be a group and suppose $g_1, g_2, \dots, g_n \in G$. Show that

$$(g_1g_2 \cdots g_n)^{-1} = g_n^{-1}g_{n-1}^{-1} \cdots g_1^{-1}.$$

† 2.14. Suppose A is a monoid, $a \in A$, and $a^{-1} \in A$. Show that if either $ab = e$ or $ba = e$, then $b = a^{-1}$.

† 2.15. Suppose that G is a group such that $g^2 = e$ for all $g \in G$. Show that G is abelian. (Hint: If $g^2 = e$, then what is g^{-1} ? Now use Proposition (2.11).)

2.16. Suppose (G, \circ) and $(G', *)$ are groups. Define a product on the Cartesian product $G \times G'$ by

$$(a, a')(b, b') = (a \circ b, a' * b').$$

Show that $G \times G'$ is a group under this product. The group $G \times G'$ is called the *direct product* of G and G' . More generally, given groups G_1, G_2, \dots, G_m , the direct product

$$G_1 \times G_2 \times \cdots \times G_m$$

is the group obtained by performing coordinatewise "multiplication" on ordered m -tuples in the Cartesian product.

- 2.17.** Suppose S is a semigroup. An element $e_L \in S$ (resp. $e_R \in S$) is called a *left (resp. right) identity* if $e_L s = s$ (resp. $se_R = s$) for all $s \in S$. Show that if S has a left identity e_L and a right identity e_R , then $e_L = e_R$ and that this element is an identity. Conclude that if S has two distinct left (or right) identities, then S is not a monoid.
- 2.18.** Let S be the set of 2×2 real matrices with the second row consisting of zeros. Find all left identities in (S, \cdot) .
- 2.19.** Suppose that S is a semigroup in which the left and right cancellation laws hold. Show that if S is a finite set, then S is a group. (Hint: Suppose the distinct elements of S are s_1, s_2, \dots, s_n . Consider the products $s_1 s_1, s_1 s_2, \dots, s_1 s_n$. By left cancellation, these elements are distinct, hence, by the Pigeonhole Principle (I.1.9) one of them, say $s_1 s_i$ must be s_1 . Show that s_i must be an identity. Then note that $s_1 s_j$, for some j , must be s_i . Show that $s_j = s_1^{-1}$.)

3. Subobjects

(3.1) DEFINITION. Suppose (A, \circ) is a semigroup, monoid, or group. A nonempty subset $B \subseteq A$ is called a *subsemigroup*, *submonoid*, or *subgroup*, respectively, of A if B together with the restriction of \circ to B forms a semigroup, monoid, or group, respectively.

(3.2) PROPOSITION. Suppose (A, \circ) is a semigroup and B is a nonempty subset of A . Then B is a subsemigroup of A if B is closed under \circ (i.e., if $b_1 \circ b_2 \in B$ for all $b_1, b_2 \in B$).

PROOF. The fact that B is closed under \circ means that the restriction of \circ to B does indeed define a binary operation on B . The only thing to be shown, then, is that the restriction of \circ to B is associative. But since \circ is associative on all of A , it is *a fortiori* (even more obviously) associative when restricted to B . ■

► (3.3) EXAMPLES

1) Let T denote the set of all 2×2 real matrices with the second column consisting of zeros. Then T is a subsemigroup of $(M(2, \mathbf{R}), \cdot)$. To prove this, we need to see that the product of every pair of elements in T is again in T . Now,

$$\begin{bmatrix} a & 0 \\ c & 0 \end{bmatrix} \cdot \begin{bmatrix} a' & 0 \\ c' & 0 \end{bmatrix} = \begin{bmatrix} aa' & 0 \\ cc' & 0 \end{bmatrix},$$

which is again in T .

2) Let U denote the set of all 2×2 real matrices with 0 as (1,1) entry (i.e., the entry in the first row and first column). Then U is *not* a subsemigroup of $(M(2, \mathbf{R}), \cdot)$. To see this we may (and should) produce an explicit pair of elements in U whose product is not in U . One such example would be

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \blacktriangleleft$$

An important way to generate subsemigroups is to consider all powers of some fixed element.

(3.4) DEFINITION. Suppose S is a semigroup and $x \in S$. Then

$$\langle x \rangle = \{x, x^2, x^3, \dots\} = \{x^n : n \in \mathbf{P}\}$$

obviously is closed under the operation on S . It is called the *cyclic subsemigroup generated by x* . If $S = \langle x \rangle$ for some $x \in S$, then S is called a *cyclic semigroup* and x is called a *generator* for S .

► (3.5) EXAMPLES

1) $(\mathbf{P}, +)$ is a cyclic semigroup generated by 1.

2) (\mathbf{P}, \cdot) is not a cyclic semigroup. Here, $\langle 1 \rangle = \{1\}$ and $\langle 2 \rangle$ is the cyclic subsemigroup consisting of the integers of the form 2^n , for $n \in \mathbf{P}$.

3) Let $X = \{0, 1\}$ and let X^+ be the semigroup of Example (1.9). Then $\langle 1 \rangle$ consists of all strings containing only 1's, while $\langle 11 \rangle$ consists of all even length strings containing only 1's. ◀

Our attention will focus more on subgroups than on subsemigroups or submonoids. We begin the study of subgroups with the following lemma.

(3.6) LEMMA. Suppose G is a group with identity e and H is a subgroup of G . Then e is the identity of H .

PROOF. Let e_1 denote the identity of H . Then $e_1 e_1 = e_1$ and $e_1 e = e_1$ (the latter equation possibly being valid only in G , since we don't know that

$e \in H$ yet). Hence, we have $e_1 e_1 = e_1 e$. But then by cancellation in G we have $e_1 = e$. ■

One might have been tempted to prove Lemma (3.6) by saying $e = ee_1 = e_1$ as in the proof of Lemma (2.3). However, this would not be correct because we do not know that $e = ee_1$ until we know that e is in H .

While Lemma (3.6) might seem rather obvious, it is actually false if we replace "group" by "monoid," as the next example shows.

► (3.7) **EXAMPLE.** Let

$$N = \left\{ \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} : a \in \mathbf{R} \right\}.$$

Then it is easy to see that N is a submonoid of $(M(2, \mathbf{R}), \cdot)$ with the identity in N being the matrix $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, while the identity in $M(2, \mathbf{R})$ is of course $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. ◀

Before giving examples of subgroups, we will give a criterion for a nonempty subset of a group to be a subgroup.

(3.8) **PROPOSITION.** Suppose G is a group and H is a nonempty subset of G . Then H is a subgroup of G if and only if $ab^{-1} \in H$ for all $a, b \in H$.

PROOF. If H is a subgroup of G , then every element in H has an inverse in H and H is closed under the operation on G , so it is clear that if $a, b \in H$, then $ab^{-1} \in H$ also.

Conversely, assume that if $a, b \in H$, then $ab^{-1} \in H$. (Note that ab^{-1} is certainly in G ; the point is that this element of G is actually in H .) Suppose $h \in H$ (here, we use the assumption that H is nonempty). Then, by our hypothesis, $hh^{-1} = e$ belongs to H , hence the Identity Axiom is satisfied. Applying the hypothesis to e and h , we see that $eh^{-1} = h^{-1} \in H$, so the Inverse Axiom is satisfied. Now suppose that h_1 and $h_2 \in H$. Then applying the hypothesis to h_1 and h_2^{-1} , we see that $h_1(h_2^{-1})^{-1} = h_1 h_2 \in H$, so the binary operation on G restricts to H to give a binary operation on H (i.e., H is closed under the operation on G). Finally, since the binary operation on G is associative, it is associative when restricted to H . Therefore, H is a subgroup of G . ■

► **(3.9) DEFINITION and EXAMPLE.** Suppose G is a group and let

$$Z(G) = \{a \in G : ag = ga \text{ for all } g \in G\};$$

i.e., $Z(G)$ is the set of all elements of G that commute with every element of G . We call $Z(G)$ the *center* of G . Note that the identity e is always in the center. If G is abelian, then of course $Z(G) = G$. For a noncommutative group, it is possible that $Z(G)$ may contain only e . For an example of a noncommutative group in which the center is not just $\{e\}$, consider D_4 ; it is easy to see from Table (1.8) that α^2 commutes with every element of D_4 .

To show that the center is a subgroup of G , we start by supposing that g_1 and $g_2 \in Z(G)$. We must show that $g_1 g_2^{-1}$ commutes with every element $g \in G$. We have

$$\begin{aligned} g_1 g_2^{-1} g &= g_1 (g^{-1} g_2)^{-1} \quad \text{by Prop. (2.11)} \\ &= (g^{-1} g_2)^{-1} g_1 \quad \text{since } g_1 \in Z(G) \\ &= (g_2 g^{-1})^{-1} g_1 \quad \text{since } g_2 \in Z(G) \\ &= g g_2^{-1} g_1 \quad \text{by Prop. (2.11)} \\ &= g g_1 g_2^{-1} \quad \text{since } g_1 \in Z(G) \end{aligned}$$

and we are done. ◀

► **(3.10) EXAMPLE.** Let

$$H = \{\mu \in S_n : \mu(n) = n\}.$$

Then if $\mu, \nu \in H$, we clearly have $(\mu\nu^{-1})(n) = n$, so H is a subgroup of S_n . (H “looks like” S_{n-1} — we will make this precise when we discuss “isomorphic” groups.) ◀

Our focus will be mainly on finite groups in this book. When H is finite, the criterion of Proposition (3.8) can be further simplified — we just need to show that H is closed under the operation on G .

(3.11) PROPOSITION. Suppose G is a group and H is a finite, non-empty subset of G . If $ab \in H$ for every $a, b \in H$, then H is a subgroup of G .

PROOF. Denote the elements of H by h_1, h_2, \dots, h_n . Consider the products $h_1 h_1, h_1 h_2, \dots, h_1 h_n$. By cancellation in G , these products are distinct and by our hypothesis they are all in H . Hence they must be all the elements of H (by the Pigeonhole Principle (I.1.9)). Thus there exists j , $1 \leq j \leq n$, such

that $h_1 h_j = h_1$. But $h_1 e = h_1$ also, so by cancellation in G , $h_j = e$. Now there must exist k , $1 \leq k \leq n$, such that $h_1 h_k = h_j (= e)$. Then by Exercise 2.14, $h_k = h_1^{-1}$. By a similar argument, the inverse of each element of H will be in H . Thus H is closed under the operation on G , H satisfies the Identity and Inverse Axioms, and as we've seen before, the operation on G certainly remains associative when restricted to H . ■

As in the case of semigroups, we can generate subgroups by looking at all (now positive and nonpositive) powers of a fixed element.

(3.12) DEFINITION. Suppose G is a group and $g \in G$. Then it is easy to see that

$$\langle g \rangle = \{g^n : n \in \mathbb{Z}\}$$

satisfies the criterion in Proposition (3.8). It is called the *cyclic subgroup generated by g* . The group G is called a *cyclic group* if $G = \langle g \rangle$ for some $g \in G$ and, in this case, g is called a *generator* for G .

We recall that if our group is additive then we write ng instead of g^n .

► (3.13) EXAMPLES

1) $(\mathbb{Z}, +)$ is an infinite cyclic group generated by 1. Another generator is -1 . Note that if we view $(\mathbb{Z}, +)$ as a semigroup, then $\langle 1 \rangle = \mathbb{P}$; so here is an example that shows that $\langle a \rangle$ may not equal (a) . This example also shows that a cyclic group may not be a cyclic semigroup.

2) $(\mathbb{Z}_n, +_n)$ is cyclic with generator 1, but there are also other generators. We claim that $m \in \mathbb{Z}_n$ generates \mathbb{Z}_n if and only if the integers n and m are relatively prime. Indeed, suppose that $(n, m) = 1$. Then by (I.2.14) there exist integers r and s such that $1 = rn + sm$. Now, if $l \in \mathbb{Z}_n$, then $l = (rl)n + (sl)m$. But this means that in \mathbb{Z}_n we have $(sl)m = l$. Hence m generates \mathbb{Z}_n . On the other hand, if m and n are not relatively prime, then there is no way to write 1 in the form $rn + sm$ for any integers r and s (for then the greatest common divisor of n and m would divide 1). It follows that in \mathbb{Z}_n we never have $tm = 1$ for any $t \in \mathbb{Z}$.

For a concrete example, consider $(\mathbb{Z}_6, +_6)$. We have

$$\langle 0 \rangle = \{0\}$$

$$\langle 1 \rangle = \mathbb{Z}_6$$

$$\langle 2 \rangle = \{0, 2, 4\}$$

$$\langle 3 \rangle = \{0, 3\}$$

$$\langle 4 \rangle = \{0, 2, 4\}$$

$$\langle 5 \rangle = \mathbb{Z}_6.$$

3) The cyclic subgroups of S_3 are

$$\begin{aligned}(\epsilon) &= \{\epsilon\}, & (\sigma) &= (\sigma^2) = \{\epsilon, \sigma, \sigma^2\} \\ (\tau) &= \{\epsilon, \tau\}, & (\sigma\tau) &= \{\epsilon, \sigma\tau\}, & (\tau\sigma) &= \{\epsilon, \tau\sigma\}.\end{aligned}$$

Note that S_3 is therefore not a cyclic group. Some readers may wonder "Where are the other powers of σ ?" The point is that every power of σ is equal to one of ϵ, σ , or σ^2 . For example,

$$\sigma^{-5} = (\sigma^{-1})^5 = (\sigma^2)^5 = \sigma. \blacktriangleleft$$

(3.14) DEFINITION. The *order* of a finite group G , denoted $|G|$, is the number of elements in G .

(3.15) DEFINITION. Suppose G is a group and $g \in G$. The *order* of g , denoted $o(g)$, is the least positive integer m such that $g^m = e$, if such an integer exists. If no such m exists, then we say that g has *infinite order*.

► **(3.16) EXAMPLES**

- 1) The identity of a group always has order 1.
- 2) In S_3 , the element σ has order 3 and the element τ has order 2.
- 3) In $(\mathbb{Z}, +)$ every element except 0 has infinite order. ◀

Note that in a finite group G every element must have finite order. Indeed, if $g \in G$, then we must have $g^r = g^s$ for some integers r and s with, say, $r < s$. But then $g^{s-r} = e$, so $o(g) \leq s - r$.

(3.17) PROPOSITION. Suppose G is a finite group and $g \in G$. Then $o(g) = | \langle g \rangle |$.

PROOF. Suppose $m = o(g)$. We must show that $\langle g \rangle = \{e, g, g^2, \dots, g^{m-1}\}$, where these elements are distinct. First, note that if $k > m$, then by the Division Algorithm, we may write $k = qm + r$ with $0 \leq r < m$. Then

$$g^k = g^{qm+r} = (g^m)^q g^r = e g^r = g^r.$$

Finally, to see that the above m elements are distinct, suppose $g^i = g^j$ for $0 \leq i < j < m$. Then, multiplying both sides by g^{-i} , it follows that $g^{j-i} = e$. But this contradicts the fact that m is the least positive integer such that $g^m = e$. ■

We close this section with two easy results concerning cyclic groups.

(3.18) PROPOSITION. Every cyclic group is abelian.

PROOF. Suppose that G is a cyclic group with generator g . Then two arbitrary elements of G must be of the form g^i and g^j for some $i, j \in \mathbf{Z}$. But then $g^i g^j = g^{i+j} = g^j g^i$. ■

(3.19) PROPOSITION. Every subgroup of a cyclic group is cyclic.

PROOF. Suppose G is a cyclic group with generator g and let H be a subgroup of G . If $H = \{e\}$, then H is cyclic, so we may assume that H contains more than just e . Let k be the least positive integer such that $g^k \in H$. (Note that H cannot just contain negative powers of g since the inverse of any element in H is also in H .) We claim that $H = \langle g^k \rangle$. Indeed, suppose $g^i \in H$ (recall that every element in G is a power of g). By the Division Algorithm, we may write $i = qk + r$ with $0 \leq r < k$. Then $g^r = g^i (g^k)^{-q}$ is in H since $g^i, g^k \in H$ and H is a subgroup. But then $r = 0$ by our choice of k and so $g^i = (g^k)^q$. ■

EXERCISES

- 3.1. Let S be a semigroup. Show that if the intersection of two subsemigroups of S is nonempty, then that intersection is a subsemigroup of S .
- 3.2. Let T denote the subset of \mathbf{P} consisting of all integers of the form $2n + 3m$, where $n, m \in \mathbf{P}$. Show that T is a subsemigroup of $(\mathbf{P}, +)$. Also prove that T is not cyclic, thus demonstrating that a subsemigroup of a cyclic semigroup need not be cyclic.
- 3.3. Let S denote the set of all 2×2 real matrices with the second column consisting of zeros. Show that S is a subsemigroup of $(M(2, \mathbf{R}), \cdot)$.
- † 3.4. Let S be the semigroup consisting of all mappings from $\{0, 1\}$ to itself, with the operation being composition. There are four elements f_1, f_2, f_3, f_4 in S , where $f_1(0) = 0, f_1(1) = 1; f_2(0) = 1, f_2(1) = 0; f_3(0) = 0, f_3(1) = 0; \text{ and } f_4(0) = 1, f_4(1) = 1$. Determine a multiplication table for this semigroup and find all subsemigroups of S .
- † 3.5. Let G be a group. Show that the intersection of two subgroups of G is a subgroup of G .
- † 3.6. Prove or give a counterexample: The union of two subgroups of a group is always a subgroup. (Hint: Consider S_3 .)

- † 3.7. What do all the subgroups of $(\mathbf{Z}, +)$ look like?
- † 3.8. Let G be a group and let g be an element of (finite) order m . If $g^k = e$, then show that m divides k .
- † 3.9. Find all cyclic subgroups of D_4 .
- † 3.10. Let G be a cyclic group of order n with generator g (so $G = \{e, g, g^2, \dots, g^{n-1}\}$). For $m \in \mathbf{P}$, prove that G has a subgroup of order m if and only if m divides n .
- † 3.11. Let G be a group and let $g \in G$ be an element of order m . If t is relatively prime to m (see I.2), then show that g^t also has order m .
- 3.12. Let G be a group and let k and l be relatively prime integers (see I.2).
 a) Suppose $g \in G$. Show that if $g^k = e$ and $g^l = e$, then $g = e$.
 b) Suppose that $a, b \in G$ with $o(a) = k$ and $o(b) = l$. Show that $\langle a \rangle \cap \langle b \rangle = \{e\}$.
 c) With a and b as in part (b), suppose that $ab = ba$. Show that $o(ab) = kl$. (Hint: Show that $k|o(ab)$ and $l|o(ab)$.)
- † 3.13. Find the order of each element of $(\mathbf{Z}_{12}, +_{12})$. Which elements are generators of the group? List the distinct subgroups of \mathbf{Z}_{12} .
- 3.14. Suppose $G = \{e, g, g^2, \dots, g^{11}\}$ is a cyclic group of order 12. Find the smallest subgroup of G that contains g^4 and g^6 .
- 3.15. a) Prove that if g is a generator of a cyclic group G , then g^{-1} also generates G .
 b) Prove that if G is a cyclic group of order at least 3, then G has an even number of generators.
- 3.16. Let G be an abelian group. Show that if g and h are of finite order, then $o(gh)$ divides $o(g)o(h)$.
- 3.17. Find the center of D_4 .
- 3.18. Let G be a group. If $a \in G$, let $C(a) = \{g \in G : ga = ag\}$. Prove that $C(a)$ is a subgroup of G . ($C(a)$ is called the *centralizer* of a in G .)
- 3.19. If H is a subgroup of a group G , then show that $aHa^{-1} = \{aha^{-1} : h \in H\}$ is also a subgroup of G for any $a \in G$.
- 3.20. If H is a subgroup of G , then show that $N(H) = \{a \in G : aHa^{-1} = H\}$ is a subgroup of G . ($N(H)$ is called the *normalizer* of H in G .)

- 3.21.** Suppose that the only subgroups of a group G are $\{e\}$ and G . Prove that G is a finite group of order p , where p is a prime.
- 3.22.** Let S be a semigroup. An element $s \in S$ is called an *idempotent* if $s^2 = s$. Prove that a finite semigroup must have an idempotent. (Hint: Pick an element $x \in S$ and consider $\langle x \rangle$, the subsemigroup generated by x .)

4. Cosets and Lagrange's Theorem

Our focus for much of the remainder of this chapter will be the derivation of results about groups and applications by considering certain equivalence relations on semigroups and groups. The reader may wish to review the material on equivalence classes in Chapter II before proceeding.

Suppose G is a group and H is a subgroup of G .

(4.1) DEFINITION. Let $a, b \in G$. We say that a is *equivalent mod H* to b , denoted $a \equiv_H b$, if $a^{-1}b \in H$.

(4.2) LEMMA. Equivalence mod H is an equivalence relation.

PROOF. Suppose $a \in G$. Since H is a subgroup, $e = a^{-1}a \in H$. Thus $a \equiv_H a$. Now suppose $a \equiv_H b$. Then $a^{-1}b \in H$. Since H is a subgroup, $(a^{-1}b)^{-1} = b^{-1}a \in H$. Thus $b \equiv_H a$. Finally, suppose $a \equiv_H b$ and $b \equiv_H c$. Then $a^{-1}b \in H$ and $b^{-1}c \in H$. Since H is closed under the operation on G , we have $(a^{-1}b)(b^{-1}c) = a^{-1}c \in H$. Thus $a \equiv_H c$ and we are done. ■

What do the equivalence classes of this relation look like? If $a \in G$, then the equivalence class of a with respect to \equiv_H is

$$\{b \in G : a^{-1}b \in H\} = \{b \in G : b = ah \text{ for some } h \in H\}.$$

(4.3) DEFINITION. The set $aH = \{ah : h \in H\}$ is called a *left coset* of H in G .

So, the left cosets of H in G are exactly the equivalence classes of \equiv_H .

We note that we could define another equivalence relation by saying that a is equivalent to b if $ab^{-1} \in H$. The equivalence classes of this relation would be the sets $Ha = \{ha : h \in H\}$, which are called the *right cosets* of H in G .

► (4.4) EXAMPLES

1) Let H_1 be the subgroup $\{\epsilon, \sigma, \sigma^2\}$ of S_3 . Then the left cosets of H_1 in G are

$$\begin{aligned}\epsilon H_1 &= \sigma H_1 = \sigma^2 H_1 = \{\epsilon, \sigma, \sigma^2\} \\ \tau H_1 &= \sigma \tau H_1 = \tau \sigma H_1 = \{\tau, \tau \sigma, \sigma \tau\}.\end{aligned}$$

Note that $\mu H_1 = \nu H_1$ if and only if μ is equivalent mod H_1 to ν . This will always be the case since the cosets are the equivalence classes for the relation "equivalence mod H ."

The right cosets of H_1 in G are the same as the left cosets; i.e., the distinct right cosets are H_1 and $\{\tau, \tau \sigma, \sigma \tau\}$.

2) Let H_2 be the subgroup $\{\epsilon, \tau\}$ of S_3 . One left coset, as always, will be H_2 itself. To find a second left coset, we must take an element not in H_2 , say σ , and form the coset σH_2 . We have $\sigma H_2 = \{\sigma, \sigma \tau\}$. To find another left coset, we take an element, say σ^2 , that has not appeared in the cosets we have found so far. Then $\sigma^2 H_2 = \{\sigma^2, \tau \sigma\}$. We have now partitioned S_3 into 3 left cosets (equivalence classes) each containing two elements.

In this case, the right cosets give a *different* partition of S_3 . Indeed, the distinct right cosets are $\{\epsilon, \tau\}, \{\sigma, \tau \sigma\}, \{\sigma^2, \sigma \tau\}$.

3) Let H be the (cyclic) subgroup $(n) = \{nk : k \in \mathbf{Z}\}$ of \mathbf{Z} . Then $a \equiv_H b$ if $-a + b \in (n)$ which says that $n|(b - a)$. In this situation, we will write \equiv_n instead of \equiv_H , and if $a \equiv_n b$ we will say that a and b are equivalent, or congruent, mod n . This agrees with the notation in Exercise II.3.4. The distinct left cosets of (n) are

$$\begin{aligned}0 + (n) &= \{nk : k \in \mathbf{Z}\} \\ 1 + (n) &= \{nk + 1 : k \in \mathbf{Z}\} \\ 2 + (n) &= \{nk + 2 : k \in \mathbf{Z}\} \\ &\vdots \\ n - 1 + (n) &= \{nk + n - 1 : k \in \mathbf{Z}\}. \blacktriangleleft\end{aligned}$$

(4.5) **LEMMA.** Suppose H is a subgroup of a group G . Then there is a 1-1 correspondence between H and any left (or right) coset of H . In particular, if H is finite, then every left (or right) coset of H contains $|H|$ elements.

PROOF. Given any left coset aH define a mapping $H \rightarrow aH$ by sending h to ah . This map is clearly onto and it is 1-1 by cancellation in G . A similar argument may be used for right cosets. ■

Note that the only coset of H that is a subgroup of G is H itself since it is the only coset that contains the identity element.

(4.6) DEFINITION. Suppose H is a subgroup of a group G and that there are a finite number of distinct left cosets of H in G (which will certainly be the case if G is finite). The number of distinct left cosets of H in G is called the *index of H in G* and is denoted $[G : H]$.

It is now a simple matter to obtain the following useful result.

(4.7) THEOREM (Lagrange). Suppose G is a finite group and H is a subgroup of G . Then

$$|G| = [G : H] \cdot |H|.$$

PROOF. The $|G|$ elements of G are partitioned into the $[G : H]$ left cosets of H with each coset containing $|H|$ elements. ■

(4.8) COROLLARY. If G is a finite group, then the order of every subgroup of G divides $|G|$.

We note that even if an integer m divides $|G|$ there may not exist a subgroup of order m (see Exercise 8.9). Exercise 3.10 shows that if G is a *cyclic* group and if m divides $|G|$, then there is always a subgroup of order m .

(4.9) COROLLARY. If G is a finite group, then the order of every element of G divides $|G|$.

PROOF. This follows from the previous corollary and the fact that $o(a) = |(a)|$. ■

(4.10) COROLLARY. If G is a finite group and $a \in G$, then $a^{|G|} = e$.

PROOF. From Corollary (4.9), we know that $|G|$ is a multiple of the order of a , so we have $|G| = ko(a)$ for some integer k . Therefore, $a^{|G|} = (a^{o(a)})^k = e^k = e$. ■

From these corollaries we see, for example, that a group of order 6 (like S_3 or Z_6) cannot have a subgroup or an element of order 4 or 5.

(4.11) COROLLARY. If $|G| = p$, where p is prime, then G is cyclic and every element different from the identity is a generator for G .

PROOF. Suppose $a \in G$ is not the identity. Then $|(a)|$ is not 1, hence, since $|(a)|$ divides the prime p , we must have $|(a)| = p$. But then $(a) = G$. ■

EXERCISES

- † 4.1. Let H be the subgroup $\{\epsilon, \sigma\tau\}$ of S_3 . Find the left and right cosets of H in S_3 .
- † 4.2. Let H be the subgroup $\{\epsilon, \alpha, \alpha^2, \alpha^3\}$ of D_4 . Find the left and right cosets of H in D_4 .
- 4.3. Let H be the subgroup $\{\epsilon, \beta\}$ of D_4 . Find the left and right cosets of H in D_4 .
- † 4.4. What are the possible orders for subgroups of D_4 ? For each such possible order, find a subgroup of that order.
- † 4.5. Suppose that G has order 4. Show that either G is cyclic or every element a of G satisfies $a^2 = e$.
- † 4.6. Suppose G is a cyclic group of order 12 with generator g . Find all the left cosets of $H = \langle g^3 \rangle$.
- 4.7. Verify that $\mu^6 = \epsilon$ for each permutation $\mu \in S_3$.
- † 4.8. Show that every proper (i.e., $\neq G$) subgroup of a group G of order 15 must be cyclic.
- † 4.9. Let G be a group and suppose H and K are subgroups of G such that $|H| = 7$ and $|K| = 12$. Prove that $H \cap K = \{e\}$.
- † 4.10. Suppose that H is a subgroup of G of index 2. Show that $g^2 \in H$ for all $g \in G$. (Hint: Show that $g^2H = H$.)
- † 4.11. Suppose G is a finite group, H is a subgroup of G , and K is a subgroup of H . Use Lagrange's Theorem to prove that

$$[G : K] = [G : H][H : K].$$

Find an example illustrating this situation in D_4 .

- 4.12. Suppose G is a finite group and H is a subgroup of G . Prove that there are $[G : H]$ distinct *right* cosets of H in G .
- 4.13. Show that, even if G is not finite, the set of left cosets of H in G is in 1-1 correspondence with the set of right cosets of H in G . (Hint: Correspond Ha^{-1} to aH .)

- 4.14. Suppose that G is a group of order p^2 , where p is a prime. Prove that G has a subgroup of order p . (Hint: Either G is cyclic or it's not. Treat each case separately.)

5. Congruences and Quotients

Let S be a semigroup and suppose R is an equivalence relation on S . For $s \in S$, let $[s]$ denote the equivalence class of s with respect to R . It is natural to consider when the equivalence relation R is compatible with the algebraic structure of S . This means we want to see when we can define a semigroup structure on the set of equivalence classes of R . The obvious operation to define on the equivalence classes would be

$$[s][t] = [st].$$

But there may be a problem with this definition as the next example shows.

- (5.1) **EXAMPLE.** Consider the semigroup (\mathbf{Z}_4, \cdot_4) . Partitioning the set \mathbf{Z}_4 into the two subsets $\{0, 1\}$ and $\{2, 3\}$ defines an equivalence relation on \mathbf{Z}_4 . It may seem reasonable to define an operation on these two equivalence classes as above, but observe that while

$$[0][2] = [0] = \{0, 1\},$$

we also would have

$$[1][2] = [2] = \{2, 3\}.$$

Since $[0] = [1]$ there is obviously something wrong here. The problem is that the operation is not *well-defined*. In other words, the product of two equivalence classes depends here on the choices of the representatives of the classes. *Whenever we wish to define an operation on a set of equivalence classes we need to see that the operation is well-defined.*

Consider a second equivalence relation on \mathbf{Z}_4 given by the partition into the two subsets $\{0, 2\}$ and $\{1, 3\}$. Now we can have a well-defined operation on the equivalence classes by defining $[s][t] = [s \cdot_4 t]$ since

$$[0][1] = [2][1] = [0][3] = [2][3] = \{0, 2\}$$

$$[0][0] = [2][2] = [0][2] = \{0, 2\}$$

$$[1][1] = [1][3] = [3][3] = \{1, 3\}. \blacktriangleleft$$

We now give a special name to an equivalence relation on a semigroup that is compatible with the operation on the semigroup, in the sense that the above natural operation on the equivalence classes is well-defined.

(5.2) DEFINITION. Suppose S is a semigroup and R is an equivalence relation on S . Then R is called a *congruence* if whenever sRs' and tRt' , then $(st)R(s't')$ for all $s, s', t, t' \in S$.

(5.3) DEFINITION. If R is an equivalence relation on a semigroup S , then we denote the set of equivalence classes of R by S/R (read “ S mod R ”).

For emphasis, we note that an element of S/R is a subset of S , namely an equivalence class of R .

(5.4) PROPOSITION. If R is a congruence on S , then S/R forms a semigroup under the operation $[s][t] = [st]$.

PROOF. The fact that this operation is well-defined is exactly the definition of a congruence. Obviously, S/R is closed under this operation. Finally, this operation “inherits” the associativity of the operation on S since

$$\begin{aligned} [s_1]([s_2][s_3]) &= [s_1][s_2s_3] = [s_1(s_2s_3)] = [(s_1s_2)s_3] \\ &= [s_1s_2][s_3] = ([s_1][s_2])[s_3]. \end{aligned}$$

This completes the proof. ■

The semigroup S/R is called the *quotient semigroup*, or *factor semigroup*, of S by R . We leave it as an exercise to show that if S is actually a monoid or a group and R is a congruence on S , then S/R is a monoid or a group, respectively.

► (5.5) EXAMPLE

We consider the equivalence relation \equiv_n on \mathbf{Z} . Recall that $a \equiv_n b$ if and only if $n|(b-a)$. Notice that $n|(b-a)$ if and only if $n|(a-b)$. First, we show that this is a congruence on $(\mathbf{Z}, +)$. Suppose $a_1 \equiv_n b_1$ and $a_2 \equiv_n b_2$. Since we are dealing with an additive semigroup, we must show that $(a_1 + a_2) \equiv_n (b_1 + b_2)$. We have

$$a_1 - b_1 = nk \text{ and } a_2 - b_2 = nl$$

for some integers n and l . Hence we have

$$(a_1 + a_2) - (b_1 + b_2) = n(k + l),$$

so $(a_1 + a_2) \equiv_n (b_1 + b_2)$.

Next, we show that \equiv_n is a congruence on (\mathbb{Z}, \cdot) . With notation as above, we must show $a_1 a_2 \equiv_n b_1 b_2$. Now,

$$a_1 a_2 - b_1 b_2 = a_1 a_2 - a_1 b_2 + a_1 b_2 - b_1 b_2 = a_1(a_2 - b_2) + (a_1 - b_1)b_2.$$

Since $n|(a_1 - b_1)$ and $n|(a_2 - b_2)$, n divides the sum on the far right of the above equalities and we are done. ◀

We now consider an important example of a congruence related to finite-state machines.

- **(5.6) EXAMPLE.** Let $M = (S, X, Z, \tau, \omega)$ be a finite-state machine. Recall from (1.10) that X^* is the monoid of all finite input strings (including the empty string) with the operation being concatenation of strings. We now define an equivalence relation on X^* .

First, we need to extend the transition mapping on the machine to a map

$$\tau^* : S \times X^* \rightarrow S$$

in the same way that we extended the output function in I.3. Put $\tau^*(\sigma, \epsilon) = \sigma$ for all $\sigma \in S$. We define τ^* on input strings of length at least 1 recursively, as follows.

- 1) If $\hat{x} = x_1$ is a string of length 1, then define $\tau^*(\sigma, \hat{x}) = \tau(\sigma, x_1)$.
- 2) Assume that τ^* has been defined for strings of length r . If $\hat{x} = x_1 x_2 \cdots x_r x_{r+1}$ is a string of length $r + 1$, then define

$$\tau^*(\sigma, \hat{x}) = \tau(\tau^*(\sigma, x_1 x_2 \cdots x_r), x_{r+1}).$$

So, for example, if $\tau(\sigma, x_1) = \sigma'$, then

$$\tau^*(\sigma, x_1 x_2) = \tau(\tau(\sigma, x_1), x_2) = \tau(\sigma', x_2).$$

In words, if the machine is in state σ and the input string \hat{x} is fed into the machine, then $\tau^*(\sigma, \hat{x})$ is the state in which the machine winds up when it has processed all the inputs in \hat{x} .

To any input string $\hat{x} = x_1 x_2 \cdots x_n$, we can then associate a state transition map

$$T_{\hat{x}} : S \rightarrow S$$

defined by

$$T_{\hat{x}}(\sigma) = \tau^*(\sigma, \hat{x}).$$

Note that

$$T_{\hat{x}\hat{y}}(\sigma) = \tau^*(\sigma, \hat{x}\hat{y}) = T_{\hat{y}}(T_{\hat{x}}(\sigma)),$$

92

so $T_{\hat{x}\hat{y}} = T_{\hat{y}} \circ T_{\hat{x}}$.

(5.7) DEFINITION. Define a relation R on X^* by

$$\hat{x}R\hat{y} \text{ if and only if } T_{\hat{x}} = T_{\hat{y}}.$$

Thus $\hat{x}R\hat{y}$ if for every $\sigma \in S$ we have $\tau^*(\sigma, \hat{x}) = \tau^*(\sigma, \hat{y})$. It is easy to see that R is an equivalence relation on X^* . Moreover, R is a congruence. For suppose $\hat{x}R\hat{x}'$ and $\hat{y}R\hat{y}'$. Then $T_{\hat{x}} = T_{\hat{x}'}$ and $T_{\hat{y}} = T_{\hat{y}'}$. So obviously

$$T_{\hat{x}\hat{y}} = T_{\hat{y}} \circ T_{\hat{x}} = T_{\hat{y}'} \circ T_{\hat{x}'} = T_{\hat{x}'\hat{y}'}.$$

Thus, we have $\hat{x}\hat{y}R\hat{x}'\hat{y}'$. Note that the restriction of R to the semigroup X^+ is a congruence on X^+ . ◀

(5.8) DEFINITION. The semigroup X^+/R is called *the semigroup of the machine M* . The monoid X^*/R is called *the monoid of the machine M* .

The semigroup of the machine describes all the possible paths between states in the machine as the input string varies.

- **(5.9) EXAMPLE.** Let M be the parity check machine of Example (I.3.5). Then two input strings \hat{x} and \hat{y} induce the same state transition mapping if and only if they have the same parity; i.e., if and only if either both strings contain an even number of 1's or both strings contain an odd number of 1's. The semigroup of the machine X^+/R then contains two elements. Let $[0]$ denote the equivalence class consisting of all strings with an even number of 1's and let $[1]$ denote the equivalence class consisting of all strings containing an odd number of 1's. If we concatenate two even strings or two odd strings, we end up with an even string. If we concatenate an even string with an odd string, we get an odd string. The multiplication table of the semigroup of the parity check machine is then

	$[0]$	$[1]$
$[0]$	$[0]$	$[1]$
$[1]$	$[1]$	$[0]$

We note that in this case the semigroup of the machine is actually a group (which “looks like” $(\mathbb{Z}_2, +_2)$). In most cases the semigroup of a finite-state machine will not be a group. We will return to the semigroup of a machine in §7. ◀

Now we turn our attention to groups. The most important type of equivalence relation on a group is the relation \equiv_H associated with a subgroup H , which was discussed in the previous section. The next result tells when this equivalence relation is a congruence.

(5.10) PROPOSITION. Suppose G is a group and H is a subgroup of G . The equivalence relation \equiv_H is a congruence if and only if the left coset aH equals the right coset Ha for every $a \in G$.

PROOF. First, suppose that \equiv_H is a congruence. Then we must show that the sets aH and Ha coincide for every $a \in G$. The usual way to show that two sets are equal is to show that each is a subset of the other. So, suppose that $b \in aH$. Then $b = ah$ for some $h \in H$. Hence we have $ba^{-1} = aha^{-1}$. We claim that $aha^{-1} \in H$. Indeed, we have $a^{-1} \equiv_H a^{-1}$ and $h^{-1} \equiv_H e$ (since $h = (h^{-1})^{-1}e \in H$). Then, since we are assuming \equiv_H is a congruence, we have $h^{-1}a^{-1} \equiv_H a^{-1}$ which means that

$$(h^{-1}a^{-1})^{-1}a^{-1} = aha^{-1} \in H.$$

Thus we have $ba^{-1} \in H$, so $ba^{-1} = h'$ for some $h' \in H$. Therefore, $b = h'a$ and $b \in Ha$. The proof that Ha is a subset of aH is entirely similar.

Now for the other direction. Suppose $aH = Ha$ for every $a \in G$. Suppose $b \equiv_H c$ and $\tilde{b} \equiv_H \tilde{c}$. We must show that $b\tilde{b}^{-1}c\tilde{c} \in H$, which means that $(b\tilde{b}^{-1})^{-1}c\tilde{c} \in H$. We have

$$(b\tilde{b}^{-1})^{-1}c\tilde{c} = \tilde{b}^{-1}b^{-1}c\tilde{c}.$$

Now, $b^{-1}c \in H$, since $b \equiv_H c$, and by assumption $H\tilde{c} = \tilde{c}H$, so $(b^{-1}c)\tilde{c} = \tilde{c}h$ for some $h \in H$. Then we have

$$(b\tilde{b}^{-1})^{-1}c\tilde{c} = \tilde{b}^{-1}b^{-1}c\tilde{c} = \tilde{b}^{-1}(\tilde{c}h) = (\tilde{b}^{-1}\tilde{c})h \in H$$

since $\tilde{b}^{-1}\tilde{c} \in H$ and H is a subgroup. This completes the proof. ■

We give a special name to subgroups that satisfy the property in the preceding proposition.

(5.11) DEFINITION. A subgroup H of a group G is called a *normal subgroup* of G if $aH = Ha$ for all $a \in G$.

Often, the notation $H \triangleleft G$ is used to signify that H is a normal subgroup of G . In Exercise 5.5(a), you are asked to show that H is a normal subgroup of G if and only if $ghg^{-1} \in H$ for every $g \in G$ and $h \in H$. In practice, this criterion for a subgroup to be normal is probably used more often than Definition (5.11).

► (5.12) EXAMPLES

1) It is obvious that every subgroup H of an abelian group G is normal. Indeed, in this case we not only have $aH = Ha$, but actually $ah = ha$ for every $a, h \in G$.

2) In Example (4.4), we showed that $\{\epsilon, \sigma, \sigma^2\}$ is a normal subgroup of S_3 , while $\{\epsilon, \tau\}$ is not a normal subgroup of S_3 .

3) Let G be the multiplicative group of 2×2 real matrices of the form

$$\begin{bmatrix} a & 0 \\ c & d \end{bmatrix} \quad \text{with } ad \neq 0.$$

Recall from linear algebra that the inverse of such a matrix is

$$\frac{1}{ad} \begin{bmatrix} d & 0 \\ -c & a \end{bmatrix}.$$

Let H denote the subset of G consisting of all matrices of the form $\begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix}$.

We claim that H is a normal subgroup of G .

First, we must show that H is a subgroup. We can use Proposition (3.8). Obviously, H is nonempty. If $B_1 = \begin{bmatrix} 1 & 0 \\ c_1 & 1 \end{bmatrix}$ and $B_2 = \begin{bmatrix} 1 & 0 \\ c_2 & 1 \end{bmatrix}$ are in H , then

$$B_1 B_2^{-1} = \begin{bmatrix} 1 & 0 \\ c_1 - c_2 & 1 \end{bmatrix},$$

which is of the right form to be in H . Hence, H is a subgroup of G .

Now for the normality. Note that in this example “ aH ” actually means “ AH ” where A is a matrix in G . If $A = \begin{bmatrix} a & 0 \\ c & d \end{bmatrix}$, then an element of AH looks like

$$\begin{bmatrix} a & 0 \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 0 \\ c_1 & 1 \end{bmatrix} = \begin{bmatrix} a & 0 \\ c + c_1 d & d \end{bmatrix}. \quad (*)$$

We must see that the matrix on the right side of equation (*) is also of the form $\begin{bmatrix} 1 & 0 \\ c_2 & 1 \end{bmatrix} A$ for some c_2 . Multiplying these two matrices and setting the product equal to the matrix on the right side of equation (*), we see that we can take $c_2 = c_1 d / a$ (note that $a \neq 0$). Hence any matrix in AH is also in HA . Similarly, one can show that any matrix in HA is also in AH .

We can also use the criterion in Exercise 5.5(a) at the end of this section to show that H is normal. For if

$$A = \begin{bmatrix} a & 0 \\ c & d \end{bmatrix} \in G \text{ and } B = \begin{bmatrix} 1 & 0 \\ c' & 1 \end{bmatrix} \in H,$$

then a computation shows that

$$ABA^{-1} = \begin{bmatrix} 1 & 0 \\ c'd/a & 1 \end{bmatrix},$$

which is in H . ◀

(5.13) NOTATION. Suppose H is a normal subgroup of a group G . We will write G/H (read “ G mod H ”) for the quotient G/\equiv_H .

Thus, the elements of G/H are the cosets of H in G . We already know from Proposition (5.4) that G/H is a semigroup, but we show now that it is actually a group (also see Exercise 5.2).

(5.14) PROPOSITION. Suppose H is a normal subgroup of a group G . Then G/H is a group under the operation $(aH)(bH) = (ab)H$.

PROOF. The Identity and Inverse axioms are what remain to be verified. The identity of G/H is clearly H . The inverse of the coset aH is the coset $a^{-1}H$. ■

If H is normal in G , then G/H is called the *quotient group*, or *factor group*, of G by H . We note that in G/H we have $(aH)^r = a^rH$ by definition of the operation in this quotient group.

► (5.15) EXAMPLES

1) Consider the quotient group $\mathbf{Z}/(n)$. If we write $[a]$ in place of the coset $a + (n)$, then the elements of this quotient group are $[0], [1], \dots, [n-1]$. It is easy to see that the operation on the quotient is just

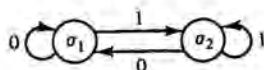
$$[a] + [b] = [a + b] = [a +_n b],$$

where $+_n$ was defined in Example (2.2.f). Thus $\mathbf{Z}/(n)$ is just the group $(\mathbf{Z}_n, +_n)$.

2) Let H denote the normal subgroup $\{\epsilon, \sigma, \sigma^2\}$ of S_3 . Then S_3/H just consists of the two elements H and τH , so the quotient group is cyclic of order 2.

EXERCISES

- † 5.1. a) Show that the equivalence relation on \mathbf{Z}_6 defined by the partition $\{0, 1, 2\}, \{3, 4, 5\}$ is not a congruence on (\mathbf{Z}_6, \cdot_6) .
 b) Define a congruence R_1 on (\mathbf{Z}_6, \cdot_6) such that the quotient semigroup \mathbf{Z}_6/R_1 has exactly two elements.
 c) Define a congruence R_2 on (\mathbf{Z}_6, \cdot_6) such that the quotient semigroup \mathbf{Z}_6/R_2 has exactly three elements.
- 5.2. a) Suppose S is a monoid with identity e and R is a congruence on S . Prove that S/R , which is a semigroup by Proposition (5.4), is a monoid.
 b) If S is a group and R is a congruence on S , then prove that S/R is a group.
- † 5.3. Consider the following finite-state machine:



- a) Show that two input strings are equivalent in the sense of Example (5.6) if and only if they end in the same input.
 b) Determine the multiplication table for the semigroup of this machine. Is this semigroup a group?
- † 5.4. Suppose H is a subgroup of index 2 in a group G . Show that H is normal. (Hint: What are the two cosets of H in G ?)
- † 5.5. Suppose H is a subgroup of a group G .
 a) Prove that H is a normal subgroup if and only if $ghg^{-1} \in H$ for all $g \in G$ and $h \in H$.
 b) For any $g \in G$, let $gHg^{-1} = \{ghg^{-1} : h \in H\}$. Prove that H is a normal subgroup of G if and only if $gHg^{-1} = H$.
- † 5.6. Let G be the multiplicative group of real 2×2 matrices of the form $\begin{bmatrix} a & b \\ 0 & d \end{bmatrix}$ where $ad \neq 0$. Let H denote the subset of G consisting of matrices of the form $\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$. Prove that H is a normal subgroup of G .
- † 5.7. Find all normal subgroups of D_4 .

- † 5.8. Show that in D_4 there exist subgroups H and K such that K is a normal subgroup of H and H is a normal subgroup of D_4 , but K is not a normal subgroup of D_4 .
- † 5.9. Prove that if G is abelian and H is a subgroup of G , then G/H is abelian.
- † 5.10. Suppose H is a normal subgroup of a group G . Show that if $o(aH) = m$ in G/H , then $m|o(a)$. (Hint: Use Exercise 3.8.)
- † 5.11. Prove that if H is a subgroup of a cyclic group G , then G/H is cyclic. (Hint: If g is a generator for G , what coset can be a generator for G/H ?)
- † 5.12. Prove that the intersection of two normal subgroups of a group is again a normal subgroup.
- 5.13. Prove that the center of a group is always a normal subgroup.
- 5.14. Suppose H is the only subgroup of G of order $|H|$. Prove that H is a normal subgroup of G . (Hint: Use Exercise 5.5(b) above.)
- 5.15. Suppose H is a subgroup of a group G . Recall from Exercise 3.20 that the normalizer of H is $N(H) = \{g \in G : gHg^{-1} = H\}$.
- Show that H is contained in $N(H)$.
 - Prove that H is a normal subgroup of $N(H)$.
 - Show that if K is a subgroup of G such that H is normal in K , then $K \subseteq N(H)$.
 - If H is normal in G , what is $N(H)$?
- 5.16. Prove *Cauchy's Theorem for abelian groups*: Let G be a finite abelian group such that p divides $|G|$, where p is a prime. Then G has an element (and hence a subgroup) of order p . (Hint: Use induction on the order of the group. If G does not have an element of order p , then let $H = \langle a \rangle$, $a \neq e$. Apply the induction hypothesis to the group G/H to get a coset gH of order p . Now show that some power of g must have order p by using Exercise 5.10.)
- 5.17. Prove that an abelian group of order pq , where p and q are primes, must be cyclic. (Hint: Use Exercise 5.16 and Exercise 3.12.)

6. Homomorphisms

The most useful mappings (functions) between two binary algebras are the ones that respect the algebraic structures, in the sense that the image of a product is the product of the images. One gives a special name to such mappings.

(6.1) DEFINITIONS. A mapping $\varphi : (A, \circ) \rightarrow (B, *)$ between two binary algebras is called a *homomorphism* if

$$\varphi(a_1 \circ a_2) = \varphi(a_1) * \varphi(a_2)$$

for all $a_1, a_2 \in A$. Such a mapping is called a *homomorphism of semigroups* (resp. *monoids*, *groups*) if the two binary algebras are semigroups (resp. monoids, groups). A homomorphism is often called a *monomorphism* if it is injective (i.e., 1-1) and an *epimorphism* if it is surjective (i.e., onto).

We leave it as an exercise to show that if φ is a homomorphism, then

$$\varphi(a_1 \circ a_2 \circ \cdots \circ a_n) = \varphi(a_1) * \varphi(a_2) * \cdots * \varphi(a_n).$$

(6.2) DEFINITIONS. An *isomorphism* is a homomorphism that is also a 1-1, onto mapping. Two semigroups (resp. monoids, groups) are called *isomorphic* if there exists an isomorphism from one onto the other. If (A, \circ) and $(B, *)$ are isomorphic, we write $A \cong B$. An isomorphism from (A, \circ) onto itself is called an *automorphism* of A .

When two semigroups (monoids, groups) are isomorphic it means that they are virtually the same. More precisely, in the finite case it means that we can set up a 1-1 correspondence between the elements of the two semigroups (monoids, groups) such that the two multiplication tables coincide under this correspondence. We note, though, that if two semigroups have the same number of elements, then it does not follow that they must be isomorphic (see Exercise 6.2).

► (6.3) EXAMPLES

1) Consider the mapping $\exp : (\mathbf{R}, +) \rightarrow (\mathbf{R}_+, \cdot)$ given by $\exp(x) = e^x$ (here \mathbf{R}_+ denotes the positive real numbers and e is the real number 2.71828...). This is a homomorphism of groups since $\exp(x + y) = e^{x+y} = e^x e^y = \exp(x) \exp(y)$. Moreover, since this mapping is 1-1 and onto (the natural log is the inverse map), it is an isomorphism and $(\mathbf{R}, +) \cong (\mathbf{R}_+, \cdot)$.

2) We will show that the groups D_3 and S_3 are isomorphic. (This was Exercise 1.7.) The group D_3 consists of the six symmetries of an equilateral

triangle. These symmetries are the identity, (counterclockwise) rotations by 120° and by 240° , and reflections about the three medians. Labeling our triangle as in Figure 6.1, it is clear how to set up our correspondence with S_3 .

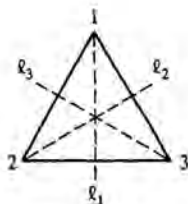


Figure 6.1

We have

$$\begin{aligned} \epsilon &= \text{the identity} \leftrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \\ \alpha &= \text{rotation by } 120^\circ \leftrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \\ \alpha^2 &= \text{rotation by } 240^\circ \leftrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \\ \beta &= \text{reflection about } l_1 \leftrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \\ \alpha\beta &= \text{reflection about } l_3 \leftrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} \\ \alpha^2\beta &= \text{reflection about } l_2 \leftrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}. \end{aligned}$$

We leave it to the reader to see that this 1-1 correspondence is in fact a homomorphism. One way to see this is to construct the multiplication table for D_3 and to see that this table coincides with the table for S_3 (Table (1.5)) when the above correspondences are made. ◀

For another example, we prove

(6.4) PROPOSITION. Any two cyclic groups of order n are isomorphic.

PROOF. Suppose that $G = \{e, g, g^2, \dots, g^{n-1}\}$ and $\tilde{G} = \{\tilde{e}, \tilde{g}, \tilde{g}^2, \dots, \tilde{g}^{n-1}\}$. Define a map $\varphi : G \rightarrow \tilde{G}$ by $\varphi(g^i) = \tilde{g}^i$ for $i = 0, 1, \dots, n-1$. Clearly, this

map is 1-1 and onto, and it is a homomorphism since

$$\varphi(g^i g^j) = \varphi(g^{i+j}) = \tilde{g}^{i+j} = \tilde{g}^i \tilde{g}^j = \varphi(g^i) \varphi(g^j).$$

This completes the proof. ■

Recall that if $\varphi : S \rightarrow T$, then the image of φ , denoted $\varphi(S)$, is $\{t \in T : t = \varphi(s) \text{ for some } s \in S\}$.

(6.5) LEMMA. Suppose $\varphi : (S, \circ) \rightarrow (T, *)$ is a homomorphism of semigroups. Then $\varphi(S)$ is a subsemigroup of T .

PROOF. This is virtually the definition of a homomorphism. Since $\varphi(s_1) * \varphi(s_2) = \varphi(s_1 \circ s_2)$, we see that $\varphi(S)$ is closed under $*$. ■

For the remainder of this section, we assume that $\varphi : (G, \circ) \rightarrow (\tilde{G}, *)$ is a homomorphism of groups. Let e denote the identity of G and let \tilde{e} denote the identity of \tilde{G} .

(6.6) LEMMA

- 1) $\varphi(e) = \tilde{e}$.
- 2) $\varphi(g)^{-1} = \varphi(g^{-1})$ for all $g \in G$.

PROOF. 1) We have

$$\tilde{e} * \varphi(e) = \varphi(e) = \varphi(e \circ e) = \varphi(e) * \varphi(e).$$

Hence, by cancellation in \tilde{G} , $\tilde{e} = \varphi(e)$.

2) We have

$$\tilde{e} = \varphi(g \circ g^{-1}) = \varphi(g) * \varphi(g^{-1}).$$

But then $\varphi(g^{-1}) = \varphi(g)^{-1}$ by Exercise 2.14. ■

Referring to Example (6.3.1), this Lemma simply would say that $e^0 = 1$ and $e^{-x} = 1/e^x$.

(6.7) COROLLARY. $\varphi(G)$ is a subgroup of \tilde{G} .

PROOF. By Lemma (6.5), we know that $\varphi(G)$ is a subsemigroup of \tilde{G} . Lemma (6.6) shows that $\varphi(G)$ also satisfies the Identity and Inverse Axioms. ■

Of special interest are the elements of G that map to the identity of \tilde{G} .

(6.8) DEFINITION. The *kernel* of φ , denoted $\text{Ker}(\varphi)$, is $\{g \in G : \varphi(g) = \tilde{e}\}$.

Note that $\text{Ker}(\varphi)$ is nonempty since e always belongs to $\text{Ker}(\varphi)$ by Lemma (6.6).

(6.9) PROPOSITION. $\text{Ker}(\varphi)$ is a normal subgroup of G .

PROOF. First we must show that $\text{Ker}(\varphi)$ is a subgroup of G . Suppose $a, b \in \text{Ker}(\varphi)$. To show that $a \circ b^{-1}$ is in $\text{Ker}(\varphi)$, we must see that the image of this element is \tilde{e} . We have

$$\varphi(a \circ b^{-1}) = \varphi(a) * \varphi(b)^{-1} = \tilde{e} * \tilde{e} = \tilde{e}.$$

Hence $a \circ b^{-1}$ is in $\text{Ker}(\varphi)$ and $\text{Ker}(\varphi)$ is a subgroup of G by Proposition (3.8).

To show $\text{Ker}(\varphi)$ is a normal subgroup of G , it suffices by Exercise 5.5(a) to show that $g \circ k \circ g^{-1} \in \text{Ker}(\varphi)$ for all $g \in G$ and $k \in \text{Ker}(\varphi)$. We have

$$\begin{aligned} \varphi(g \circ k \circ g^{-1}) &= \varphi(g) * \varphi(k) * \varphi(g)^{-1} \\ &= \varphi(g) * \tilde{e} * \varphi(g)^{-1} \quad \text{since } k \in \text{Ker}(\varphi) \\ &= \varphi(g) * \varphi(g)^{-1} \\ &= \tilde{e} \end{aligned}$$

and we are done. ■

We now show that to see that different elements of G map to different elements of \tilde{G} it is enough to see that e is the only element that maps to \tilde{e} .

(6.10) PROPOSITION. φ is a 1-1 mapping if and only if $\text{Ker}(\varphi) = \{e\}$.

PROOF. One direction is clear – if φ is 1-1, then only e can map to \tilde{e} . As for the other direction, we suppose that $\text{Ker}(\varphi) = \{e\}$, and we must show that φ is 1-1. The standard way to show that a map φ is 1-1 is to suppose that $\varphi(a) = \varphi(b)$ and then to show that a must equal b . If $\varphi(a) = \varphi(b)$, then

$$\varphi(a \circ b^{-1}) = \varphi(a) * \varphi(b)^{-1} = \tilde{e}.$$

But this says that $a \circ b^{-1} \in \text{Ker}(\varphi)$. Hence we conclude that $a \circ b^{-1} = e$, so $a = b$. ■

An important example of a homomorphism of groups is the mapping

$$\begin{aligned}\pi_H : G &\longrightarrow G/H \\ g &\longmapsto gH\end{aligned}$$

where H is a normal subgroup of G . The fact that π_H is a homomorphism is immediate from the definition of the operation in G/H . Also, since $gH = H$ if and only if $g \in H$, we see that $\text{Ker}(\pi_H) = H$.

(6.11) THEOREM. Suppose $\varphi : G \rightarrow \tilde{G}$ is a homomorphism of groups. Put $H = \text{Ker}(\varphi)$. Then the mapping

$$\bar{\varphi} : G/H \longrightarrow \tilde{G}$$

defined by $\bar{\varphi}(gH) = \varphi(g)$ is a 1-1 homomorphism of groups.

PROOF. Our first task is to see that the map $\bar{\varphi}$ is well-defined; i.e., if $gH = g'H$, we must see that $\varphi(g) = \varphi(g')$. (Recall the possible problem we had in defining an operation on G/H .) But $gH = g'H$ if and only if $g^{-1} \circ g' \in H = \text{Ker}(\varphi)$. Hence, if $gH = g'H$, then $\bar{e} = \varphi(g^{-1} \circ g') = \varphi(g)^{-1} * \varphi(g')$, so $\varphi(g) = \varphi(g')$.

Next we see that $\bar{\varphi}$ is a homomorphism. We have

$$\begin{aligned}\bar{\varphi}(gH \circ g'H) &= \bar{\varphi}((g \circ g')H) = \varphi(g \circ g') \\ &= \varphi(g) * \varphi(g') = \bar{\varphi}(gH) * \bar{\varphi}(g'H).\end{aligned}$$

Finally, we see that $\bar{\varphi}$ is 1-1. We will use Proposition (6.10). Suppose $\bar{\varphi}(gH) = \bar{e}$. Then $\varphi(g) = \bar{e}$, hence $g \in H$, since $H = \text{Ker}(\varphi)$, and so $gH = H$. Therefore the kernel of $\bar{\varphi}$ contains only the identity of G/H . ■

(6.12) COROLLARY. (First Isomorphism Theorem) If $\varphi : G \rightarrow \tilde{G}$ is a surjective (i.e., onto) homomorphism of groups, then $\bar{\varphi} : G/\text{Ker}(\varphi) \rightarrow \tilde{G}$, as defined in Theorem (6.11), is an isomorphism.

PROOF. Clearly, if φ is an onto mapping, then $\bar{\varphi}$ will also be onto. Since $\bar{\varphi}$ was shown to be a homomorphism and 1-1 in Theorem (6.11), it is an isomorphism. ■

Note that the definition of $\bar{\varphi}$ means that the following diagram is “commutative,” in the sense that $\varphi = \bar{\varphi} \circ \pi_H$.

$$\begin{array}{ccc} G & \xrightarrow{\varphi} & \tilde{G} \\ \pi_H \searrow & & \nearrow \bar{\varphi} \\ & G/H & \end{array}$$

► (6.13) EXAMPLES

1) Define a mapping φ from $(\mathbf{Z}_4, +_4)$ to $(\mathbf{Z}_2, +_2)$ by

$$0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 0, 3 \mapsto 1.$$

It is not hard to check that this is a homomorphism of groups. The kernel of φ is the (normal) subgroup $\{0, 2\}$ of $(\mathbf{Z}_4, +_4)$ and by the First Isomorphism Theorem above, we have

$$\mathbf{Z}_4 / \{0, 2\} \cong \mathbf{Z}_2.$$

2) Consider the map $\varphi : \mathbf{Z} \rightarrow \mathbf{Z}_n$ that takes an integer a to the remainder when a is divided by n (i.e., if $a = qn + r$ with $0 \leq r < n$, then $\varphi(a) = r$). If $a_1 = q_1n + r_1$ and $a_2 = q_2n + r_2$, with $0 \leq r_1, r_2 < n$, then

$$a_1 + a_2 = \begin{cases} (q_1 + q_2)n + (r_1 + r_2) & \text{if } r_1 + r_2 < n \\ (q_1 + q_2 + 1)n + (r_1 + r_2 - n) & \text{if } (r_1 + r_2) \geq n, \end{cases}$$

so $\varphi(a_1 + a_2) = \varphi(a_1) +_n \varphi(a_2)$. Obviously, the kernel of φ is (n) , the cyclic subgroup generated by n , and φ is surjective. Hence, by the First Isomorphism Theorem,

$$(\mathbf{Z}/(n), +) \cong (\mathbf{Z}_n, +_n),$$

so we may identify these two groups.

3) Consider the mapping

$$\det : \text{GL}(2, \mathbf{R}) \longrightarrow (\mathbf{R}^*, \cdot)$$

where $\text{GL}(2, \mathbf{R})$ is the group of 2×2 real matrices with nonzero determinant, \mathbf{R}^* is the group of nonzero real numbers under multiplication, and the map takes a matrix to its determinant. (The group $\text{GL}(2, \mathbf{R})$ is called the *general linear group* of nonsingular 2×2 real matrices.) Since we know from linear algebra that $\det(AB) = \det(A)\det(B)$, the map \det is a homomorphism of groups. This map is onto, since if $r \in \mathbf{R}^*$, then

$$\det \left(\begin{bmatrix} r & 0 \\ 0 & 1 \end{bmatrix} \right) = r.$$

The identity of \mathbf{R}^* is 1, so the kernel of this map is the normal subgroup

$$\text{SL}(2, \mathbf{R}) = \{A \in \text{GL}(2, \mathbf{R}) : \det(A) = 1\}$$

of $\text{GL}(2, \mathbf{R})$. (The group $\text{SL}(2, \mathbf{R})$ is called the *special linear group*.) Therefore, by the First Isomorphism Theorem, we have

$$\text{GL}(2, \mathbf{R}) / \text{SL}(2, \mathbf{R}) \cong \mathbf{R}^*. \blacktriangleleft$$

EXERCISES

- 6.1. Show that the semigroup of the parity check machine (Example (5.9)) is isomorphic to the group $(\mathbb{Z}_2, +_2)$.
- † 6.2. The semigroup of mappings from $\{0, 1\}$ into itself (with composition as the operation) has four elements. Is this semigroup isomorphic to either $(\mathbb{Z}_4, +_4)$ or (\mathbb{Z}_4, \cdot_4) ?
- 6.3. Suppose R is a congruence on a semigroup S and for $s \in S$, let $[s]$ denote the equivalence class of s with respect to R . Prove that the mapping $\varphi : S \rightarrow S/R$ given by $s \mapsto [s]$ is a homomorphism of semigroups.
- † 6.4. Suppose $\varphi : (S, \circ) \rightarrow (T, *)$ is a homomorphism of semigroups. Use mathematical induction to show that

$$\varphi(a_1 \circ a_2 \circ \cdots \circ a_n) = \varphi(a_1) * \varphi(a_2) * \cdots * \varphi(a_n)$$

for all $a_1, a_2, \dots, a_n \in S$.

- 6.5. Let $(S, *)$ be a semigroup. To each element $s \in S$, associate a mapping $f_s : S \rightarrow S$ where $f_s(t) = s * t$ for all $t \in S$. Let $F_S = \{f_s : s \in S\}$.

- a) Show that (F_S, \circ) is a semigroup, where \circ denotes composition. (Show that $f_s \circ f_{s'} = f_{s * s'}$.)
- b) Show that the map

$$\begin{aligned} \varphi : S &\longrightarrow F_S \\ s &\longmapsto f_s \end{aligned}$$

is a homomorphism of semigroups.

- c) If $(S, *)$ is a monoid, show that the homomorphism φ in part (b) is 1-1 (thus S is isomorphic to $\varphi(S)$).
- † 6.6. Exhibit two different automorphisms of the group $(\mathbb{Z}_6, +_6)$.
- † 6.7. Suppose $\varphi : (S, *) \rightarrow (T, \diamond)$ and $\psi : (T, \diamond) \rightarrow (U, \cdot)$ are homomorphisms of semigroups. Show that $\psi \circ \varphi$ is a homomorphism from S to U .
- 6.8. Suppose $\varphi : (S, \circ) \rightarrow (T, *)$ is an isomorphism of semigroups. Show that S is a commutative semigroup if and only if T is also.
- † 6.9. Let G be a group and fix an element $a \in G$. Define a mapping $\varphi_a : G \rightarrow G$ by $\varphi_a(g) = aga^{-1}$ for all $g \in G$. Show that φ_a is an automorphism of G . (Such an automorphism is called an *inner automorphism* of G .)

- † 6.10. Suppose G is an abelian group. Show that the map $\varphi(g) = g^{-1}$ defines an automorphism of G .
- 6.11. Let $(G, +)$ be the group of all functions $f : [0, 1] \rightarrow \mathbf{R}$ with the operation being addition $((f+g)(x) = f(x) + g(x))$. Let H denote the subset of G consisting of all such f with $f(0) = 0$. Define a surjective homomorphism $\varphi : (G, +) \rightarrow (\mathbf{R}, +)$ such that the kernel of φ is H . Thus you have shown that H is a normal subgroup of G and $G/H \cong (\mathbf{R}, +)$.
- † 6.12. Suppose $\varphi : (G, \circ) \rightarrow (\tilde{G}, *)$ is a homomorphism of groups. If \tilde{H} is a subset of \tilde{G} , then the inverse image of \tilde{H} , denoted $\varphi^{-1}(\tilde{H})$, is $\{g \in G : \varphi(g) \in \tilde{H}\}$.
- Show that if \tilde{H} is a subgroup of \tilde{G} , then $\varphi^{-1}(\tilde{H})$ is a subgroup of G and $\text{Ker}(\varphi) \subseteq \varphi^{-1}(\tilde{H})$.
 - Show that if \tilde{H} is a normal subgroup of \tilde{G} then $\varphi^{-1}(\tilde{H})$ is a normal subgroup of G .
- † 6.13. Suppose $\varphi : (G, \circ) \rightarrow (\tilde{G}, *)$ is a *surjective* homomorphism of groups. Show that if H is a normal subgroup of G , then $\varphi(H) = \{\varphi(h) : h \in H\}$ is a normal subgroup of \tilde{G} . (We already know it is a subgroup by Corollary (6.7).)
- 6.14. Suppose $\varphi : (G, \circ) \rightarrow (\tilde{G}, *)$ is a surjective homomorphism of groups. Show that there is a 1-1 correspondence between the normal subgroups of \tilde{G} and the normal subgroups of G that contain $\text{Ker}(\varphi)$. (Use the previous two exercises.)
- 6.15. In this problem, suppose that H is a subgroup of a group (G, \cdot) and that K is a normal subgroup of G .
- Let $HK = \{hk : h \in H, k \in K\}$. Show that HK is a subgroup of G .
 - Find two (nonnormal) subgroups H_1, H_2 of S_3 such that H_1H_2 is *not* a subgroup of S_3 .
 - Show that K is a normal subgroup of HK .
 - Define a mapping $\varphi : H \rightarrow HK/K$ by $\varphi(h) = hK$. Show that φ is a surjective homomorphism of groups.
 - Prove the *Second Isomorphism Theorem* :

$$H/(H \cap K) \cong HK/K$$

by showing that the kernel of φ is $H \cap K$.

- 6.16. Suppose H and K are subgroups of a finite group G and suppose K is normal in G . Use the previous exercise to show that

$$|HK| = \frac{|H||K|}{|H \cap K|}.$$

- 6.17. Suppose H and K are normal subgroups of a group G such that $HK = G$ and $H \cap K = \{e\}$. Prove that G is isomorphic to the direct product $H \times K$ (see Exercise 2.16). (Hint: Define $\varphi : H \times K \rightarrow G$ by $\varphi((h, k)) = hk$.)
- 6.18. Suppose $K \triangleleft H \triangleleft G$ and $K \triangleleft G$. (Note that, by Exercise 5.8, we cannot infer from $K \triangleleft H$ and $H \triangleleft G$ that K must be normal in G .) Define a map $\varphi : G/K \rightarrow G/H$ by $\varphi(gK) = gH$.
- Show that φ is well-defined.
 - Show that φ is a surjective homomorphism of groups.
 - Prove the *Third Isomorphism Theorem*:

$$G/K \big/ H/K \cong G/H$$

by showing that the kernel of φ is H/K .

- d) Prove that

$$(\mathbf{Z}/(4) \big/ 2\mathbf{Z}/(4), +) \cong (\mathbf{Z}/(2), +).$$

7. Semigroups and Machines

In this section, we investigate the connection between finite-state machines and semigroups. We will be interested only in the state transitions within a machine and not in the outputs of the machine. So, for this section a “machine” will mean a triple

$$M = (S, X, \tau)$$

where S is the (finite) set of states, X is the (finite) input alphabet, and $\tau : S \times X \rightarrow S$ is the next state (or transition) mapping.

We recall that X^+ denotes the semigroup of all finite length (≥ 1) input strings and X^* denotes the monoid $X^+ \cup \{\epsilon\}$, with the operation on these binary algebras being concatenation. We defined a congruence R on

these semigroups as follows: To each input string \hat{x} we associate the map $T_{\hat{x}} : S \rightarrow S$ defined by $T_{\hat{x}}(\sigma) = \tau^*(\sigma, \hat{x})$. (The extension of τ to τ^* was defined in Example (5.6).) We then say that two strings \hat{x} and \hat{y} are related by R if $T_{\hat{x}} = T_{\hat{y}}$.

We want to put a semigroup structure on the set of mappings $\{T_{\hat{x}} : \hat{x} \in X^+\}$. The obvious operation to consider is composition of mappings, but since we want the map that associates \hat{x} to $T_{\hat{x}}$ to be a homomorphism, we will define the operation on this set to be composition in the reverse order (that's the way we concatenate strings). More precisely, put

$$\begin{aligned}\mathcal{T} &= \mathcal{T}(M) = \{T_{\hat{x}} : \hat{x} \in X^+\} \\ \mathcal{T}^0 &= \mathcal{T}^0(M) = \{T_{\hat{x}} : \hat{x} \in X^*\}\end{aligned}$$

(so $\mathcal{T}^0 = \mathcal{T} \cup \{T_{\epsilon}\}$, where T_{ϵ} is the identity map on S). We define \diamond by

$$T_{\hat{x}} \diamond T_{\hat{y}} = T_{\hat{y}} \circ T_{\hat{x}} = T_{\hat{x}\hat{y}}.$$

By slight abuse of notation, we will use the same symbol for the operation on \mathcal{T} and on \mathcal{T}^0 . Then (\mathcal{T}, \diamond) is a semigroup and $(\mathcal{T}^0, \diamond)$ is a monoid. (In specific examples, these may be equal, as we shall see.)

Recall that we defined the semigroup of the machine to be X^+/R and the monoid of the machine to be X^*/R .

(7.1) PROPOSITION. We have

$$X^+/R \cong \mathcal{T} \text{ and } X^*/R \cong \mathcal{T}^0,$$

where the isomorphism in both cases is given by the map φ that takes $[\hat{x}]$ to $T_{\hat{x}}$.

PROOF. As always, when working with equivalence classes we need to see that our map is well-defined; i.e., if \hat{x} and \hat{x}' are two representatives of the same equivalence class, we must see that $T_{\hat{x}} = T_{\hat{x}'}$. But \hat{x} and \hat{x}' being in the same equivalence class says exactly that $T_{\hat{x}} = T_{\hat{x}'}$, by the definition of R .

We have set up the operation on \mathcal{T} so that φ will be a homomorphism. Indeed,

$$\varphi([\hat{x}][\hat{y}]) = \varphi([\hat{x}\hat{y}]) = T_{\hat{x}\hat{y}} = T_{\hat{x}} \diamond T_{\hat{y}} = \varphi([\hat{x}]) \diamond \varphi([\hat{y}]).$$

That φ is 1-1 again follows directly from the definition of R . For if $\varphi([\hat{x}]) = \varphi([\hat{y}])$, then $\hat{x}\hat{y} = \hat{y}\hat{y}$, so $[\hat{x}] = [\hat{y}]$. Finally, φ clearly maps \mathcal{S}^+ (resp. X^*) onto \mathcal{T} (resp. \mathcal{T}^0). ■

We will now describe an algorithm to compute the semigroup of a machine. By the above isomorphism, we can compute the semigroup of the machine by computing T , and it is this that the algorithm does.

(7.2) ALGORITHM FOR COMPUTING T .

- Step 1. Compute T_x for each $x \in X$. Discard any map that equals a previously found map. Set $n = 2$.
- Step 2. Compute $T_{\hat{x}}$ for each input string \hat{x} of length n . Discard any map that equals a previously found map.
- Step 3. If every map $T_{\hat{x}}$ for all input strings \hat{x} of length n equaled a previously found map (i.e., if Step 2 yielded no new maps), then stop. Otherwise, replace n by $n + 1$ and go to Step 2.

Notice that, since there are only finitely many states in the machine, there are only finitely many "paths" in the machine and the semigroup T must be finite. To see why the algorithm ends when no new maps are produced for all input strings of length n , we suppose that every map $T_{\hat{x}}$, where \hat{x} has length n , equals a previously found map and let \hat{y} be an input string of length $n + 1$. Write $\hat{y} = \hat{x}z$, where \hat{x} has length n and $z \in X$. Then

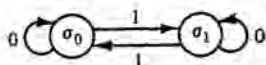
$$T_{\hat{y}} = T_{\hat{x}} \diamond T_z = T_{\hat{x}'} \diamond T_z$$

for some \hat{x}' of length $< n$. So $T_{\hat{y}} = T_{\hat{x}'z}$ and $\hat{x}'z$ has length at most n . Thus $T_{\hat{y}}$ equals a map previously found. This shows that no string of length $n + 1$ yields a new map. It is now not hard to give an inductive argument to show that no input string of length $> n$ can yield a new map.

We illustrate the algorithm with some examples.

► (7.3) EXAMPLES

- 1) We compute the semigroup of the parity check machine:



The following table displays the image of each state under the maps $T_{\hat{x}}$.

State	T_0	T_1	T_{00}	T_{01}	T_{10}	T_{11}
σ_0	σ_0	σ_1	σ_0	σ_1	σ_1	σ_0
σ_1	σ_1	σ_0	σ_1	σ_0	σ_0	σ_1

From the table, we see that

$$T_{00} = T_0, T_{01} = T_1, T_{10} = T_1, T_{11} = T_0. \quad (*)$$

The algorithm thus stops here because each $T_{\hat{x}}$ for \hat{x} of length 2 equals a previously found map (either T_0 or T_1).

It follows from $(*)$ that the multiplication table for the semigroup \mathcal{T} is

\diamond	T_0	T_1
T_0	T_0	T_1
T_1	T_1	T_0

This is obviously isomorphic to the semigroup X^+/R that was computed in Example 5.9 with the isomorphism being $[0] \mapsto T_0, [1] \mapsto T_1$. Notice that the semigroup \mathcal{T} here is actually a group. In particular, T_0 is the identity map, so $\mathcal{T} = \mathcal{T}^0$ for this machine.

2) We now consider an example that is more representative of the general situation. Consider the machine with digraph shown in Figure 7.1.

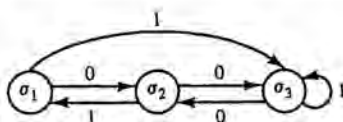


Figure 7.1

Here is the table displaying the maps $T_{\hat{x}}$ for \hat{x} of length ≤ 2 :

State	T_0	T_1	T_{00}	T_{01}	T_{10}	T_{11}
σ_1	σ_2	σ_3	σ_3	σ_1	σ_2	σ_3
σ_2	σ_3	σ_1	σ_2	σ_3	σ_2	σ_3
σ_3	σ_2	σ_3	σ_3	σ_1	σ_2	σ_3

Note that T_{10} takes every state to σ_2 . Therefore, $T_{\hat{x}} \diamond T_{10}$ will equal T_{10} for any string \hat{x} . (The element T_{10} is called a *right zero* in \mathcal{T} .) Similarly, $T_{\hat{x}} \diamond T_{11}$ will equal T_{11} for any \hat{x} . This means that the following maps associated to input strings of length 3 will equal either T_{10} or T_{11} : $T_{010}, T_{011}, T_{110}, T_{111}$.

Here is the table for the remaining maps associated to input strings of length 3:

State	T_{000}	T_{001}	T_{100}	T_{101}
σ_1	σ_2	σ_3	σ_3	σ_1
σ_2	σ_3	σ_1	σ_3	σ_1
σ_3	σ_2	σ_3	σ_3	σ_1

We see that $T_{000} = T_0$, $T_{001} = T_1$, $T_{100} = T_{11}$, and T_{101} is a new map.

We must now consider maps associated to input strings of length 4. But if \hat{x} has length 4, then $T_{\hat{x}} = T_z \diamond T_{\hat{y}}$, where $z \in \{0, 1\}$ and \hat{y} has length 3. Now, for any length 3 input string \hat{y} except 101 we have $T_{\hat{y}} = T_{\hat{y}'}$, where \hat{y}' has length at most 2. So if \hat{y} is anything except 101, then $T_{\hat{x}}$ must equal a previously found map. But if $\hat{y} = 101$, then $T_{\hat{x}} = T_{101} \circ T_z = T_{101}$, since T_{101} takes every state to σ_1 . Therefore, no new maps occur for input strings of length 4 and the algorithm stops.

The table for the semigroup \mathcal{T} of this machine may be computed to be

(7.4) TABLE

\diamond	T_0	T_1	T_{00}	T_{01}	T_{10}	T_{11}	T_{101}
T_0	T_{00}	T_{01}	T_0	T_1	T_{10}	T_{11}	T_{101}
T_1	T_{10}	T_{11}	T_{11}	T_{101}	T_{10}	T_{11}	T_{101}
T_{00}	T_0	T_1	T_{00}	T_{01}	T_{10}	T_{11}	T_{101}
T_{01}	T_{10}	T_{11}	T_{11}	T_{101}	T_{10}	T_{11}	T_{101}
T_{10}	T_{11}	T_{101}	T_{10}	T_{11}	T_{10}	T_{11}	T_{101}
T_{11}	T_{10}	T_{11}	T_{11}	T_{101}	T_{10}	T_{11}	T_{101}
T_{101}	T_{10}	T_{11}	T_{11}	T_{101}	T_{10}	T_{11}	T_{101}

Note that T_{00} is a left identity in this semigroup, but there is no two-sided identity; i.e., \mathcal{T} is not a monoid. The monoid \mathcal{T}^0 is obtained by adjoining T_ϵ (= the identity map on the set of states) to \mathcal{T} . We note that T_{00} is no longer a left identity in \mathcal{T}^0 since $T_{00} \diamond T_\epsilon = T_{00}$. ◀

We now consider how to associate a machine to any finite semigroup. Suppose (S, \circ) is a finite semigroup. We construct a machine by taking the set of states and the input alphabet to be S and by taking the transition function to be “multiplication” in S . More precisely, we make the following definition.

(7.5) DEFINITION. Given a finite semigroup (S, \circ) , we define the *machine of S* , denoted $\mathcal{A}(S)$, to be $\mathcal{A}(S) = (S, S, \tau_\circ)$ where $\tau_\circ(s, t) = s \circ t$.

- **(7.6) EXAMPLE.** Consider the group $(\mathbb{Z}_2, +_2)$. The machine of this group is then



This machine should look familiar. It is our friend, the parity check machine. Note that if we now compute the semigroup of this machine we get $(\mathbb{Z}_2, +_2)$ back again (Example (7.3.1) above). So if M is the parity check machine and S is the (semi)group $(\mathbb{Z}_2, +_2)$, we have

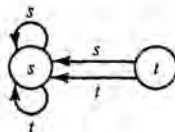
$$T(\mathcal{A}(S)) = S \text{ and } \mathcal{A}(T(M)) = M. \blacktriangleleft$$

Do these relationships always hold? No; but we investigate how much we can say in this direction.

- **(7.7) EXAMPLE.** Let (S, \circ) be the semigroup with the following multiplication table.

\circ	s	t
s	s	s
t	s	s

The machine of this semigroup is then



Since any input string takes every state to s , the semigroup of this machine is the trivial group $\{T_s\}$ (where $T_s \circ T_s = T_s$). Thus in this case the semigroup $T(\mathcal{A}(S))$ is not isomorphic to S . ◀

However, we do get an isomorphism if the semigroup we start with satisfies a fairly weak property.

(7.8) DEFINITION. A semigroup (S, \circ) is called *left discriminative*¹² if whenever $s \circ a = s \circ b$ for all $s \in S$, then $a = b$.

Equivalently, a finite semigroup (S, \circ) is left discriminative if there are not two identical columns in the multiplication table for S . Note that this condition is weaker than left cancellation (which says that if $s \circ a = s \circ b$ for *some* $s \in S$, then $a = b$). Also note that a monoid is left discriminative since $e \circ a = e \circ b$ obviously gives $a = b$. We state without proof the following result. A proof may be found in [2].

(7.9) THEOREM. Suppose (S, \circ) is a finite semigroup. Then $\mathcal{T}(\mathcal{A}(S)) \cong S$ if and only if S is left discriminative.

Finally, we consider the machine associated to the semigroup of a machine. To illustrate what can be said here, we return to Example (7.3.2). If we construct the machine associated to the semigroup of Table (7.4), then it is obvious that it will not equal the original machine since the input alphabets will be different. But we can try to get around this by restricting the input alphabet of the machine of this semigroup to $\{T_0, T_1\}$. More generally, we make the following definition.

(7.10) DEFINITION. If (\mathcal{T}, \diamond) is the semigroup of a machine $M = (S, X, \tau)$, then we define $\mathcal{A}_X(\mathcal{T})$ to be the machine $(\mathcal{T}, \{T_x : x \in X\}, \tau_\diamond)$.

► **(7.11) EXAMPLE.** We now construct the machine $\mathcal{A}_X(\mathcal{T})$ of the semigroup (\mathcal{T}, \diamond) given by Table (7.4). We write \hat{x} instead of $T_{\hat{x}}$ for the states and we write 0 and 1 for the inputs instead of T_0 and T_1 . The state diagram is shown in Figure 7.2.

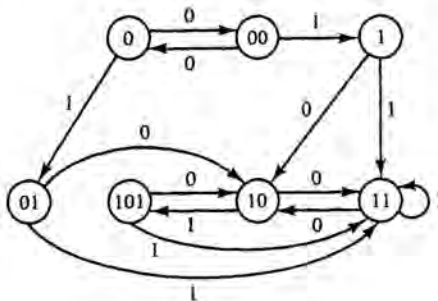


Figure 7.2

At first glance, this may not resemble the original machine of Example (7.3.2) very much. But if we partition the set of states into the three subsets

$$\{101, 01\}, \{10, 0\}, \{11, 00, 1\},$$

then note that an input of 0 takes a state in the first subset to a state in the second subset, a state in the second subset to a state in the third subset, and a state in the third subset to a state in the second subset. And an input of 1 takes a state in the first subset to a state in the third, a state in the second to a state in the first, and a state in the third to another state in the third. This is exactly what we need to define the quotient of the above machine by this equivalence relation on the states, in a similar manner to the way we defined a quotient machine in II.4. The state diagram of this quotient machine is shown in Figure 7.3, and this looks just like the original machine in Example (7.3.2)! ◀

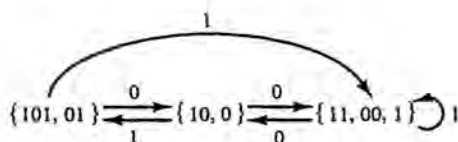


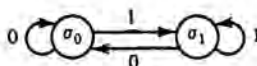
Figure 7.3

It turns out that this example is representative of what always happens if the original machine has a starting state, where a state σ_0 is called a *starting state* for $M = (S, X, \tau)$ if for every $\sigma \in S$, there exists $\hat{x} \in X^+$ such that $\tau^*(\sigma_0, \hat{x}) = \sigma$. We refer the reader to [15] for details.

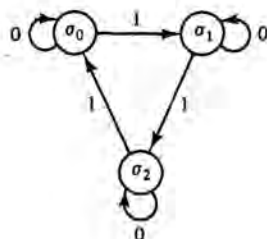
EXERCISES

In each of Exercises 7.1–7.4, find the multiplication table of the semigroup $T = T(M)$ of the machine M , determine if T is a monoid, construct the machine $\mathcal{A}_X(T(M))$, and partition the states of $\mathcal{A}_X(T(M))$, as in Example (7.11), to obtain a quotient machine that looks like the original machine.

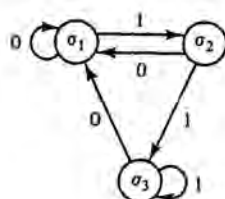
† 7.1. (Unit delay machine)



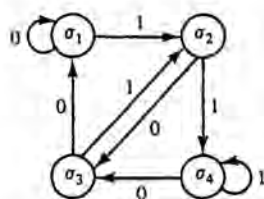
† 7.2. (Modulo 3 binary counter)



† 7.3.



7.4.



In Exercises 7.5–7.8, find the machine $\mathcal{A}(T)$ of the semigroup T and find the multiplication table of the semigroup $T(\mathcal{A}(T))$.

7.5. T is the semigroup given by the table

	a	b	c
a	b	b	a
b	b	b	b
c	b	b	c

† 7.6. T is the semigroup (\mathbb{Z}_3, \cdot_3) .

7.7. T is the group $(\mathbb{Z}_4, +_4)$.

- 7.8. T is the semigroup (\mathbf{Z}_4, \cdot_4) .
- 7.9. Show that the partition $\{T_{101}, T_{01}\}, \{T_{10}, T_0\}, \{T_{11}, T_{00}, T_1\}$ defines a congruence R on the semigroup (T, \circ) given by Table (7.4). Show that the machine $(T/R, \{T_0, T_1\}, \tau_\circ)$ is the machine of Figure 7.3.
- 7.10. Let $M = (S, X, \tau)$ be a machine. Suppose T is a subset of S and $\sigma \in S$. The *acceptor language* L associated to M, T , and σ is the set of input strings (words) $L = \{\hat{x} \in X^* : \tau^*(\sigma, \hat{x}) \in T\}$. The states in T are called designated final states, and the words in L are those words that carry the machine from the state σ to a state in T .
- Suppose M is the parity check machine of Example (7.3.1). Find the acceptor language associated to $M, T = \{\sigma_0\}$, and $\sigma = \sigma_0$. Also find the acceptor language associated to $M, T = \{\sigma_1\}$, and $\sigma = \sigma_0$.
 - Suppose M is the mod 3 binary counter of Example (I.3.8). Find the acceptor language associated to $M, T = \{\sigma_1\}$, and $\sigma = \sigma_0$.

8. Dihedral and Permutation Groups; Cayley's Theorem

We will deal exclusively with groups for the rest of this chapter. In this section, we return to dihedral and permutation groups, which were introduced briefly in §1. These groups will be essential for the applications to come in the following two sections.

Recall that D_n is the group of symmetries of a regular polygon with n sides. Geometrically, we can see that this group has order $2n$ as follows. It has a cyclic subgroup of order n generated by rotation by $(360/n)^\circ$. We denote this generator by α , so that the subgroup of rotations is $\langle \alpha \rangle = \{e, \alpha, \alpha^2, \dots, \alpha^{n-1}\}$.

There are also n distinct reflections of such a polygon. To see these geometrically, we must consider separately the cases when n is even and when n is odd. When n is even, the polygon has the $n/2$ lines through opposite (i.e., at opposite ends of a diameter) vertices and $n/2$ lines through midpoints of opposite sides as axes of symmetry and reflection about each of these lines yields n distinct symmetries. When n is odd we can no longer speak of opposite vertices. In this case, the medians (lines through a vertex and the midpoint of the opposite side) give n axes of symmetries and n reflections about each of these yield n distinct symmetries. (See Figures 8.1 and 8.2.)

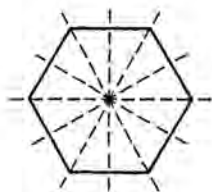


Figure 8.1

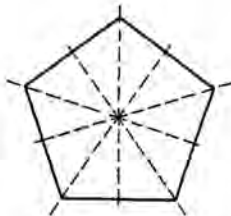


Figure 8.2

For any n , let β denote a reflection in D_n .

(8.1) PROPOSITION. The elements of D_n are

$$\begin{aligned} &\alpha^k \text{ for } k = 0, 1, \dots, n-1 \\ &\alpha^k \beta \text{ for } k = 0, 1, \dots, n-1. \end{aligned}$$

PROOF. By cancellation, we have that $\alpha^j \beta \neq \alpha^k \beta$ if $j \neq k$. Since we know that D_n has exactly $2n$ elements, all that remains to be shown is that $\alpha^k \beta \neq \alpha^i$ for any k and i . But if $\alpha^k \beta = \alpha^i$, then $\beta = \alpha^{i-k}$, which is a rotation. It is easy to see geometrically that no reflection is equal to any rotation. (Reflections, other than the ones about lines through midpoints of opposite sides, leave some, but not all of the vertices fixed — no rotation does this. A reflection about a line through midpoints of opposite sides switches vertices in pairs, as does rotation by 180° ; but under such a reflection two of the pairs of vertices that are switched consist of adjacent vertices, and this is not true of rotation by 180° .) ■

(8.2) DEFINITION. Let (G, \circ) be a group. The elements $g_1, g_2, \dots, g_m \in G$ are said to *generate* G if every element of G can be expressed as a product of powers of g_1, g_2, \dots, g_m .

Proposition (8.1) says that α and β are generators for D_n . There are also certain “relations” satisfied by these generators. (This use of the word “relation” is different from the definition of relation in II.1.) Clearly, we have $\alpha^n = \epsilon$ and $\beta^2 = \epsilon$. Also, since $\alpha\beta$ is a reflection, $(\alpha\beta)^2 = \alpha\beta\alpha\beta = \epsilon$. (Caution: recall that D_4 is not commutative; in particular, $\alpha\beta \neq \beta\alpha$ and $(\alpha\beta)^2 \neq \alpha^2\beta^2 = \alpha^2$.) Hence $\beta\alpha\beta = \alpha^{-1} = \alpha^{n-1}$. It can be shown that any relation in D_n can be derived from the three relations $\alpha^n = \epsilon$, $\beta^2 = \epsilon$, $\beta\alpha\beta = \alpha^{-1}$; i.e., the multiplication table for D_n can be filled in by using these three relations.

We now consider in some detail the symmetric groups S_n of permutations of the set $\{1, 2, \dots, n\}$. We assume for the remainder of this section that n is at least 2. As in §1, we will express an element $\sigma \in S_n$ by

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{pmatrix}.$$

(8.3) DEFINITION. A permutation $\sigma \in S_n$ is called a *cycle of length r* if there are r distinct integers a_1, a_2, \dots, a_r with $1 \leq a_i \leq n$ for $i = 1, 2, \dots, r$, such that

$$\sigma(a_1) = a_2, \sigma(a_2) = a_3, \dots, \sigma(a_{r-1}) = a_r, \sigma(a_r) = a_1$$

and $\sigma(j) = j$ for all other $j \in \{1, 2, \dots, n\}$. If σ is a cycle of length r as above, then we write

$$\sigma = (a_1, a_2, \dots, a_r).$$

(It is also quite common to omit the commas in the notation for a cycle.) Two cycles $\sigma = (a_1, a_2, \dots, a_r)$ and $\tau = (b_1, b_2, \dots, b_s)$ are called *disjoint* if $a_i \neq b_j$ for all i and j .

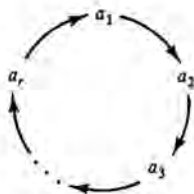


Figure 8.3

For example, the permutation

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}$$

is a cycle of length 3 and we could write $\sigma = (1, 3, 2)$ (or $\sigma = (2, 1, 3)$ or $\sigma = (3, 2, 1)$). Figure 8.3 may help to understand the action of the permutation (a_1, a_2, \dots, a_r) and why it is called a cycle of length r .

Note that the identity permutation ϵ could be written (1) (or (2) or (3) , etc.).

Even though S_n is not a commutative group if $n > 2$, we do have the next result.

(8.4) LEMMA. Two disjoint cycles commute.

PROOF. If $\sigma = (a_1, a_2, \dots, a_r)$ and $\tau = (b_1, b_2, \dots, b_s)$ are two disjoint cycles, then

$$\begin{aligned}\sigma\tau(a_i) &= \tau\sigma(a_i) = \begin{cases} a_{i+1} & \text{if } i < r; \\ a_1 & \text{if } i = r \end{cases} \\ \sigma\tau(b_j) &= \tau\sigma(b_j) = \begin{cases} b_{j+1} & \text{if } j < s; \\ b_1 & \text{if } j = s \end{cases} \\ \sigma\tau(m) &= \tau\sigma(m) = m \text{ if } m \neq a_i, b_j \text{ for all } i, j\end{aligned}$$

and we are done. ■

(8.5) LEMMA. Every permutation in S_n may be written as the product of disjoint cycles.

PROOF. It is easy to give an algorithm to write a given permutation as a product of cycles. Suppose $\sigma \in S_n$. Begin with 1 and find $\sigma(1), \sigma(\sigma(1)) = \sigma^2(1), \sigma^3(1)$, and so on. Since σ is 1-1 and $\{1, 2, \dots, n\}$ is finite, we must have $\sigma^{r_1}(1) = 1$ for some r_1 . When this happens, the cycle containing 1 has "closed up." If this cycle contains all the integers from 1 to n , then stop. If not, then start with the smallest positive integer not in this cycle, call it m_2 , and find $\sigma(m_2), \sigma^2(m_2), \dots$ until $\sigma^{r_2}(m_2) = m_2$. If all integers from 1 to n appear in one of the two cycles found so far, stop. If not, let m_3 be the smallest integer not appearing and continue the above process. Obviously, this algorithm terminates in at most n steps, and it is not hard to see that σ is the product of the disjoint cycles so produced. ■

► **(8.6) EXAMPLE.** Consider the permutation

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 6 & 2 & 8 & 1 & 7 & 5 \end{pmatrix} \in S_8.$$

Then $\sigma(1) = 3, \sigma^2(1) = \sigma(3) = 6, \sigma^3(1) = \sigma(6) = 1$, and this cycle has closed up. Now start with 2. We have $\sigma(2) = 4, \sigma^2(2) = \sigma(4) = 2$. Next, start with 5. Then $\sigma(5) = 8, \sigma^2(5) = 5$. Finally, $\sigma(7) = 7$. Hence we may write

$$\sigma = (1, 3, 6)(2, 4)(5, 8)(7).$$

Note that we could leave off the last cycle since $(7) = \epsilon$. However, it will be helpful in §10 if we get in the habit of writing all the cycles including the trivial ones. ◀

(8.7) LEMMA. The order of a permutation $\sigma \in S_n$ is the least common multiple of the lengths of the cycles when σ is written as a product of disjoint cycles.

PROOF. First, note that the order of a cycle $\tau = (a_1, a_2, \dots, a_r)$ of length r is r . Indeed, it is easy to see that $\tau^r(a_i) = a_i$ for all i and $\tau^k(a_1) = a_{k+1}$ for $1 \leq k < r$, so $\tau^r = \epsilon$ and $\tau^k \neq \epsilon$ for $1 \leq k < r$.

Now write $\sigma = \tau_1 \tau_2 \cdots \tau_m$, where the τ_j are disjoint cycles. Since disjoint cycles commute, we have

$$\sigma^k = \tau_1^k \tau_2^k \cdots \tau_m^k,$$

and since these cycles are disjoint $\sigma^k = \epsilon$ if and only if $\tau_j^k = \epsilon$ for $j = 1, \dots, m$. It then follows that the least exponent k such that $\sigma^k = \epsilon$ is the least common multiple of the orders (=lengths) of the τ_j . ■

► **(8.8) EXAMPLE.** The order of the permutation $\sigma = (1, 3, 6)(2, 4)(5, 8)(7)$ of Example (8.6) is 6 (= the least common multiple of 3, 2, 2, and 1). It would be a good exercise for the reader to check this by computing the permutations $\sigma^2, \sigma^3, \dots, \sigma^6$. ◀

► **(8.9) EXAMPLE.** In this example, we multiply three nondisjoint cycles to illustrate how to work with cycles. Let $\sigma = (1, 3, 5)(2, 1, 5)(1, 3, 4) \in S_5$. Then to find $\sigma(1)$ we start with the rightmost cycle and “follow 1 along.” Thus, the cycle $(1, 3, 4)$ takes 1 to 3; then the cycle $(2, 1, 5)$ leaves 3 fixed; finally, the cycle $(1, 3, 5)$ takes 3 to 5. Hence, in the composition, 1 goes to 5, so $\sigma(1) = 5$. To find $\sigma(2)$, we see that $(1, 3, 4)$ leaves 2 fixed, $(2, 1, 5)$ takes 2 to 1, and $(1, 3, 5)$ takes 1 to 3. So, we have $\sigma(2) = 3$. We leave it to the reader to see that

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 4 & 1 & 2 \end{pmatrix}.$$

If we wish to obtain the answer here as a product of disjoint cycles, then that can be done directly when we compute the composition. To do this, we start with 1 and as we’ve just seen $\sigma(1) = 5$. We then determine $\sigma(5)$ (by starting with the rightmost cycle and following 5 along), finding here that $\sigma(5) = 2$. Next we determine $\sigma(2)$, and so on until we find the integer whose image under σ is 1. This closes up a cycle. In this example, it turns out that σ is itself a cycle, namely $(1, 5, 2, 3, 4)$. If our first cycle had closed up and some of the integers from 1 to n had not appeared, then we would begin the next cycle by finding the image of the first positive integer that has not yet appeared, just as we did in the proof of Lemma (8.5). ◀

(8.10) **DEFINITION.** A cycle of length 2 is called a *transposition*.

(8.11) **LEMMA.** Every cycle is a product of transpositions.

PROOF. First, note that $\epsilon = (1, 2)(1, 2)$. To complete the proof, we just need to note that the cycle $\sigma = (a_1, a_2, \dots, a_r)$ may also be written

$$\sigma = (a_1, a_r)(a_1, a_{r-1}) \dots (a_1, a_3)(a_1, a_2).$$

(8.12) **COROLLARY.** Every permutation in S_n may be written as a product of transpositions.

PROOF. This follows from Lemmas (8.5) and (8.11). ■

Note that a representation as a product of transpositions is not unique. For a rather trivial example, note that we can write

$$\begin{aligned}\epsilon &= (1, 2)(1, 2) \\ &= (1, 2)(1, 2)(1, 2)(1, 2).\end{aligned}$$

(8.13) **DEFINITION.** A permutation is called *even* if it can be written as the product of an even number of transpositions and is called *odd* if it can be written as the product of an odd number of transpositions.

We need to see that this is a sound definition, in the sense that every permutation is either even or odd but not both.

(8.14) **LEMMA.** No permutation is both even and odd.

PROOF. Although we will not formally study polynomials until Chapter IV, we will assume, for this proof, that the reader has a familiarity with them. An alternative proof without using polynomials is outlined in Exercise 8.15.

Consider the polynomial P in the n indeterminates X_1, X_2, \dots, X_n given by

$$P = (X_1 - X_2)(X_1 - X_3) \cdots (X_{n-1} - X_n) = \prod_{i < j} (X_i - X_j).$$

Given a permutation $\sigma \in S_n$, we define $\sigma(P)$ to be the polynomial

$$\prod_{i < j} (X_{\sigma(i)} - X_{\sigma(j)}).$$

Since all this does to P is introduce a certain number of factors of (-1) , we have that $\sigma(P)$ is either P or $-P$ (and not both of course).

Now write $\sigma = \tau_1 \tau_2 \cdots \tau_m$ where the τ_j are transpositions. Then $\sigma(P) = \tau_1 \tau_2 \cdots \tau_m(P)$, and it is not too hard to see that if τ is a transposition, then $\tau(P) = -P$ (the factors of (-1) occur in pairs except for the one coming from the term involving both X_i and X_j — work out an example). Thus $\sigma(P) = P$ if σ can be written as a product of an even number of transpositions and $\sigma(P) = -P$ if σ can be written as an odd number of transpositions. Hence it cannot be possible to write a permutation both as the product of an even number of transpositions and as the product of an odd number of transpositions. ■

Note that the product of two even permutations is again even (since the sum of two even integers is even), the product of two odd permutations is even (since the sum of two odd integers is even), and the product of an even and an odd permutation is odd (since the sum of an even and an odd integer is odd).

(8.15) DEFINITION. Let A_n denote the set of all even permutations in S_n .

(8.16) PROPOSITION. Suppose $n > 1$. Then A_n is a normal subgroup of S_n of index 2.

PROOF. Let \tilde{G} be the group $\{-1, 1\}$ under multiplication. Define a map

$$\begin{aligned} \varphi : S_n &\longrightarrow \tilde{G} \\ \sigma &\longmapsto \begin{cases} 1 & \text{if } \sigma \text{ is even;} \\ -1 & \text{if } \sigma \text{ is odd.} \end{cases} \end{aligned}$$

To see that φ is a homomorphism, note that $\varphi(\sigma_1 \sigma_2) = -1$ if and only if one of the permutations σ_1 and σ_2 is odd and the other one is even (by the remark above about the product of two permutations), and this is true if and only if $\varphi(\sigma_1) \cdot \varphi(\sigma_2) = -1$. Also, notice that φ is an onto mapping since $\varphi(\epsilon) = 1$ and $\varphi((1, 2)) = -1$.

The identity of \tilde{G} is 1, so the kernel of φ is exactly the set A_n of even permutations. By Proposition (6.9), A_n is then a normal subgroup of S_n . By the First Isomorphism Theorem (6.12), we have

$$S_n/A_n \cong \tilde{G}.$$

Hence

$$2 = |S_n/A_n| = [S_n : A_n],$$

which completes the proof. ■

Note that since the index of A_n is 2, the two left cosets of A_n in S_n are A_n and the set of all odd permutations in S_n . Thus, by Lemma (4.5), there are $n!/2$ even permutations and $n!/2$ odd permutations. The group A_n is called the *alternating group on n letters*.

We now show that in studying the subgroups of the symmetric groups, one is actually studying all finite groups.

(8.17) THEOREM (Cayley). Every group (G, \cdot) is isomorphic to a subgroup of $(\text{Perm}(G), \circ)$. In particular, a finite group of order n is isomorphic to a subgroup of S_n .

PROOF. Suppose $a \in G$. We will associate to a the map

$$\begin{aligned} f_a : G &\longrightarrow G \\ g &\longmapsto ag. \end{aligned}$$

We need to see that f_a is a 1-1 and onto mapping. (Note: f_a is not a homomorphism unless $a = e$ since a homomorphism must take the identity to the identity.) To see that f_a is 1-1, note that if $f_a(g_1) = f_a(g_2)$, then $ag_1 = ag_2$, hence $g_1 = g_2$ by cancellation in G . To see that f_a is onto, note that for any $g \in G$, $f_a(a^{-1}g) = g$.

Now we define a map

$$\begin{aligned} \Phi : G &\longrightarrow \text{Perm}(G) \\ a &\longmapsto f_a. \end{aligned}$$

We need to see that Φ is a homomorphism and that Φ is 1-1 (for then G will be isomorphic to $\Phi(G)$, a subgroup of $\text{Perm}(G)$). We have

$$\begin{aligned} \Phi(ab)(g) &= f_{ab}(g) = (ab)g \\ &= a(bg) = f_a(f_b(g)) = (f_a \circ f_b)(g), \end{aligned}$$

hence $\Phi(ab) = \Phi(a) \circ \Phi(b)$ and Φ is a homomorphism. Finally, to see that Φ is 1-1, we compute its kernel. If $\Phi(a) = f_a$ is the identity map on G , then, in particular, $f_a(e) = ae$ must equal e . But since ae also equals a , we have $a = e$ and $\text{Ker}(\Phi) = \{e\}$, showing that Φ is 1-1. ■

► **(8.18) EXAMPLE.** In this example, we show that D_4 is isomorphic to a subgroup of S_4 and to a subgroup of S_8 .

First, to see that D_4 is isomorphic to a subgroup of S_4 (this was Exercise 1.6), we label the vertices of a square 1, 2, 3, and 4 as in Figure 1.2 and

we associate to each symmetry in D_4 the permutation in S_4 determined by the action of the symmetry on these vertices (as we described in §1). With notation as in §1, the map from D_4 to S_4 is given by

$$\begin{aligned}\epsilon &\mapsto (1)(2)(3)(4) \\ \alpha &\mapsto (1, 2, 3, 4) \\ \alpha^2 &\mapsto (1, 3)(2, 4) \\ \alpha^3 &\mapsto (1, 4, 3, 2) \\ \beta &\mapsto (1, 2)(3, 4) \\ \alpha\beta &\mapsto (1, 3)(2)(4) \\ \alpha^2\beta &\mapsto (1, 4)(2, 3) \\ \alpha^3\beta &\mapsto (1)(2, 4)(3)\end{aligned}$$

We leave it to the reader to check that this defines a homomorphism — this amounts to seeing that the way we compose symmetries “agrees” with the way we compose permutations (note, for example, that $\alpha\beta$ corresponds to the product of $(1,2,3,4)$ with $(1,2)(3,4)$).

The above representation of D_4 as a subgroup of S_4 is not the permutational representation one gets out of the proof of Cayley’s Theorem (8.17). There we consider the group D_4 as acting on itself, not on a square. What do we mean by D_4 acting on itself? Index the elements of D_4 as follows:

$$\begin{aligned}g_1 &= \epsilon, g_2 = \alpha, g_3 = \alpha^2, g_4 = \alpha^3 \\ g_5 &= \beta, g_6 = \alpha\beta, g_7 = \alpha^2\beta, g_8 = \alpha^3\beta.\end{aligned}$$

Let’s see to what permutation in S_8 the symmetry α corresponds. We see that

$$\begin{aligned}\alpha g_1 &= g_2, \alpha g_2 = g_3, \alpha g_3 = g_4, \alpha g_4 = g_1 \\ \alpha g_5 &= g_6, \alpha g_6 = g_7, \alpha g_7 = g_8, \alpha g_8 = g_5.\end{aligned}$$

We then associate to α the permutation $(1, 2, 3, 4)(5, 6, 7, 8) \in S_8$. We leave it to the reader to check (use Table (1.8)) that the homomorphism from D_4 into S_8 is given by

$$\begin{aligned}\epsilon &\mapsto (1)(2)(3)(4)(5)(6)(7)(8) \\ \alpha &\mapsto (1, 2, 3, 4)(5, 6, 7, 8) \\ \alpha^2 &\mapsto (1, 3)(2, 4)(5, 7)(6, 8) \\ \alpha^3 &\mapsto (1, 4, 3, 2)(5, 8, 7, 6) \\ \beta &\mapsto (1, 5)(2, 8)(3, 7)(4, 6) \\ \alpha\beta &\mapsto (1, 6)(2, 5)(3, 8)(4, 7) \\ \alpha^2\beta &\mapsto (1, 7)(2, 6)(3, 5)(4, 8) \\ \alpha^3\beta &\mapsto (1, 8)(2, 7)(3, 6)(4, 5)\end{aligned}$$

Note that the order of the corresponding permutation in S_8 is the same as the order of a symmetry in D_4 (as must be the case under an isomorphism).

It is often helpful to represent an abstract group concretely as a subgroup of permutations and, as this example shows, there may be more than one way to do this. ◀

EXERCISES

8.1. Show that α^3 and β also generate D_4 .

† 8.2. a) Determine the multiplication table for D_5 .

b) Find all subgroups of D_5 .

c) Viewing D_5 as the group of symmetries of a regular pentagon, represent D_5 as a subgroup of S_5 .

† 8.3. In S_4 , let

$$\rho = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix} \text{ and } \tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix}.$$

a) Find $\rho\tau$ and $\tau\rho$.

b) Find ρ^2, ρ^3, ρ^4 .

c) Find ρ^{-1} and τ^{-1} .

† 8.4. Write

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 2 & 4 & 5 & 1 & 6 & 8 & 10 & 3 & 9 & 7 \end{pmatrix}$$

as a product of disjoint cycles. What is the order of σ ?

† 8.5. Write $(1,2,4,5)(2,4,3)$ as a product of disjoint cycles.

8.6. Find the order of each element in S_4 by writing each of these permutations as a product of disjoint cycles.

† 8.7. In S_5 , let

$$\sigma = (1, 2, 4)(3, 5) \text{ and } \rho = (1, 3, 5, 2)(4).$$

a) Find $\sigma\rho$ and $\rho\sigma$.

b) Find the order of σ and the order of ρ .

c) Find σ^{-1} and ρ^{-1} .

8.8. a) Show that a cycle of length r is even if r is odd and is odd if r is even.

- b) Suppose that σ equals the product of (not necessarily disjoint) cycles $\rho_1, \rho_2, \dots, \rho_k$ where ρ_i has length l_i for $i = 1, 2, \dots, k$. Show that σ is even if and only if $l_1 + l_2 + \dots + l_k - k$ is even.
- 8.9. Prove that A_4 , which has order 12, has no subgroup of order 6 as follows:
- a) First show that if H is a subgroup of index 2 in a group G , then $g^2 \in H$ for every $g \in G$.
- b) Now show that at least 7 elements in A_4 are squares of elements in A_4 .
- 8.10. Show that $(1, 2)$ and $(1, 2, 3)$ generate S_3 .
- 8.11. a) Prove that $\sigma \in S_n$ is even if and only if σ^{-1} is even.
 b) Prove that there is no element $\sigma \in S_4$ such that $\sigma(1, 2)\sigma^{-1} = (1, 3)(2, 4)$.
- † 8.12. a) Let $\sigma \in S_n$. Show that

$$\sigma(a_1, a_2, \dots, a_r)\sigma^{-1} = (\sigma(a_1), \sigma(a_2), \dots, \sigma(a_r)).$$

(Hint: View $\sigma(a_1, a_2, \dots, a_r)\sigma^{-1}$ as the composition of three maps. See what this composition does to $\sigma(a_1)$.)

- b) Find an element $\sigma \in S_4$ such that $\sigma(1, 2, 3, 4)\sigma^{-1} = (1, 4, 3, 2)$.
 c) In S_4 , show that there is no element σ such that $\sigma(1, 2, 3)\sigma^{-1} = (1, 3)(2, 4)$.
- 8.13. The Klein four-group $G = \{e, a, b, c\}$ has the following multiplication table:

	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Show that G is isomorphic to a subgroup of S_4 .

- 8.14. Show that S_3 is isomorphic to a subgroup of S_6 .
- 8.15. In this exercise, we sketch another proof, due to W. Miller [25], of Lemma (8.14).
- a) Show that if some permutation were both even and odd, then the identity permutation ϵ would be odd (as well as even).
 b) Now suppose

$$\epsilon = \tau_1 \tau_2 \cdots \tau_m$$

(*)

where each τ_i is a transposition and m is odd. Let a be an integer occurring on the right side of $(*)$ and let j be the greatest integer such that a appears in τ_j . Show that we would reach a contradiction if $j = 1$.

- c) Now suppose $j > 1$ and consider the product $\tau_{j-1}\tau_j$. If $\tau_{j-1} = \tau_j$, then this product is ϵ and it can be deleted from $(*)$. We then would go back to part (b) and start with a shorter expression. If $\tau_{j-1} \neq \tau_j$, then show that $\tau_{j-1}\tau_j = \tau'_{j-1}\tau'_j$ with a appearing in τ'_{j-1} but not in τ'_j . (For example, if $\tau_{j-1}\tau_j = (c, d)(a, b)$ with $c, d \notin \{a, b\}$, then use the fact that $(c, d)(a, b) = (a, b)(c, d)$. You must also treat the cases when $c = a$ or $c = b$.)
- d) Now reason that by repeating the arguments above, one eventually reaches a contradiction.

9. Group Actions and Burnside's Theorem

We begin this section with a problem.

(9.1) PROBLEM. How many different squares are there with vertices colored red or blue?

At first glance, this may seem quite simple. There are four vertices and each can be colored in two different ways, so the answer should be $2^4 = 16$. But we need to make clear what we mean by “different” squares. We will call two colored squares “different” (or “distinct”) if one cannot be obtained from the other by performing a symmetry. Thus the two 2-colored squares in Figure 9.1 are different, but the two 2-colored squares in Figure 9.2 are not different.



Figure 9.1



Figure 9.2

Now that we have made it clear what we mean in Problem (9.1), it is not too hard to see that there are only six different 2-colored squares when we take symmetry into account (see Example (9.7)). We will now develop a general method for handling problems such as this one. What happened above was that the group of symmetries of the square (i.e., the group D_4) “acted” on the set of 2-colored squares, and two 2-colored squares were viewed as equivalent if they differed by a symmetry. This leads us to consider the general situation of a group acting on a set.

(9.2) DEFINITION. Suppose (G, \circ) is a group and X is a set. A *left action of G on X* is a mapping

$$\begin{aligned} G \times X &\longrightarrow X \\ (g, x) &\longmapsto gx \end{aligned}$$

(i.e., we denote the image of (g, x) under this mapping by gx) such that

$$g_1(g_2x) = (g_1 \circ g_2)x \text{ and } ex = x$$

for all $x \in X$ and $g_1, g_2 \in G$. (Here, e denotes the identity in G .)

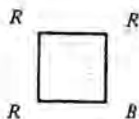
Similarly, one defines a right action of G on X as a map $X \times G \rightarrow X$ such that $(xg_1)g_2 = x(g_1 \circ g_2)$ and $xe = x$ for all $x \in X$ and $g_1, g_2 \in G$. Although we will be dealing with left actions throughout this section, all our results are also valid for right actions.

► (9.3) EXAMPLES

1) In Problem (9.1), the set X is the set of 2-colored squares. The group G is the group D_4 of symmetries of the square. If $x \in X$ and $\gamma \in D_4$, then γx is the 2-colored square obtained by performing the symmetry γ on the square x .

We will denote a colored square by an ordered 4-tuple, where the i^{th} coordinate is the color of the i^{th} vertex and we number the vertices as in

Figure 1.2. Thus, for example, the 4-tuple (R, R, B, R) will stand for the square



For a specific example of the group action, if $x = (R, R, B, R)$ and α is rotation by 90° (in the counterclockwise direction), then $\alpha x = (R, R, R, B)$.

2) Another example of a group action is the group $\text{Perm}(X)$ acting on a set X ; for example, the group S_n acting on the set $\{1, 2, \dots, n\}$. Here, if $a \in \{1, 2, \dots, n\}$ and $\rho \in S_n$, then, of course, $\rho a = \rho(a)$. (We note that this is a left action by the way that we defined the operation on S_n . If we had defined the product of two permutations σ and τ by $\sigma\tau(a) = \tau(\sigma(a))$, then S_n would act on the right on $\{1, 2, \dots, n\}$.) ◀

Now we need to make precise the notion of two elements of X being equivalent when we take the action of the group into account.

(9.4) DEFINITION. Suppose the group G acts on the left on the set X . We say that x is G -equivalent to y , where $x, y \in X$, if there exists $g \in G$ such that $gx = y$.

(9.5) LEMMA. G -equivalence is an equivalence relation on X .

PROOF. Since $ex = x$ we have that x is G -equivalent to itself for every $x \in X$. Now suppose that x is G -equivalent to y . Then there exists $g \in G$ such that $y = gx$. Hence $g^{-1}y = g^{-1}(gx) = (g^{-1} \circ g)x = ex = x$ and y is G -equivalent to x . Finally, suppose that x is g -equivalent to y and y is G -equivalent to z . Then there exists $g_1, g_2 \in G$ such that $y = g_1x$ and $z = g_2y$. Hence we have $z = g_2(g_1x) = (g_2 \circ g_1)x$, so x is G -equivalent to z . ■

(9.6) DEFINITION. The equivalence classes of the equivalence relation “ G -equivalence” are called the *orbits* of G on X . We say that G acts *transitively* on X if X itself is the only orbit of G .

Thus, G acts transitively on X if for each pair of elements x and $y \in X$, there exists some $g \in G$ such that $y = gx$. It is obvious that the group S_n acts transitively on the set $X = \{1, 2, \dots, n\}$.

Recall that since the orbits are equivalence classes of an equivalence relation, they define a partition of X . It should be clear that when we asked for the number of “different” 2-colored squares in Problem (9.1), we were

asking for the number of distinct orbits of the action of D_4 on the set of all sixteen 2-colored squares.

► (9.7) EXAMPLES

1) It is not hard to see that the six distinct orbits of D_4 on the set of all 2-colored squares are:

$$\mathcal{O}_1 = \{(R, R, R, R)\}$$

$$\mathcal{O}_2 = \{(R, R, R, B), (R, R, B, R), (R, B, R, R), (B, R, R, R)\}$$

$$\mathcal{O}_3 = \{(R, R, B, B), (R, B, B, R), (B, B, R, R), (B, R, R, B)\}$$

$$\mathcal{O}_4 = \{(R, B, R, B), (B, R, B, R)\}$$

$$\mathcal{O}_5 = \{(R, B, B, B), (B, R, B, B), (B, B, R, B), (B, B, B, R)\}$$

$$\mathcal{O}_6 = \{(B, B, B, B)\}.$$

2) Let X be the set of all ordered pairs (a, b) where $a, b \in \{1, 2, 3\}$. Define an action of S_3 on X by $\rho(a, b) = (\rho(a), \rho(b))$. Then the orbits of S_3 on X are

$$\mathcal{O}_1 = \{(1, 1), (2, 2), (3, 3)\}$$

$$\mathcal{O}_2 = \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\}. \blacktriangleleft$$

(9.8) DEFINITION. Suppose a group (G, \circ) acts on the left on a set X . If $x \in X$, then the *stabilizer* of x , denoted G_x , is the set of all $g \in G$ that leave x fixed; i.e., $G_x = \{g \in G : gx = x\}$.

(9.9) LEMMA. G_x is a subgroup of G .

PROOF. First note that G_x is nonempty since e leaves each element of X fixed by Definition (9.2). Now, suppose $g, h \in G_x$. We will show that $g \circ h^{-1} \in G_x$. We have

$$\begin{aligned} (g \circ h^{-1})x &= (g \circ h^{-1})hx && \text{since } h \in G_x \\ &= g(h^{-1} \circ h)x && \text{using Definition (9.2)} \\ &= g(ex) = gx \\ &= x && \text{since } g \in G_x. \end{aligned}$$

Thus $g \circ h^{-1} \in G_x$ and we are done. ■

► (9.10) EXAMPLES

1) Consider the action of D_4 on the set of 2-colored squares. The stabilizer of (R, R, R, B) is $\{e, \alpha\beta\}$. (See §1 for the definition of $\alpha\beta$.) The stabilizer of (R, B, R, B) is $\{e, \alpha^2, \alpha\beta, \alpha^3\beta\}$. 130

2) Consider the action of S_3 on the set $\{1, 2, 3\} \times \{1, 2, 3\}$ in Example (9.6.2). The stabilizer of the ordered pair $(1, 1)$ is

$$\left\{ \epsilon, \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \right\}.$$

The stabilizer of the ordered pair $(1, 2)$ is $\{\epsilon\}$, since the permutation must leave both 1 and 2 fixed. ◀

(9.11) PROPOSITION. Suppose a finite group G acts on a finite set X . Let $x \in X$. Then $[G : G_x]$, the index in G of the stabilizer of x , is equal to the number of elements in the orbit containing x .

PROOF. Before giving the proof, let's look at an example above. Note that in Example (9.7.2), the orbit of the ordered pair $(1, 1)$ contained three elements and the orbit of the ordered pair $(1, 2)$ contained six elements. And in Example (9.10.2), we found that the order of the stabilizer of the ordered pair $(1, 1)$ was 2 and the order of the stabilizer of the ordered pair $(1, 2)$ was 1. Hence, by Lagrange's Theorem (and the fact that $|S_3| = 6$), the index of the stabilizer of the ordered pair $(1, 1)$ is 3 and the index of the stabilizer of the ordered pair $(1, 2)$ is 6.

Now for the proof. Suppose the orbit containing x is

$$\mathcal{O} = \{x = x_1, x_2, \dots, x_m\}.$$

By the definition of orbit, there exist $g_1, g_2, \dots, g_m \in G$ such that $g_i x = x_i$ for $i = 1, 2, \dots, m$. We claim that the distinct left cosets of G_x in G are

$$g_1 G_x, g_2 G_x, \dots, g_m G_x.$$

First, let's see that any element of G is in one of these cosets. If $g \in G$, then, by the definition of orbit, $gx = x_i$ for some $i \in \{1, 2, \dots, m\}$. Hence, $gx = g_i x$, so we have $(g_i^{-1} \circ g)x = x$. But this says that $g_i^{-1} \circ g \in G_x$, hence that $g \in g_i G_x$.

Finally, we need to see that these cosets are distinct. Suppose

$$g \in g_i G_x \cap g_j G_x.$$

Then there exist $h, h' \in G_x$ such that $g = g_i \circ h = g_j \circ h'$. But then

$$\begin{aligned} x_i &= g_i x = g_i(hx) \text{ since } h \in G_x \\ &= (g_i \circ h)x = (g_j \circ h')x \\ &= g_j(h'x) = g_j x = x_j. \end{aligned}$$

This implies that $i = j$, since x_i and x_j are distinct if $i \neq j$. Hence, the cosets $g_1 G_x, \dots, g_m G_x$ are pairwise disjoint, and the number of distinct left cosets equals m , the number of elements in the orbit containing x . ■

(9.12) COROLLARY. With assumptions as in Proposition (9.11), suppose y is in the same orbit as x . Then $|G_x| = |G_y|$.

PROOF. From Lagrange's Theorem, we have

$$|G| = [G : G_x]|G_x| = [G : G_y]|G_y|.$$

Then $|G_x| = |G_y|$ since Proposition (9.11) shows that $[G : G_x] = [G : G_y]$. ■

(9.13) DEFINITION. Suppose the group G acts on the left on the set X . Let $g \in G$. The *fixed set of g* , denoted X_g , is the set of all $x \in X$ such that g leaves x fixed; i.e., $X_g = \{x \in X : gx = x\}$.

The reader should stop and understand the distinction between G_x and X_g . Note that G_x is a subgroup of G and X_g is a subset of X .

- **(9.14) EXAMPLE.** Consider the action of D_4 on the set X of 2-colored squares. Then the fixed set of α , where α denotes rotation by 90° , is $X_\alpha = \{(R, R, R, R), (B, B, B, B)\}$. The fixed set of β , where β denotes reflection about the horizontal axis is

$$X_\beta = \{(R, R, R, R), (R, R, B, B), (B, B, R, R), (B, B, B, B)\}. \blacktriangleleft$$

The next key result relates the number of distinct orbits of a group G acting on a set X with the number of elements in each X_g , where $g \in G$. The number of distinct orbits is what we are interested in in problems like (9.1), and it is usually easier to determine the number of elements in each X_g than to determine the number of distinct orbits directly. In the next section, we will see that the number of elements in each X_g can be related to the cycle decomposition of g when we represent G as a subgroup of a permutation group.

(9.15) THEOREM (Burnside). Suppose a finite group G acts on a finite set X and let N denote the number of distinct orbits of G on X . Then

$$N = \frac{1}{|G|} \sum_{g \in G} |X_g|,$$

where $|X_g|$ denotes the number of elements in X_g .

PROOF. The proof will amount to counting the same thing in two different ways. Suppose $G = \{g_1, g_2, \dots, g_r\}$ and $X = \{x_1, x_2, \dots, x_s\}$. Form the $r \times s$ matrix $A = [a_{ij}]$ where

$$a_{ij} = \begin{cases} 1 & \text{if } g_i x_j = x_j; \\ 0 & \text{if } g_i x_j \neq x_j. \end{cases}$$

Note that this is the matrix of the relation R from G to X defined by gRx if and only if $gx = x$ (cf. II.2). We will now count the number of 1's in this matrix. The number of 1's in the i^{th} row is just $|X_{g_i}|$, so the total number of 1's in the matrix A is

$$\sum_{g \in G} |X_g|.$$

Now we count by columns. The number of 1's in the j^{th} column is $|G_{x_j}|$. Consider the total of the 1's in the columns corresponding to elements in the k^{th} orbit \mathcal{O}_k . Using Corollary (9.12), we see that this total is $|\mathcal{O}_k| \cdot |G_x|$, where x is any element in \mathcal{O}_k . By Proposition (9.11) and Lagrange's Theorem, we have

$$|\mathcal{O}_k| \cdot |G_x| = [G : G_x] \cdot |G_x| = |G|.$$

Since the columns corresponding to each orbit contribute $|G|$ 1's and there are N (disjoint) orbits, the total number of 1's in the matrix A must be $N \cdot |G|$. Hence, we have

$$|G| = \frac{1}{N} \sum_{g \in G} |X_g|,$$

and we are done. ■

► (9.16) EXAMPLES

1) We return to Problem (9.1) and solve it using Burnside's Theorem. We have the following table giving the number of elements in the fixed set corresponding to each element of D_4 :

g	$ X_g $
ϵ	16
α	2
α^2	4
α^3	2
β	4
$\alpha\beta$	8
$\alpha^2\beta$	4
$\alpha^3\beta$	8
Total	48

The entries in the table are easy to compute. For example, consider $\alpha\beta$, a reflection about a diagonal. For a 2-colored square to be left fixed under this symmetry, the vertices that lie on the diagonal can be any color (four possibilities), while the vertices not on this diagonal must be either both red or both blue. Thus eight 2-colored squares are left fixed by $\alpha\beta$. The other computations are similar. It follows from Burnside's Theorem that the number of different 2-colored squares (i.e., distinct orbits) is $48/8 = 6$.

2) Suppose a wheel is evenly divided into five compartments. Each compartment is to be painted either red, white, or blue. The back of the wheel is black. How many different (taking symmetry into account) such wheels are there?

Note that since the back of the wheel is black, the wheel cannot be flipped over. Therefore, the group of symmetries of such a wheel is just the cyclic group of order 5 generated by a rotation by 72° . Call this generator γ . How many such wheels will γ leave fixed? Three — only the wheels that are all red or all white or all blue. It is easy to see that the fixed set of γ^i for $i = 2, 3, 4$ is the same as the fixed set of γ . (The fact that 5 is prime plays a key role here.) The identity $\gamma^0 (= \gamma^5)$ obviously leaves all of the $3^5 = 243$ possible color wheels fixed. Therefore, by Burnside's Theorem the number of different color wheels (i.e., distinct orbits) is $255/5 = 51$. ◀

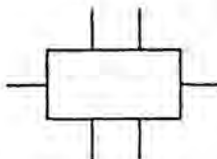
EXERCISES

- 9.1. a) Find the number of different squares with vertices colored red, white, or blue.
b) Find the number of different m -colored squares for any m .
- † 9.2. Find the number of different regular pentagons with vertices colored red, white or blue.
- 9.3. Find the number of different regular hexagons with vertices colored red or blue.
- † 9.4. A wheel is divided evenly into four compartments. Each compartment can be painted red, white, or blue. The back of the wheel is black. How many different such color wheels are there?
- 9.5. A wheel is divided evenly into six different compartments. Each compartment can be painted red or white. The back of the wheel is black. How many different such color wheels are there?
- † 9.6. A benzene ring consists of six carbon atoms at the vertices of a regular hexagon with a hydrogen atom bonded to each of the carbons. How

many different compounds can be made by replacing some (or none or all) of the hydrogen atoms by chlorine atoms? (The group of symmetries is D_6 — don't worry about single or double bonds between the carbon atoms.)

- † 9.7. a) A rectangular necktie is divided evenly into five bands and each band may be colored green, red, or blue. How many different neckties are there? (The group of symmetries is S_2 .)
 b) Answer (a) if we make the additional assumption that two adjacent bands must be colored differently.

† 9.8. A certain rectangular electrical relay box appears as shown below:



If each wire may be red or black, how many such boxes (taking symmetry into account) are there? If each wire may be red, black, or white, how many such boxes are there?

- † 9.9. Let X be the set of all binary words (or strings) of length 4 (so X has 16 elements). Let G be the cyclic subgroup of S_4 generated by the cycle $\rho = (1, 2, 3, 4)$. Define an action of G on X by

$$\rho x_1 x_2 x_3 x_4 = x_4 x_1 x_2 x_3.$$

(Then $\rho^2 \hat{x} = \rho(\rho \hat{x})$ and $\rho^3 \hat{x} = \rho(\rho^2 \hat{x})$.) Find the number of distinct orbits of this action of G on X .

- 9.10. Suppose a finite group G acts transitively on a finite set X . Prove that the number of elements in X divides the order of G .

10. Pólya Enumeration Theory

In this section, we present the elements of Pólya Enumeration Theory. The Hungarian mathematician Georg Pólya (1888–1985) developed this theory to handle difficult counting problems such as determining the numbers of different isomers of chemical compounds (especially hydrocarbons). Pólya realized that if a group of permutations G acts on a set X , then the number

of elements in X_g , $g \in G$, depends on the representation of g as the product of disjoint cycles. He also saw how this cycle representation could be used to yield representatives for the orbits. Many of Pólya's ideas appeared earlier in a paper of J.H. Redfield, but this article was evidently unknown to Pólya.

Suppose

$$C = \{c_1, c_2, \dots, c_n\}$$

$$R = \{r_1, r_2, \dots, r_m\}$$

are two finite sets. (In Problem (9.1), C would be the vertices of the square and R would be the colors red and blue.) Suppose G is a subgroup of $\text{Perm}(C) = S_n$. Let R^C denote the set of all mappings from C to R . (If C is the set of vertices of a square and R is a set of m colors, then a function in R^C is an m -coloring of the square.) We can define a right action of G on R^C by

$$\begin{aligned} R^C \times G &\longrightarrow R^C \\ (f, \rho) &\longmapsto f \circ \rho; \end{aligned}$$

i.e., $f\rho$ is the composition of f with ρ .

Put $X = R^C$. We want to relate the number of elements in X_ρ to the representation of ρ as a product of disjoint cycles. Write

$$\rho = (c_{11}, c_{12}, \dots, c_{1q_1})(c_{21}, c_{22}, \dots, c_{2q_2}) \cdots (c_{p1}, c_{p2}, \dots, c_{pq_p})$$

where the cycles on the right are disjoint and give a partition of C . (Note that we include cycles of length 1 in this decomposition.)

(10.1) LEMMA. Suppose $f \in X$. Then $f \in X_\rho$ if and only if $f(c_{ij}) = f(c_{ik})$ for $i = 1, 2, \dots, p$ and $1 \leq j, k \leq q_i$ (i.e., if and only if f is constant on each cycle of ρ).

PROOF. Suppose $f \in X_\rho$. Then $f \circ \rho = f$, hence $f \circ \rho^k = f$ for all $k \geq 0$. But c_1 and c_2 are in the same cycle of ρ if and only if $c_2 = \rho^k(c_1)$ for some $k \geq 0$. Therefore, if c_1 and c_2 are in the same cycle, then $f(c_2) = f(\rho^k(c_1)) = f(c_1)$.

Conversely, suppose that f is constant on each cycle of ρ . Then since c and $\rho(c)$ are in the same cycle of ρ for every $c \in C$, we have $f(c) = f(\rho(c))$ for all $c \in C$. Therefore $f = f \circ \rho$ and $f \in X_\rho$. ■

(10.2) DEFINITION. If $\rho \in S_n$, let $l(\rho)$ denote the number of cycles in the representation of ρ as a product of disjoint cycles, including cycles of length 1.

136

So, for example, if $\epsilon \in S_n$ is the identity, then $\epsilon = (1)(2) \cdots (n)$, so $l(\epsilon) = n$.

(10.3) COROLLARY

$$|X_\rho| = |R|^{l(\rho)}.$$

PROOF. By Lemma (10.1), the number of elements in X_ρ is the same as the number of functions in X that are constant on each cycle of ρ . There are $l(\rho)$ cycles of ρ , and a function constant on each cycle may take any of the $|R|$ values in R on each cycle. Therefore, the number of functions in R^C that are constant on each cycle of ρ is $|R|^{l(\rho)}$. ■

(10.4) COROLLARY. Let N denote the number of distinct orbits of the action of G on X . Then

$$N = \frac{1}{|G|} \sum_{\rho \in G} |R|^{l(\rho)}.$$

PROOF. This follows from Burnside's Theorem (9.15) and the preceding corollary. ■

► **(10.5) EXAMPLE.** We compute the number of different m -colorings of a square. The set C consists of the four vertices of the square, the set R is a set of m colors, and the group G is D_4 . We have the following table:

ρ	Cycle rep.	$l(\rho)$	$ R ^{l(\rho)}$
ϵ	(1)(2)(3)(4)	4	m^4
α	(1, 2, 3, 4)	1	m
α^2	(1, 3)(2, 4)	2	m^2
α^3	(1, 4, 3, 2)	1	m
β	(1, 2)(3, 4)	2	m^2
$\alpha\beta$	(1, 3)(2)(4)	3	m^3
$\alpha^2\beta$	(1, 4)(2, 3)	2	m^2
$\alpha^3\beta$	(1)(2, 4)(3)	3	m^3

Hence, from Corollary (10.4), the number of distinct orbits N is

$$\frac{1}{8}(m^4 + 2m^3 + 3m^2 + 2m).$$

For $m = 2$, we get $N = (16 + 16 + 12 + 4)/8 = 48/8 = 6$. For $m = 3$, we get $(81 + 54 + 27 + 6)/8 = 168/8 = 21$. ◀

We will now associate to each permutation in S_n a polynomial in n indeterminates. Although we will not formally treat polynomials until the next chapter, we assume a basic familiarity with them here.

(10.6) DEFINITION. If $\rho \in S_n$ has $d_i(\rho)$ cycles of length i , for $i = 1, 2, \dots, n$, in its representation as a product of disjoint cycles, then we associate to ρ the polynomial

$$P_\rho(X_1, X_2, \dots, X_n) = X_1^{d_1(\rho)} X_2^{d_2(\rho)} \dots X_n^{d_n(\rho)}$$

and we call this polynomial the *cycle index of ρ* . (Here, X_1, \dots, X_n are indeterminates; they are not fixed sets.) The *cycle index of G* is the polynomial

$$P_G(X_1, X_2, \dots, X_n) = \frac{1}{|G|} \sum_{\rho \in G} P_\rho.$$

For example, the cycle index of $\alpha^2 \in D_4$ is X_2^2 , while the cycle index of $\alpha^3\beta$ is $X_1^2X_2$. From the table in Example (10.5) one can see that the cycle index of D_4 , represented as a subgroup of S_4 , is

$$P_{D_4} = (X_1^4 + 2X_1^2X_2 + 3X_2^2 + 2X_4)/8.$$

(10.7) COROLLARY. With notation as in Corollary (10.4), $N = P_G(m, m, \dots, m)$; i.e., the number of distinct orbits can be obtained by substituting the number of elements in R for each of the indeterminates in the cycle index of G .

PROOF. This follows from Corollary (10.4) once we note that $l(\rho) = d_1(\rho) + d_2(\rho) + \dots + d_n(\rho)$. ■

(10.8) DEFINITION. Suppose $f \in R^C$. The *weight of f* , denoted $W(f)$, is defined by

$$W(f) = r_1^{e_1(f)} r_2^{e_2(f)} \dots r_m^{e_m(f)}$$

where $e_i(f)$ is the number of $c \in C$ such that $f(c) = r_i$, for $i = 1, 2, \dots, m$.

Going back to the situation of Problem (9.1), the weight of the colored square (R, R, R, B) is R^3B , while the colored squares (R, R, B, B) and (R, B, R, B) each have weight R^2B^2 . Note that the weight is a "formal" expression in r_1, r_2, \dots, r_m — we do not assume that there is a multiplication defined on the set R .

(10.9) LEMMA. Two elements in the same orbit of G on X have the same weight.

PROOF. Suppose f_1 and f_2 are in the same orbit. Then $f_1 = f_2 \circ \rho$ for some $\rho \in G$. But then f_1 and f_2 carry the same number of elements of C to

each $r_i \in R$ (although f_1 and f_2 may take different elements of C to a given element of R) since ρ just permutes the elements of C . Hence $e_i(f_1) = e_i(f_2)$ for $i = 1, 2, \dots, m$. ■

(10.10) DEFINITION. Suppose that W_1, W_2, \dots, W_s are all the possible weights of elements of R^C and let p_i denote the number of orbits such that each element of that orbit has weight W_i , $i = 1, 2, \dots, s$. Then

$$p_1 W_1 + p_2 W_2 + \dots + p_s W_s$$

is called the *pattern inventory* of G on R^C .

We now come to the main result of this section. For a proof of the next theorem, the reader may consult Roberts [29]. However, we note that the proof is not particularly enlightening, and the reader will gain much more insight into the theorem by working completely through an example similar to those we will do below.

(10.11) THEOREM (Pólya-Redfield). With notation given at the beginning of this section, put

$$Y_i = r_1^i + r_2^i + \dots + r_m^i$$

for $i = 1, 2, \dots, m$. Then

- 1) The weights of elements of R^C that are left fixed by $\rho \in G$ are the terms of $P_\rho(Y_1, Y_2, \dots, Y_m)$.
- 2) The pattern inventory of G on R^C is given by

$$P_G(Y_1, Y_2, \dots, Y_m).$$

► (10.12) EXAMPLES

- 1) We return to Problem (9.1). We have the following table:

ρ	P_ρ	Sum of weights of elements fixed by ρ
(1)(2)(3)(4)	X_1^4	$(R + B)^4$
(1, 2, 3, 4)	X_4	$R^4 + B^4$
(1, 3)(2, 4)	X_2^2	$(R^2 + B^2)^2$
(1, 4, 3, 2)	X_4	$R^4 + B^4$
(1, 2)(3, 4)	X_2^2	$(R^2 + B^2)^2$
(1, 3)(2)(4)	$X_1^2 X_2$	$(R + B)^2(R^2 + B^2)$
(1, 4)(2, 3)	X_2^2	$(R^2 + B^2)^2$
(1)(2, 4)(3)	$X_1^2 X_2$	$(R + B)^2(R^2 + B^2)$

As we noted after Definition (10.6), the cycle index of D_4 is

$$P_{D_4} = (X_1^4 + 2X_1^2X_2 + 3X_2^2 + 2X_4)/8.$$

Hence, by Theorem (10.11), the pattern inventory of D_4 on R^C is

$$\begin{aligned} & \frac{1}{8} [(R+B)^4 + 2(R+B)^2(R^2+B^2) + 3(R^2+B^2)^2 + 2(R^4+B^4)] \\ &= R^4 + R^3B + 2R^2B^2 + RB^3 + B^4. \end{aligned}$$

The terms in the pattern inventory are exactly the weights of representatives of the six distinct orbits of D_4 on R^C . (Recall that there were two orbits whose elements had weight R^2B^2 — one in which the two like-colored vertices were adjacent and one in which they were diagonally opposite.)

2) Consider the situation in Exercise 9.6. There we were concerned with the number of different compounds that could be obtained by replacing some of the hydrogen atoms in a benzene ring by chlorine atoms. If we ignore the type of bonds between carbon atoms, the group of symmetries here is D_6 . We may represent D_6 as a subgroup of S_6 and obtain the following table of cycle indices:

Symmetry	Cycle index
(1)(2)(3)(4)(5)(6)	X_1^6
(1,2,3,4,5,6)	X_6
(1,3,5)(2,4,6)	X_3^2
(1,4)(2,5)(3,6)	X_2^3
(1,5,3)(2,6,4)	X_3^2
(1,6,5,4,3,2)	X_6
(1,6)(2,5)(3,4)	X_2^3
(1,2)(3,6)(4,5)	X_2^3
(1,4)(2,3)(5,6)	X_2^3
(1)(4)(2,6)(3,5)	$X_1^2X_2^2$
(2)(5)(1,3)(4,6)	$X_1^2X_2^2$
(3)(6)(1,5)(2,4)	$X_1^2X_2^2$

The cycle index of D_6 is then

$$(X_1^6 + 3X_1^2X_2^2 + 4X_3^2 + 2X_2^3 + 2X_6)/12.$$

The number of different compounds when we have two choices of atoms (hydrogen or chlorine) to bond to the carbon atoms is then $(2^6 + 3 \cdot 16 + 4 \cdot 8 + 2 \cdot 4 + 2 \cdot 2)/12 = 13$. If we let h denote hydrogen and c denote chlorine, then, using (2) of Theorem (10.11), we see that the pattern inventory is

$$h^6 + h^5c + 3h^4c^2 + 3h^3c^3 + 3h^2c^4 + hc^5 + c^6.$$

3) For a final example we consider the group of rigid symmetries of a cube. We only consider the symmetries of a cube that are rotations about a line in 3-space. Thus we exclude, for example, reflections of 3-space through a plane going through diagonally opposite edges of the cube; these would carry the cube onto itself, but they could not be performed on a physical cube without breaking the cube.

The group of rotations of a cube is called the *octahedral group* (it is also the group of rotations of a regular octahedron); we will denote it by \mathcal{O} . This group has 24 elements. We will break up these rotations into the following seven classes:

Class I: the identity

Class II: 3 rotations by 90° with axes through the center of opposite faces

Class III: 3 rotations by 180° with axes through the center of opposite faces

Class IV: 3 rotations by 270° with axes through the center of opposite faces

Class V: 6 rotations by 180° with axes through the midpoints of opposite edges

Class VI: 4 rotations by 120° with axes through opposite vertices

Class VII: 4 rotations by 240° with axes through opposite vertices

It may help the reader to get a child's block or mark the faces of some cube in order to visualize these symmetries.

We may view the elements of the group \mathcal{O} as acting on either the faces, or vertices, or edges of the cube. This amounts to representing \mathcal{O} as a subgroup of S_6 or S_8 or S_{12} by numbering the faces or vertices or edges (respectively) and observing how each class of rotations permutes these objects. In the following table, we give the cycle index of each element of \mathcal{O} viewed as a subgroup of S_6 , S_8 , and S_{12} . The reader should verify the entries in the table. One way to find these cycle indices is to note which objects are left fixed by a given group element and the order of that group element. For example, consider the action on the vertices of a rotation by 120° with axis the line through a pair of opposite vertices. This group element obviously just leaves the two vertices on the axis fixed and has order 3 in the group \mathcal{O} . Since the order of a permutation is the least common multiple of the lengths of the cycles when we write the permutation as a product of disjoint cycles, this rotation, viewed as an element of S_8 , must equal the product of two cycles of length 3 and two cycles of length 1. Thus its cycle index is $X_1^2 X_3^2$. Now for the table:

	Cycle index on faces	Cycle index on vertices	Cycle index on edges
Identity	X_1^6	X_1^8	X_1^{12}
Class II	$X_1^2 X_4$	X_4^2	X_4^3
Class III	$X_1^2 X_2^2$	X_2^4	X_2^6
Class IV	$X_1^2 X_4$	X_4^2	X_4^3
Class V	X_2^3	X_2^4	$X_1^2 X_2^5$
Class VI	X_3^2	$X_1^2 X_3^2$	X_3^4
Class VII	X_3^2	$X_1^2 X_3^2$	X_3^4

The cycle index of \mathcal{O} acting on the set of faces is then

$$\frac{1}{24}(X_1^6 + 3X_1^2 X_2^2 + 6X_1^2 X_4 + 6X_2^3 + 8X_3^2).$$

Hence, by Corollary (10.7), the number of different m -colorings of the faces of a cube is

$$\frac{1}{24}(m^6 + 3m^4 + 12m^3 + 8m^2).$$

The cycle index of \mathcal{O} acting on the set of vertices is

$$\frac{1}{24}(X_1^8 + 8X_1^2 X_3^2 + 9X_2^4 + 6X_4^2).$$

The cycle index of \mathcal{O} acting on the set of edges is

$$\frac{1}{24}(X_1^{12} + 6X_1^2 X_2^5 + 3X_2^6 + 8X_3^4 + 6X_4^3). \blacktriangleleft$$

For applications of Pólya Enumeration Theory to graph theory, the reader may consult F. Harary and E. M. Palmer [13].

EXERCISES

- 10.1. a) Find the number of different (i.e., inequivalent) 2-colorings of the faces of a cube.
 b) If C is the set of faces of a cube and $R = \{\text{red, blue}\}$, find the pattern inventory of \mathcal{O} on R^C .

142

- † 10.2. a) Find the number of different 3-colorings of the faces of a cube.

- b) If the three colors are red, blue, and white, how many of the different cubes in (a) have 3 red faces, 2 blue faces, and 1 white face.
- 10.3. Determine the number of different 2-colorings, 3-colorings, and m -colorings of the vertices of a cube.
- 10.4. Determine the number of different 2-colorings, 3-colorings, and m -colorings of the edges of a cube.
- † 10.5. a) Determine the number of different 3-colorings of the vertices of a regular pentagon.
 b) Determine the pattern inventory of D_5 on the set of 3-colored (red, blue, white) regular pentagons.
 c) Determine the number of different m -colorings of a regular pentagon.
- † 10.6. Determine the number of different m -colorings of a regular hexagon.
- † 10.7. Prove that $m^6 + 3m^4 + 12m^3 + 8m^2$ must be divisible by 24 for any positive integer m .
- † 10.8. The group of rigid symmetries of a regular tetrahedron is called the *tetrahedral group*. We will denote it by \mathcal{T} . This group has 12 elements: the identity, 4 rotations by 120° with axes the lines through a vertex and the center of the opposite face, 4 rotations by 240° with the same axes, and 3 rotations by 180° with axes the lines through the midpoints of two opposite edges.
 a) Find the cycle index of \mathcal{T} as it acts on the 4 vertices of a regular tetrahedron.
 b) Find the cycle index of \mathcal{T} as it acts on the 4 faces of a regular tetrahedron.
 c) Find the cycle index of \mathcal{T} as it acts on the 6 edges of a regular tetrahedron.
- 10.9. Here is the type of problem to which Pólya applied his theory in his 1937 paper. Consider all molecules that have the “shape” of a regular tetrahedron with a carbon atom at the center of the tetrahedron and one of the following atoms or (chemical) groups at each of the vertices: H, Cl, CH_3 (methyl), or C_2H_5 (ethyl).
 a) Determine the number of different (not equivalent by a symmetry of \mathcal{T}) molecules of this form. (Use Exercise 10.8(a).)
 b) Of these different molecules how many have exactly 1 chlorine atom? How many have exactly 2 chlorine atoms?

For an application of Enumeration Theory to the study of gating networks and switching functions, see Exercise VI.3.8.

11. Conjugacy and the Class Equation

In this section, we return to abstract group theory and consider a useful action of a group on itself. This will lead to some nice results that are of a combinatorial nature.

Suppose that (G, \circ) is a finite group. Define a mapping from $G \times G$ to G by

$$(11.1) \quad (g, x) \mapsto g \circ x \circ g^{-1}.$$

We will show below that this defines an action of G on itself. Using the notation introduced in §9, we will then write gx for $g \circ x \circ g^{-1}$. Obviously, this may be confusing, so remember that in this section gx is *not* the product in G .

(11.2) **LEMMA.** (11.1) defines a left action of G on itself.

PROOF. For $g_1, g_2, x \in G$, we have

$$\begin{aligned} g_1(g_2x) &= g_1 \circ (g_2 \circ x \circ g_2^{-1}) \circ g_1^{-1} \\ &= (g_1 \circ g_2) \circ x \circ (g_1 \circ g_2)^{-1} \\ &= (g_1 \circ g_2)x. \end{aligned}$$

Also, $ex = e \circ x \circ e^{-1} = x$. Hence, (11.1) defines an action of G on G . ■

(11.3) **DEFINITION.** Suppose $x \in G$. An element $y \in G$ is called a *conjugate* (in G) of x if there exists $g \in G$ such that $y = g \circ x \circ g^{-1}$.

In other words, two elements are conjugate in G if they belong in the same orbit of the action of G on itself given by (11.1). From the theory of group actions developed in §9, we then know that conjugacy is an equivalence relation on G . The orbit of the above action containing an element x is usually called the *conjugacy class* of x . We will let C_x denote the conjugacy class of x . Of course, it is a subset of G and all the conjugacy classes define a partition of G . 144

(11.4) LEMMA. $C_x = \{x\}$ if and only if x belongs to the center $Z(G)$ of G .

PROOF, Exercise 11.1.

What is the stabilizer G_x of an element x under the above action? We have

$$\begin{aligned} G_x &= \{g \in G : gx = x\} \\ &= \{g \in G : g \circ x \circ g^{-1} = x\} \\ &= \{g \in G : g \circ x = x \circ g\}. \end{aligned}$$

This subgroup of G is called the *centralizer*, or *normalizer*, of x and appeared in Exercise 3.18.

(11.5) LEMMA. The index of the centralizer of x , $[G : G_x]$, is equal to the number of elements in G that are conjugate to x , $|C_x|$.

PROOF. This follows immediately from Proposition (9.10). ■

Since the conjugacy classes partition G and since the conjugacy classes consisting of precisely one element are exactly the classes of elements in the center $Z(G)$, if we let g_1, g_2, \dots, g_m be representatives of the conjugacy classes with more than one element, then we have

(11.6) PROPOSITION (The Class Equation).

$$|G| = |Z(G)| + \sum_{i=1}^m [G : G_{g_i}].$$

► **(11.7) EXAMPLE.** We find the class equation of S_3 . Using Exercise 8.12(a), the reader may verify that we have the following conjugacy classes:

$$\begin{aligned} C_e &= \{e\} \\ C_\tau &= \{\tau, \sigma\tau, \tau\sigma\} \\ C_\sigma &= \{\sigma, \sigma^2\}. \end{aligned}$$

The class equation of S_3 is then $6 = 1 + 3 + 2$. Note that $Z(S_3) = \{e\}$. ◀

Although it is not obvious from the preceding rather trivial example, the class equation can be used to derive some nontrivial results.

(11.8) PROPOSITION. Suppose G is a finite group of order p^n , where p is a prime. Then p divides $|Z(G)|$; in particular, the center $Z(G)$ is not trivial.

PROOF. If G is abelian, then $Z(G) = G$ and we are done. So we may assume that $Z(G) \neq G$. From the class equation, there exist g_1, g_2, \dots, g_m in G , but not in $Z(G)$, such that

$$(11.9) \quad p^n = |Z(G)| + \sum_{i=1}^m [G : G_{g_i}].$$

Now, since $g_i \notin Z(G)$, we have $G_{g_i} \neq G$. Therefore, $|G_{g_i}| = p^{r_i}$ for some $r_i < n$ and p must divide $[G : G_{g_i}]$ for $i = 1, 2, \dots, m$. But then p divides each term in (11.9) other than $|Z(G)|$, so p must divide $|Z(G)|$ also. ■

(11.10) PROPOSITION. A group G of order p^2 , where p is a prime, must be abelian.

PROOF. By Proposition (11.8), we know that the center of G is not trivial. Since the center is a subgroup, it follows from Lagrange's Theorem that the order of the center is then either p or p^2 . Suppose $|Z(G)| = p$ and let g be an element in G that is not in $Z(G)$. Then the centralizer of g , G_g , is a subgroup of G that contains $Z(G)$ and g , hence has order greater than p . But then G_g must have order p^2 , hence we have $G_g = G$. This says that $g \in Z(G)$, which is a contradiction. We are forced to conclude that the order of the center must be p^2 , which implies that $Z(G) = G$, showing that G is abelian. ■

We will close this chapter by stating, without proof, a theorem that completely classifies finite abelian groups. A proof of this theorem, as well as proofs of further results in group theory such as the Sylow Theorems (which deal with subgroups of prime power order), may be found in Herstein [16] or Lang [19].

(11.11) THEOREM (Fundamental Theorem of Finite Abelian Groups). Suppose G is a finite abelian group of order $n > 1$. Then G is isomorphic to a direct product (see Exercise 2.16)

$$\mathbf{Z}_{p_1^{r_1}} \times \mathbf{Z}_{p_2^{r_2}} \times \cdots \times \mathbf{Z}_{p_s^{r_s}},$$

where p_1, p_2, \dots, p_s are (not necessarily distinct) primes and r_1, r_2, \dots, r_s are positive integers (and where \mathbf{Z}_m denotes the additive group $(\mathbf{Z}_m, +)$).¹⁴⁶

Furthermore, this decomposition of G is unique in the sense that this is the only (up to order) direct product of groups of the form \mathbf{Z}_{p^r} , where p is prime, that is isomorphic to G .

Notice that the number of elements in the direct product in Theorem (11.11) is equal to $p_1^{r_1} p_2^{r_2} \cdots p_s^{r_s}$, hence we must have n equal to this product of prime powers (since isomorphic groups have the same order). However, this is not necessarily the prime factorization of n since the primes p_1, p_2, \dots, p_s may not be distinct. Finding the number of nonisomorphic abelian groups of order n then amounts to counting how many ways n can be written as a product of (not necessarily distinct) prime powers. This can be easily determined from the prime factorization of n .

(11.12) DEFINITION. If m is a positive integer, then the number of *partitions* of m , denoted $\pi(m)$, is the number of ways that m can be expressed as the sum of positive integers.

For example, we have $4 = 4, 4 = 3 + 1, 4 = 2 + 2, 4 = 2 + 1 + 1, 4 = 1 + 1 + 1 + 1$, so $\pi(4) = 5$.

(11.13) COROLLARY. If $n = q_1^{t_1} q_2^{t_2} \cdots q_k^{t_k}$ is the prime factorization of the positive integer n , then the number of nonisomorphic abelian groups of order n is $\pi(t_1)\pi(t_2) \cdots \pi(t_k)$.

PROOF. Suppose n is expressed as a product of prime powers

$$n = p_1^{r_1} p_2^{r_2} \cdots p_s^{r_s}$$

and suppose $p_{i_1}, p_{i_2}, \dots, p_{i_j}$ all equal q_j . Then by the uniqueness of prime factorization, we must have $r_{i_1} + r_{i_2} + \cdots + r_{i_j} = t_j$. The corollary then follows from Theorem (11.11). ■

► (11.14) EXAMPLES

1) By Proposition (11.10), every group of order 4 must be abelian. Since $4 = 2^2$ and $\pi(2) = 2$, Corollary (11.13) shows that there are precisely two nonisomorphic groups of order 4; namely, the cyclic group \mathbf{Z}_4 and the Klein four-group $\mathbf{Z}_2 \times \mathbf{Z}_2$. (Actually, this is not hard to show by just trying to fill in a “multiplication” table for a group of order 4).

2) There are three nonisomorphic abelian groups of order 120. They are

$$\mathbf{Z}_8 \times \mathbf{Z}_3 \times \mathbf{Z}_5, \mathbf{Z}_4 \times \mathbf{Z}_2 \times \mathbf{Z}_3 \times \mathbf{Z}_5 \text{ and } \mathbf{Z}_2 \times \mathbf{Z}_2 \times \mathbf{Z}_2 \times \mathbf{Z}_3 \times \mathbf{Z}_5.$$

The first of these groups is cyclic (and isomorphic to \mathbf{Z}_{120}), while the other two are not cyclic. Note that there are also noncommutative groups of order 120 (for example, S_5). ◀

EXERCISES

- 11.1. Prove Lemma (11.4).
- 11.2. a) Prove that if x and y are conjugate in a group, then so are x^k and y^k for any positive integer k .
 b) Prove that if x and y are conjugate in a finite group G , then the order of x in G equals the order of y in G .
- 11.3. Suppose that N is a normal subgroup of a group G and suppose $g \in N$. Show that all conjugates of g in G are in N .
- 11.4. Find all conjugacy classes in S_4 and verify the class equation.
- † 11.5. a) Use Exercise 8.12(a) to show that two permutations ρ and $\mu \in S_n$ are conjugate if and only if they have the same cycle index (cf. Definition (10.6)).
 b) Show that the number of distinct conjugacy classes in S_n equals $\pi(n)$, the number of partitions of n (see Definition (11.12)).
- † 11.6. Find two even permutations in S_5 that are conjugate in S_5 , but not conjugate in A_5 .
- † 11.7. Suppose there exists an element x in a finite group G of order > 2 such that the conjugacy class of x contains exactly two elements. Show that G has a nontrivial normal subgroup. (Hint: Use Exercise 5.4.)
- 11.8. Let G be a nonabelian group of order p^3 , where p is a prime (e.g., D_4). Show that the center of G must have order p . (Hint: Reason as in Proposition (11.10).)
- 11.9. In Exercise 5.16 we sketched a proof of Cauchy's Theorem for abelian groups. Now we ask you to drop the assumption that G is abelian and prove Cauchy's Theorem: If G is a finite group such that a prime p divides $|G|$, then G has an element (and hence a subgroup) of order p . (Hint: By an inductive argument, reduce to the case that p does not divide any proper subgroup of G . Now use the class equation to conclude that $Z(G)$ must be G .)
- 11.10. Find, up to isomorphism, all abelian groups of order:

- a) 8
- b) 21
- c) 72

- 11.11.** Suppose n is a positive integer such that n is not divisible by the square of any prime. Show that an abelian group of order n must be cyclic (cf. Exercise 5.17).
- 11.12.** If G is a finite *abelian* group and if m divides $|G|$, then use the Fundamental Theorem of Finite Abelian Groups to show that G has a subgroup of order m . (Hint: Consider the prime factorization of m and use Exercise 3.10. Note that Exercise 8.9 shows that the result in this exercise does not hold if we drop the “abelian” assumption.)

CHAPTER IV

Rings and Fields

In this chapter, we will study rings. A ring is a set with two algebraic operations defined on it that satisfy certain properties. The main models for rings are the integers and the ring of $n \times n$ matrices. We will use some of our theory to prove the Chinese Remainder Theorem, which is used to implement “fast adders” in computers and to describe the RSA public key cryptosystem, a clever idea for sending secret messages. We will also consider fields, which are (commutative) rings with the property that the nonzero elements form a group under multiplication. Examples of fields are the rational numbers and the real numbers. We will see that the ring of polynomials in one indeterminate with coefficients in a field has many properties in common with the ring of integers. Finally, we will construct and study fields with a finite number of elements. These fields have applications to combinatorial problems, and we will consider two applications, Latin squares and Steiner triples. Finite fields also play a key role in algebraic coding theory (Chapter V).

1. Definitions

We begin with the central definition of this chapter.

(1.1) DEFINITION. A *ring* $(R, +, \cdot)$ is a set R with two binary operations $+$ and \cdot on R such that

- 1) $(R, +)$ is a commutative group with identity element denoted by 0.
- 2) (R, \cdot) is a semigroup.

- 3) (Distributive laws) For all $a, b, c \in R$, we have $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(a + b) \cdot c = a \cdot c + b \cdot c$.

By standard abuse of language, we will usually refer to a ring R (instead of $(R, +, \cdot)$). The main models for rings are the integers and the ring of $n \times n$ matrices (see (1.5) below for more examples). We call $+$ “addition” in R and we call \cdot “multiplication” in R . The additive identity 0 is, of course, called “zero.” The distributive laws say that “multiplication distributes over addition.” Note that we do not assume that multiplication is a commutative operation, so we really need to state both the left and right distributive laws in our definition.

(1.2) NOTATION. We will write ab in place of $a \cdot b$.

Before giving definitions of rings with certain special properties, we prove the following lemma, which may seem obvious, but which requires proof. Other results of a similar elementary nature will appear in the exercises.

(1.3) LEMMA. $a0 = 0a = 0$ for all $a \in R$.

PROOF. We have

$$\begin{aligned} 0 + a0 &= a0 = a(0 + 0) && \text{since } 0 \text{ is the additive identity} \\ &= a0 + a0 && \text{Distributive Law} \end{aligned}$$

Hence, by cancellation in the group $(R, +)$, we conclude that $a0 = 0$. The proof that $0a = 0$ is entirely similar. ■

(1.4) DEFINITIONS. Suppose $(R, +, \cdot)$ is a ring.

- 1) If there exists an identity element for multiplication, then R is called a *ring with identity*. In such a case, we usually denote the multiplicative identity by “1.”
- 2) If \cdot is a commutative operation, then R is called a *commutative ring*.
- 3) Suppose R is a ring with identity. An element in R that has a multiplicative inverse is called a *unit*.
- 4) A *field* is a commutative ring with identity such that every element not equal to 0 is a unit. Equivalently, a commutative ring $(R, +, \cdot)$ is a field if the nonzero elements of R form a group under multiplication.
- 5) An element a in R is called a *zero divisor* if there exist elements $b \neq 0$ and $c \neq 0$ in R such that $ab = 0 = ca$. (Here, b may equal c .)

We note that 0 is a zero divisor according to this definition; some authors prefer not to call 0 a zero divisor.

- 6) An *integral domain* is a commutative ring with identity element $1 \neq 0$ that has no zero divisors except 0. (We require $1 \neq 0$ so that the ring with 0 as its only element is not called an integral domain.)

Now it's time for lots of examples to make these definitions concrete.

► (1.5) EXAMPLES

1) $(\mathbf{Z}, +, \cdot)$ is an integral domain. It is not a field; the only elements with multiplicative inverses are 1 and -1 .

2) $(M(n, \mathbf{R}), +, \cdot)$, the set of $n \times n$ real matrices with the operations of addition and multiplication of matrices is a ring with identity (the identity matrix). As is well known, it is not a commutative ring. Also, this ring has zero divisors other than 0. For example,

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The units in $M(n, \mathbf{R})$ are the $n \times n$ matrices with nonzero determinant (since it is shown in linear algebra courses that these are exactly the matrices with a multiplicative inverse).

3) The important class of rings $(\mathbf{Z}_n, +_n, \cdot_n)$ will be studied in the next section. We will show there that \mathbf{Z}_n is a field if and only if n is prime. If n is not prime, then \mathbf{Z}_n has zero divisors other than 0.

4) \mathbf{Q} , \mathbf{R} , and \mathbf{C} are all examples of fields.

5) Polynomial rings will be studied in §6, but for now let's look informally at all polynomials in one indeterminate X with integer coefficients. This set together with the usual operations of addition and multiplication of polynomials forms a ring with identity (the constant polynomial 1), which is denoted $\mathbf{Z}[X]$. We will see in §6 that this ring is an integral domain.

6) Let S be any set and let $\mathcal{P}(S)$ denote the set of all subsets (the power set) of S . The operations that first come to mind on $\mathcal{P}(S)$ are union and intersection (\cup and \cap). However, $\mathcal{P}(S)$ does not form a group under either of these operations (the Inverse Axiom fails in general). Define an operation \oplus , called "symmetric difference," on $\mathcal{P}(S)$ by

$$A \oplus B = \{s \in S : s \in A \text{ or } s \in B \text{ but } s \notin A \cap B\}$$

152

We leave it as an exercise to see that $(\mathcal{P}(S), \oplus, \cap)$ is a commutative ring with identity. The 0 in this ring is \emptyset . ◀

Suppose R is a ring and a, b , and c are nonzero elements of R such that $ab = ac$. If R is a field, then the nonzero elements form a group under multiplication, so we could apply the cancellation law in a group to conclude that $b = c$. But we now show that a multiplicative cancellation law holds even if R is just an integral domain. (We leave it as an exercise to show that any field is an integral domain.)

(1.6) PROPOSITION. Suppose R is an integral domain. If a, b , and $c \in R$ with $a \neq 0$ such that $ab = ac$, then $b = c$.

PROOF. We have

$$\begin{aligned} 0 &= ac - ab \\ &= a(c - b) \quad (\text{Distributive Law}) \end{aligned}$$

Since R has no zero divisors other than 0 and since $a \neq 0$, we may conclude that $c - b = 0$, so $c = b$. ■

(1.7) PROPOSITION. A finite integral domain is a field.

PROOF. Suppose $R = \{r_1 = 0, r_2 = 1, r_3, \dots, r_n\}$ is an integral domain. We must show that each nonzero element has a multiplicative inverse. Suppose r is a nonzero element in R and consider the elements rr_1, rr_2, \dots, rr_n . By the preceding proposition, these n elements are distinct, hence they are all the elements of R (by the Pigeonhole Principle). Therefore, $rr_k = 1$ for some k . Since R is commutative, we also have $r_k r = 1$, so $r_k = r^{-1}$. ■

We now define integer coefficients and exponents in an arbitrary ring. Although these definitions are redundant here since they were given in III.2 for semigroups, we repeat them for the sake of clarity.

(1.8) DEFINITION. Suppose R is a ring and $r \in R$. Then define

- 1) $0r = 0$ (here, the "0" on the left is the integer zero and the "0" on the right is the additive identity of R).
- 2) If $n \in \mathbf{P}$, then

$$nr = \underbrace{r + r + \cdots + r}_{n \text{ times}}$$

- 3) If $n \in \mathbf{P}$, then $(-n)r = n(-r)$ (here, $-r$ denotes the additive inverse of r and $n(-r)$ makes sense by (2)).
- 4) $r^1 = r$; if R has an identity, then $r^0 = 1$ (the multiplicative identity of R).

5) If $n \in \mathbf{P}$, then

$$r^n = \underbrace{r \cdot r \cdots r}_{n \text{ times}}$$

6) If $n \in \mathbf{P}$ and r has a multiplicative inverse r^{-1} , then $r^{(-n)} = (r^{-1})^n$.

EXERCISES

† 1.1. Let R be a ring and suppose $a, b \in R$.

- Show that $a(-b) = (-a)b = -(ab)$. (Hint: Show that $a(-b)$ and $(-a)b$ both satisfy the necessary property to be the additive inverse of ab .)
- Show that $(-a)(-b) = ab$. (Recall from group theory that we have $-(-a) = a$.)

1.2. Prove that if R is a ring, $a, b \in R$, and $n, m \in \mathbf{Z}$, then $(ma)(nb) = (mn)(ab)$.

1.3. Prove that if R is a ring with identity and R has at least two elements, then $0 \neq 1$ (i.e., the additive identity cannot equal the multiplicative identity).

1.4. Show that the ring Z_6 has zero divisors other than 0.

1.5. Verify that $(\mathcal{P}(S), \oplus, \cap)$ (Example (1.5.6)) is a commutative ring with identity.

† 1.6. Suppose R is a ring. A nonempty subset S of R is called a *subring* of R if S forms a ring under the operations in R .

- Show that a nonempty subset S is a subring of R if the following condition is satisfied: if $x, y \in S$, then $x - y \in S$ and $xy \in S$.
- Show that $2\mathbf{Z} = \{2n : n \in \mathbf{Z}\}$ is a subring of \mathbf{Z} . Note that $2\mathbf{Z}$ is an example of a ring without (multiplicative) identity.

1.7. a) Suppose R is an integral domain with identity element 1 and suppose S is a subring of R that contains 1. Show that S is an integral domain.

b) With notation as in (a), if R is a field, must S be a field?

† 1.8. Prove that any field is an integral domain.

† 1.9. Suppose R is a ring with identity. Prove that the set of all units in R forms a group under the multiplication in R .

- 1.10. Show that $S = \{m + n\sqrt{-1} : m, n \in \mathbf{Z}\}$ is a subring of the field of complex numbers. S is called the ring of Gaussian integers and is usually denoted $\mathbf{Z}[i]$.
- † 1.11. Denote the elements of \mathbf{Z}_5 by $\bar{0}, \bar{1}, \bar{2}, \bar{3}$, and $\bar{4}$ to distinguish them from ordinary integers.
- Find $2 \cdot \bar{3}, 3 \cdot \bar{3}, 4 \cdot \bar{3}, 5 \cdot \bar{3}$. (The integer without the bar here is a coefficient.)
 - Find $\bar{3}^2, \bar{3}^3, \bar{3}^4, \bar{3}^5$.
 - Show that \mathbf{Z}_5 is a field.
- † 1.12. A ring R is called a *Boolean ring* if $r^2 = r$ for all $r \in R$. (An element r such that $r^2 = r$ is called an *idempotent*.)
- In a Boolean ring R , show that $r + r = 0$ for all $r \in R$. (Hint: Expand $(r + r)^2$.)
 - Show that a Boolean ring must be commutative. (Hint: Expand $(r + s)^2$.)
 - Show that $\mathcal{P}(S)$ (Example (1.5.6)) is a Boolean ring.
- † 1.13. An element r of a ring R is called *nilpotent* if $r^n = 0$ for some $n \in \mathbf{P}$.
- Prove that a nilpotent element is a zero divisor.
 - Find all nilpotent elements in the ring \mathbf{Z}_8 .
- 1.14. Show that a ring with three elements must be commutative. (Hint: You should know what the addition table must look like. Use the distributive laws to see that multiplication must be commutative.)
- 1.15. How many distinct rings with identity with precisely the three elements $0, 1$, and a are there?
- 1.16. In the ring \mathbf{Z}_{12} , find all units and all zero divisors.
- 1.17. Prove that a unit in a ring with identity cannot be a zero divisor.
- 1.18. Let R be an integral domain. Let A denote the set of all ordered pairs (a, b) where $a, b \in R$ and $b \neq 0$. Define a relation \equiv on A by $(a, b) \equiv (c, d)$ if and only if $ad = bc$. (Think of the elements of R as being like the integers and the elements of A as being like fractions a/b .)
- Prove that \equiv is an equivalence relation on A .
 - Let F denote the set of equivalence classes of \equiv and let a/b denote the equivalence class of (a, b) . Define two operations on F by

$$a/b + c/d = (ad + bc)/bd \text{ and } a/b \cdot c/d = ac/bd,$$

where addition and multiplication on the right sides of the above two equations are the operations in R . Show that these two operations are well-defined.

- c) Prove that $(F, +, \cdot)$ is a field. F is called the *quotient field* or *field of fractions* of the integral domain R .

2. \mathbf{Z}_n

In this section, we consider the rings of the form $(\mathbf{Z}_n, +_n, \cdot_n)$, where $n > 1$. (The operations $+_n$ and \cdot_n were defined in Example (III.2.2).) These rings will be critical for our later work on finite fields and applications. By Examples (III.6.13.2) and (III.5.5), we can view the elements of \mathbf{Z}_n as the n distinct equivalence classes (or cosets) of the relation congruence modulo n on \mathbf{Z} . We recall that congruence modulo n , \equiv_n , is defined by $a \equiv_n b$ if and only if $n|(b - a)$ or, equivalently, if and only if $n|(a - b)$. This means that two integers lie in the same equivalence class if and only if they differ by an integral multiple of n .

We will be using here many of the fundamental facts about the integers that were presented in I.2; the reader may wish to review that section.

(2.1) NOTATION. Suppose $a \in \mathbf{Z}$. We will use two notations for the equivalence class of a with respect to \equiv_n . When we are dealing with more than one modulus at a time, we will let $[a]_n$ denote this equivalence class. (For example, we could write $[0]_4 \subset [0]_2$.) However, when we are dealing with only one modulus, which will be most of the time, we will let \bar{a} denote this equivalence class. Also, by our conventions in the previous section, we may, and will, denote the additive identity in \mathbf{Z}_n by 0 instead of $\bar{0}$ and the multiplicative identity by 1 instead of $\bar{1}$.

Thus we can denote the elements of \mathbf{Z}_n by $0, 1, \bar{2}, \dots, \overline{n-1}$. Since we can view these elements as equivalence classes, we could also denote \bar{a} by \bar{c} where c is any integer that differs from a by an integral multiple of n ; for example, $\overline{-1} = \overline{n-1}$. Additionally, we will be denoting the sum and product of two elements in \mathbf{Z}_n by $\bar{a} + \bar{b}$ and $\bar{a}\bar{b}$ instead of using $+_n$ and \cdot_n . It will be clear from the presence of the "bars" over the elements that the operation is occurring in \mathbf{Z}_n . Note that $\overline{a+b} = \bar{a} + \bar{b}$ and $\overline{ab} = \bar{a}\bar{b}$ by Example (III.5.5).

(2.2) PROPOSITION. \mathbf{Z}_n is an integral domain if and only if n is prime.

156

PROOF. First, note that the only property of an integral domain that we must worry about is the property of having no zero divisors except 0. Now,

suppose n is not prime. Then there exist two integers a and b such that $0 < a, b < n$ and $n = ab$. But then $\bar{a}\bar{b} = 0$ and $\bar{a}, \bar{b} \neq 0$. This says exactly that \bar{a} and \bar{b} are zero divisors in \mathbf{Z}_n , hence this ring is not an integral domain.

Now suppose n is prime and suppose $\bar{a}\bar{b} = 0$. Then $n|ab$ and since n is prime, it follows from Corollary (I.2.19) that $n|a$ or $n|b$. Hence either $\bar{a} = 0$ or $\bar{b} = 0$. Therefore, \mathbf{Z}_n has no zero divisors except 0 and is an integral domain. ■

(2.3) COROLLARY. \mathbf{Z}_n is a field if and only if n is prime.

PROOF. This follows from Propositions (1.7), (2.2), and Exercise 1.8. ■

(2.4) DEFINITION. The field \mathbf{Z}_p , where p is a prime, is often denoted $GF(p)$ and is called the *Galois field with p elements*. (Évariste Galois (1811–1832) was a French mathematician who was one of the first to study finite fields. He died in a duel at age 20.)

► **(2.5) EXAMPLE.** Let's find the multiplicative inverse of each nonzero element in \mathbf{Z}_7 . We have

$$\begin{aligned} 1^{-1} &= 1, \quad 2^{-1} = \bar{4}, \quad 3^{-1} = \bar{5} \\ \bar{4}^{-1} &= \bar{2}, \quad \bar{5}^{-1} = \bar{3}, \quad \bar{6}^{-1} = \bar{6}. \quad \blacktriangleleft \end{aligned}$$

Recall from I.2 that if $a, b \in \mathbf{Z}$, then (a, b) denotes the greatest common divisor of a and b and that a and b are called relatively prime if $(a, b) = 1$. The next proposition describes exactly what the units (i.e., elements with a multiplicative inverse) are in \mathbf{Z}_n and could be used to give another proof of Corollary (2.3).

(2.6) PROPOSITION. An element $\bar{a} \in \mathbf{Z}_n$ has a multiplicative inverse if and only if $(a, n) = 1$.

PROOF. Suppose $(a, n) = 1$. Then, by Proposition (I.2.14), there exist integers s and t such that $1 = sa + tn$. But then in \mathbf{Z}_n we have $1 = \bar{s}\bar{a}$ and so \bar{a} is a unit in \mathbf{Z}_n .

Conversely, suppose $\bar{a}\bar{b} = \bar{1}$. Then there exists an integer k such that $ab - 1 = kn$, or $1 = ab - kn$. But then if an integer divides a and n , it must divide 1, so $(a, n) = 1$. ■

(2.7) DEFINITION. If n is any positive integer, then $\phi(n)$ is the number of positive integers less than or equal to n that are relatively prime to n . The function ϕ is called the *Euler phi function* or *totient function*.

For example, we have $\phi(1) = 1, \phi(2) = 1, \phi(3) = 2, \phi(4) = 2, \phi(5) = 4, \phi(6) = 2$. Note that if p is prime, then $\phi(p) = p - 1$. We will give a proof for the next lemma in §4.

(2.8) LEMMA. If $n = p_1^{r_1} p_2^{r_2} \cdots p_m^{r_m}$ is the prime factorization of n , then

$$\phi(n) = (p_1^{r_1} - p_1^{r_1-1})(p_2^{r_2} - p_2^{r_2-1}) \cdots (p_m^{r_m} - p_m^{r_m-1}).$$

Proposition (2.6) says that the (multiplicative) group of units in \mathbf{Z}_n has order $\phi(n)$. Applying Corollary (III.4.10), we then have

(2.9) COROLLARY (Euler). If $(a, n) = 1$, then $\bar{a}^{\phi(n)} = 1$ (equivalently, $a^{\phi(n)} \equiv_n 1$).

► **(2.10) EXAMPLE.** We will find the last three digits of $(243)^{81}$. (Try this on your calculator — this number is too big for your calculator to handle.) Finding the last three digits of a number a is the same as finding $a \bmod 1000$ (i.e., the number b , $0 \leq b < 1000$ such that $[a]_{1000} = [b]_{1000}$). This example is constructed so that we can make good use of Corollary (2.9). First, note that the prime factorization of 1000 is $1000 = 2^3 5^3$. Hence, by Lemma (2.8), we have $\phi(1000) = (8 - 4)(125 - 25) = 400$. Now,

$$\begin{aligned} (243)^{81} &= (3^5)^{81} \\ &= 3^{405} = 3^{400} \cdot 3^5. \end{aligned}$$

But, by Corollary (2.9), $3^{400} \equiv 1 \bmod 1000$. Hence $(243)^{81} \equiv 3^5 \bmod 1000$ and so the last three digits of $(243)^{81}$ are $3^5 = 243$. Note that since $(243)^{80} \equiv 1 \bmod 1000$, the last three digits of $(243)^{80}$ are 001. ◀

(2.11) COROLLARY (Fermat). If p is a prime, then

$$\begin{aligned} a^{p-1} &\equiv 1 \pmod{p} && \text{if } p \nmid a \\ a^p &\equiv a \pmod{p} && \text{for all } a \in \mathbf{P}. \end{aligned}$$

PROOF. The first assertion follows immediately from Corollary (2.9) and the fact that $\phi(p) = p - 1$ if p is prime. As for the second assertion, if $p \nmid a$, then $\bar{a}^{p-1} = 1$ in \mathbf{Z}_p and multiplying both sides by \bar{a} yields $\bar{a}^p = \bar{a}$. If $p \mid a$, then $\bar{a} = 0$ and $\bar{a}^p = 0$. ■

This last result, often called “Fermat’s Little Theorem,” may be used to show that an integer is *not* prime.

- **(2.12) EXAMPLE.** Is 511 prime? If it were, then a^{511} would have to be congruent to $a \bmod 511$ for every $a \in \mathbf{P}$. But consider $2^{511} \bmod 511$. Note that $2^9 = 512$, so $\bar{2}^9 = 1$ in \mathbf{Z}_{511} . Now write $511 = 56 \cdot 9 + 7$. Then in \mathbf{Z}_{511} we have

$$\begin{aligned}\bar{2}^{511} &= \bar{2}^{(56 \cdot 9 + 7)} \\ &= (\bar{2}^9)^{56} \cdot \bar{2}^7 \\ &= \bar{2}^7 = \overline{128}.\end{aligned}$$

Thus 2^{511} is congruent to 128, and not to 2, mod 511. We may then conclude from Corollary (2.11) that 511 is not prime. (Of course, $511 = 7 \cdot 73$.) Note that if 2^{511} had turned out to have been congruent to 2 mod 511, then we could not have concluded anything. Corollary (2.11) can be used to show that an integer is *not* prime, but it cannot be used to prove an integer is prime. It is, in general, a very difficult problem to prove that a large prime is in fact prime and an even harder problem to obtain the prime factorization of a large integer. ◀

Note that if we add 1 to itself n times in \mathbf{Z}_n we get 0. This motivates the following definition.

(2.13) DEFINITION. A ring R with identity is said to be of *characteristic* n if n is the least positive integer such that $n1 = 0$ in R . If $m1 \neq 0$ in R for every positive integer m , then R is said to be of *characteristic* 0.

Thus, \mathbf{Z}_n is of characteristic n , while \mathbf{Z} is of characteristic 0. We leave it to the reader to show as exercises that any *finite* ring with identity must be of positive (i.e., not zero) characteristic and that any integral domain of positive characteristic must be of characteristic p for some prime p . We note that an infinite ring can still be of positive characteristic (for example, the ring of polynomials in one indeterminate with coefficients in \mathbf{Z}_2 is infinite and of characteristic 2).

To close this section, we consider some simple algebra over the finite field $GF(p)$. First, note that every linear equation may be solved over $GF(p)$ since every nonzero element has a multiplicative inverse.

- **(2.14) EXAMPLE.** Let's solve the equation $\bar{3}x + \bar{2} = 0$ in \mathbf{Z}_7 . The additive inverse of $\bar{2}$ in \mathbf{Z}_7 is $\bar{5}$ and adding $\bar{5}$ to both sides of our equation yields $\bar{3}x = \bar{5}$. Now we want to multiply both sides by the multiplicative inverse of $\bar{3}$, which is $\bar{5}$. This yields $x = \bar{5} \cdot \bar{5} = \bar{4}$. ◀

Now we consider quadratic equations. In general, not every quadratic equation will have two solutions over $GF(p)$. (Of course, this is also the case over \mathbf{Q} and over \mathbf{R} .) Again, we will assume here a basic familiarity with

polynomials. Recall that a polynomial is called irreducible over a field if it cannot be written as the product of two polynomials of smaller degree. A quadratic polynomial is irreducible over a field F if and only if it has no roots in F . (A formal proof of this is given in §6.) In the next example, we find the quadratic polynomials that are irreducible over $GF(2)$ and over $GF(3)$.

► (2.15) EXAMPLES

1) The quadratic polynomials over $GF(2)$ are just X^2 , $X^2 + X$, $X^2 + 1$, and $X^2 + X + 1$. The only one of these that is irreducible is $X^2 + X + 1$, for we have

$$\begin{aligned} X^2 &= X \cdot X \\ X^2 + X &= X \cdot (X + 1) \\ X^2 + 1 &= (X + 1) \cdot (X + 1), \end{aligned}$$

while if we put $f(X) = X^2 + X + 1$, then $f(0) = 1$ and $f(1) = 1$, so $f(X)$ has no roots in $GF(2)$.

2) Over $GF(3)$, there are 9 monic (leading coefficient 1) quadratic polynomials, since there are 3 choices for the coefficient of X and 3 choices for the constant term. There are 3 monic polynomials of degree 1 over $GF(3)$, and thus there are 3 monic quadratics that equal the square of a linear polynomial and $\binom{3}{2} = 3$ monic quadratics that are the product of two different monic linear polynomials. Therefore, there must be 3 irreducible monic quadratic polynomials over $GF(3)$. By multiplying the monic linear polynomials in pairs and eliminating the resulting quadratic from a list of all 9 monic quadratics, one sees that the irreducible monic quadratic polynomials over $GF(3)$ are $X^2 + 1$, $X^2 + X + \bar{2}$, and $X^2 + \bar{2}X + \bar{2}$. Alternatively, one could find these irreducible quadratics by testing each monic quadratic polynomial at 0, 1, and $\bar{2}$ to see which ones have no roots in $GF(3)$. ◀

EXERCISES

- 2.1. Find the multiplicative inverse of each nonzero element in \mathbf{Z}_5 and in \mathbf{Z}_{11} .
- 2.2. Determine which elements are units in \mathbf{Z}_{18} and find the multiplicative inverse of each unit.
- 2.3. Let S denote the subset $\{0, \bar{2}, \bar{4}, \bar{6}, \bar{8}\}$ of the ring \mathbf{Z}_{10} .
 - a) Show that S is a subring of \mathbf{Z}_{10} .
 - b) Show that S is actually a field. (Of course, the identity in S is not the identity in \mathbf{Z}_{10} .)

2.4. Find $\phi(16)$, $\phi(31)$, and $\phi(60)$.

† 2.5. a) Determine the last three digits of $(81)^{102}$.

b) Determine the last three digits of $(125)^{134}$. (Careful — don't try to use a result that is not applicable here.)

2.6. Verify that $2^p \equiv_p 2$ for $p = 11$ and $p = 17$.

† 2.7. a) If a_0, a_1, \dots, a_n are nonnegative integers, then show that

$$10^n a_n + 10^{n-1} a_{n-1} + \cdots + a_0 \equiv_9 a_n + a_{n-1} + \cdots + a_0.$$

b) Establish the “rule of nine”: a positive integer m is divisible by 9 if and only if the sum of the digits in the base 10 representation of m is divisible by 9. (So, for example, 738 is divisible by 9 since $7 + 3 + 8 = 18$.)

† 2.8. Show how Fermat's Theorem (2.11) can be used to show that 63 is not prime.

† 2.9. Solve each of the following equations:

a) $\bar{3}x + \bar{3} = 0$ in \mathbf{Z}_7 .

b) $\bar{2}x - 1 = 0$ in \mathbf{Z}_{11} .

c) $x^2 = \bar{2}$ in \mathbf{Z}_7 .

2.10. Show that the equation $\bar{3}x + \bar{2} = \bar{0}$ has no solutions in \mathbf{Z}_6 . (Note that this is not a contradiction to the remark before Example (2.14) since \mathbf{Z}_6 is not a field.)

† 2.11. Prove that a finite ring with identity must be of positive characteristic. (Hint: The sequence $1, 2 \cdot 1, 3 \cdot 1, \dots$ must eventually have a repetition.)

† 2.12. Suppose R , a ring with identity, is of characteristic n . Prove that $nr = 0$ for every $r \in R$.

† 2.13. Prove that an integral domain of positive characteristic must be of prime characteristic. (Hint: Note that $(mn)1 = (m1)(n1)$.)

2.14. Suppose R is a ring with identity of characteristic n . Show that if $m1 = 0$ in R , then $n|m$. (Hint: By the Division Algorithm, there exist q and r such that $m = qn + r$ with $0 \leq r < n$. Show that $r1 = 0$.)

† 2.15. Find the inverse of $\bar{35}$ in \mathbf{Z}_{43} . (Hint: Use the Euclidean Algorithm to express 1 in the form $35s + 43t$.)

- † 2.16. Construct addition and multiplication tables for a field of characteristic 2 with 4 elements. (Note that \mathbf{Z}_4 is not a field, so does not provide a solution; use Exercise 2.12 above to see that $a = -a$ for every a in a field of characteristic 2.)
- † 2.17. Find all irreducible cubic polynomials over $GF(2)$.
- 2.18. Compute the number of irreducible monic quadratic polynomials with coefficients in $GF(5)$.
- 2.19. Derive a formula for the number of irreducible monic quadratic polynomials with coefficients in $GF(p)$ where p is prime.

3. Ideals and Ring Homomorphisms

Let $(R, +, \cdot)$ be a ring and suppose I is a subgroup of $(R, +)$. Since R forms a commutative group under $+$, I is a normal subgroup of $(R, +)$ and, as in III.5, we may define a group structure on R/I , the set of equivalence classes of R with respect to the equivalence relation \equiv_I of III.4. We recall that $a \equiv_I b$ if and only if $(b - a) \in I$. We will denote the equivalence class containing a by $[a]$. It is natural to ask now whether we may put a ring structure on R/I . The answer is “no” unless I satisfies an important additional property.

- (3.1) **EXAMPLE.** \mathbf{Z} is a subgroup of \mathbf{R} ; indeed, \mathbf{Z} is even a subring of \mathbf{R} . Consider the (additive) quotient group \mathbf{R}/\mathbf{Z} . We wish to see that the operation $[a] \cdot [b] = [ab]$ is *not* well-defined on this quotient. Indeed, we have $[0] = [1](= \mathbf{Z})$ and clearly $[\sqrt{2}] = [\sqrt{2}]$, but $[0] = [0 \cdot \sqrt{2}] \neq [1 \cdot \sqrt{2}] = [\sqrt{2}]$. ◀

(3.2) **DEFINITION.** Suppose $(R, +, \cdot)$ is a ring and I is a subgroup of $(R, +)$. Then I is called an *ideal* of R if whenever $r \in R$ and $x \in I$, then $rx \in I$ and $xr \in I$; that is, if

$$R \cdot I \subseteq I \text{ and } I \cdot R \subseteq I.$$

Here, $R \cdot I$ (respectively, $I \cdot R$) denotes the set of all elements of the form rx (respectively, xr) where $r \in R$ and $x \in I$. The “closure” property in the above definition is very strong. Not only does it say that I is closed under multiplication (meaning that if $x, y \in I$, then $xy \in I$), but that the product of any element in R times any element in I must be in I . Note that if R is

commutative then, of course, we need only see that $R \cdot I \subseteq I$ to conclude that an additive subgroup I is an ideal. Also note that the ring R and the trivial subgroup $\{0\}$ are clearly ideals of R and that 0 must belong to any ideal. We remark that the subring \mathbf{Z} of \mathbf{R} is not an ideal of \mathbf{R} since $\sqrt{2} \in \mathbf{R}$ and $1 \in \mathbf{Z}$, but $\sqrt{2} \cdot 1 \notin \mathbf{Z}$.

Recall from III.5 that the operation $[a] \cdot [b] = [ab]$ will be well-defined if and only if \equiv_I is a congruence on the semigroup (R, \cdot) . The next proposition shows that the property in Definition (3.2) is exactly what is needed to put a ring structure on R/I . One should think of ideals as playing the same role in the theory of rings as normal subgroups play in the theory of groups.

(3.3) PROPOSITION. \equiv_I is a congruence on (R, \cdot) if and only if I is an ideal of R .

PROOF. First, suppose \equiv_I is a congruence on (R, \cdot) and suppose $r \in R$ and $x \in I$. Then $x \equiv_I 0$ and $r \equiv_I r$, so, since \equiv_I is a congruence, we have $rx \equiv_I r0 = 0$. This says that $rx \in I$. Similarly, we have $0 = r0 \equiv_I rx$, so $rx \in I$.

Conversely, suppose I is an ideal of R . Suppose $a \equiv_I b$ and $c \equiv_I d$, where $a, b, c, d \in R$. We need to show that $ac \equiv_I bd$, which is the same as showing $bd - ac \in I$. Now,

$$\begin{aligned} bd - ac &= bd - bc + bc - ac \\ &= b(d - c) + (b - a)c. \end{aligned}$$

But $(d - c) \in I$ and $(b - a) \in I$ and I is an ideal, so $b(d - c) \in I$ and $(b - a)c \in I$. Since I is a subgroup, the difference of two elements in I is again in I , so $bd - ac \in I$ and we're done. ■

(3.4) COROLLARY. If I is an ideal of R , then R/I is a ring under the operations $[a] + [b] = [a + b]$ and $[a] \cdot [b] = [ab]$.

PROOF. The operation $[a] \cdot [b] = [ab]$ is well-defined by Proposition (3.3). It is now easy to check that R/I inherits from R all the necessary algebraic properties to be ring. ■

If I is an ideal of R , then the ring R/I (read “ R mod I ”) is called the *quotient ring*, or *factor ring*, of R by I .

The next proposition gives a description of all ideals of \mathbf{Z} .

(3.5) PROPOSITION. Every ideal of \mathbf{Z} is of the form $(m) = \{km : k \in \mathbf{Z}\}$ for some $m \in \mathbf{Z}$. Conversely, for every integer m the cyclic subgroup (m) is an ideal of \mathbf{Z} .

PROOF. Suppose I is an ideal of \mathbf{Z} . Then in particular I is a subgroup of the additive group $(\mathbf{Z}, +)$, which is a cyclic group. Hence, by Proposition (III.3.19), I must be cyclic, so of the form $(m) = \{km : k \in \mathbf{Z}\}$ for some m . We note that an inspection of the proof of (III.3.19) shows that if I is nonzero, then I is generated by the least positive integer in I .

Conversely, if $n \in \mathbf{Z}$ and if $km \in (m)$, then $n(km) = (km)n = (nk)m \in (m)$. Hence the cyclic subgroup (m) is an ideal of \mathbf{Z} . ■

(3.6) DEFINITION. Suppose R is a commutative ring with identity and $a \in R$. It is not hard to see that the subset

$$(a) = \{ra : r \in R\}$$

is an ideal of R . It is called the *principal ideal generated by a* .

In a ring other than \mathbf{Z} , there is a potential for confusion since our notations for cyclic subgroup and principal ideal are the same. But it will be clear from the context whether we are considering the cyclic subgroup $\{na : n \in \mathbf{Z}\}$ or the principal ideal $\{ra : r \in R\}$.

(3.7) DEFINITION. If R is an integral domain in which every ideal is principal, then R is called a *principal ideal domain*.

Proposition (3.5) then says that \mathbf{Z} is a principal ideal domain. Principal ideals will be the type of ideals in which we will be predominantly interested, so the reader may concentrate his or her efforts on understanding them and not be overly concerned with more abstract ideal theory.

Note that the quotient ring $\mathbf{Z}/(m)$ is exactly the ring \mathbf{Z}_m of the previous section.

The most useful mappings between two rings are naturally those that respect the algebraic structures of the rings. This means that the map should be a homomorphism of the additive groups and of the multiplicative semi-groups of the rings. Accordingly, we make the following definition.

(3.8) DEFINITIONS. Suppose $(R, +, \cdot)$ and $(\tilde{R}, \tilde{+}, \tilde{\cdot})$ are rings. A mapping $\varphi : R \rightarrow \tilde{R}$ is called a *ring homomorphism* if for all $a, b \in R$ the following two properties hold:

- 1) $\varphi(a + b) = \varphi(a) \tilde{+} \varphi(b)$.
- 2) $\varphi(a \cdot b) = \varphi(a) \tilde{\cdot} \varphi(b)$.

A ring homomorphism is called a *ring isomorphism* if it is a 1-1 and onto mapping. The rings R and \tilde{R} are said to be *isomorphic* if there exists a ring isomorphism from R to \tilde{R} .

► **(3.9) EXAMPLE.** Suppose R is a ring and I is an ideal of R . Then, as in the theory of groups, we have a natural ring homomorphism:

$$\begin{aligned}\pi_I : R &\longrightarrow R/I \\ a &\longmapsto [a].\end{aligned}$$

The fact that π_I is a ring homomorphism follows immediately from the definition of the operations in R/I . This is the main example of a ring homomorphism in which we will be interested.

For a more specific example, given an integer $m > 1$, we have the ring homomorphism $\pi_m : \mathbf{Z} \rightarrow \mathbf{Z}/(m)$ that takes an integer a to $[a]_m$, its equivalence class mod m . ◀

The kernel of a ring homomorphism is its kernel as a homomorphism of additive groups (Definition (III.6.8)); i.e., if $\varphi : R \rightarrow \tilde{R}$ is a ring homomorphism, then $\text{Ker}(\varphi) = \{r \in R : \varphi(r) = 0_{\tilde{R}}\}$. Note that the kernel of the ring homomorphism π_m above is the (principal) ideal (m) of \mathbf{Z} .

Recall from Proposition (III.6.10) that φ is a 1-1 mapping if and only if $\text{Ker}(\varphi) = \{0_R\}$. We also know from Proposition (III.6.9) that $\text{Ker}(\varphi)$ is a normal subgroup of $(R, +)$; but even more is true.

(3.10) PROPOSITION. Suppose $\varphi : R \rightarrow \tilde{R}$ is a ring homomorphism. Then $\text{Ker}(\varphi)$ is an ideal of R .

PROOF. We already know that $\text{Ker}(\varphi)$ is a subgroup of $(R, +)$. Now, suppose $a \in R$ and $x \in \text{Ker}(\varphi)$. Then

$$\varphi(ax) = \varphi(a)\varphi(x) = \varphi(a) \cdot 0_{\tilde{R}} = 0_{\tilde{R}},$$

and so $ax \in \text{Ker}(\varphi)$. Similarly, one can show that $xa \in \text{Ker}(\varphi)$, so $\text{Ker}(\varphi)$ is an ideal of R . ■

(3.11) THEOREM. Suppose $\varphi : R \rightarrow \tilde{R}$ is a ring homomorphism. Then the map

$$\bar{\varphi} : R/\text{Ker}(\varphi) \longrightarrow \tilde{R}$$

defined by $\bar{\varphi}([a]) = \varphi(a)$ is a 1-1 ring homomorphism.

PROOF. All that needs to be shown is that the map $\bar{\varphi}$ of Theorem (III.6.11) is a well-defined homomorphism from the semigroup $(R/\text{Ker}(\varphi), \cdot)$ to the semigroup (\tilde{R}, \cdot) , and this we leave to the reader. ■

(3.12) COROLLARY. (First Isomorphism Theorem for Rings) If $\varphi : R \rightarrow \tilde{R}$ is a surjective ring homomorphism, then $\bar{\varphi} : R/\text{Ker}(\varphi) \rightarrow \tilde{R}$ is a ring isomorphism.

PROOF. If φ is surjective, then the map $\bar{\varphi}$ is surjective. Theorem (3.11) shows that $\bar{\varphi}$ is also a 1-1 ring homomorphism. ■

As an example, if we consider the mapping $\varphi : \mathbf{Z} \rightarrow \mathbf{Z}_m$ defined by $\varphi(a) =$ the remainder when a is divided by m , then it is easy to see that φ is a surjective ring homomorphism with kernel (m) . This shows yet again that we may identify the rings $\mathbf{Z}/(m)$ and \mathbf{Z}_m .

EXERCISES

- 3.1. Suppose I is an ideal of a ring R .
- Show that if R is commutative, then R/I is commutative.
 - Show that if R has an identity element, then R/I has an identity element.
- † 3.2. Let R be a ring with identity and suppose I is an ideal of R .
- Show that if $1 \in I$, then $I = R$.
 - Show that if I contains a unit (an element with a multiplicative inverse), then $I = R$.
 - Show that the only ideals in a field F are (0) and F .
- † 3.3. a) Show that the intersection of two ideals is an ideal.
b) Give an example to show that the union of two ideals need not be an ideal.
- 3.4. Suppose I and J are two ideals in a ring R . Let
- $$I + J = \{x + y : x \in I \text{ and } y \in J\}.$$
- Show that $I + J$ is an ideal of R and that it is the smallest ideal of R that contains both I and J .
- 3.5. An ideal I of a commutative ring R is called *prime* if $I \neq R$ and whenever $a, b \in R$ and $ab \in I$, then $a \in I$ or $b \in I$.
- Show that the ideal (n) of \mathbf{Z} is prime if and only if n is prime.
 - Suppose R is a commutative ring with identity. Show that I is a prime ideal if and only if R/I is an integral domain.
 - Suppose R is a commutative ring. Suppose $\varphi : R \rightarrow S$ is a ring homomorphism and that S has no zero divisors. Show that $\text{Ker}(\varphi)$ is a prime ideal of R .
- 3.6. Suppose R and S are rings with identity. Let $\varphi : R \rightarrow S$ be a surjective ring homomorphism. Show that $\varphi(1_R) = 1_S$.

- 3.7.** Suppose R and S are rings with identity. Let $\varphi : R \rightarrow S$ be a ring homomorphism such that $\varphi(r) \neq 0$ for some $r \in R$. Show that if S is an integral domain, then $\varphi(1_R) = 1_S$.
- 3.8.** Suppose R and S are rings with identity and $\varphi : R \rightarrow S$ is a ring homomorphism such that $\varphi(1_R) = 1_S$. Show that the image of a unit in R is a unit in S .
- 3.9. a)** Suppose $\varphi : R \rightarrow S$ is a ring homomorphism. Show that if J is an ideal of S , then

$$\varphi^{-1}(J) = \{r \in R : \varphi(r) \in J\}$$

is an ideal of R that contains $\text{Ker}(\varphi)$.

- b)** Suppose R and S are commutative rings. Show that if J is a prime ideal of S , then $\varphi^{-1}(J)$ is a prime ideal of R .

- 3.10.** Suppose $\varphi : R \rightarrow S$ is a surjective ring homomorphism. Show that if I is an ideal of R , then $\varphi(I) = \{\varphi(x) : x \in I\}$ is an ideal of S . (Note that $\varphi(I)$ is a subgroup of S by Corollary (III.6.7).)
- 3.11.** Suppose $\varphi : R \rightarrow S$ is a surjective ring homomorphism. Use the previous two exercises to show that there is a 1-1 correspondence between the ideals of S and the ideals of R that contain $\text{Ker}(\varphi)$. In particular, show that there is a 1-1 correspondence between ideals of R/I and ideals of R containing I .
- † **3.12. a)** Show that if I is an ideal of \mathbf{Z}_n , then $I = (\bar{a})$ for some $\bar{a} \in \mathbf{Z}_n$.
b) Show that the distinct nonzero ideals of \mathbf{Z}_n are the ideals of the form (\bar{a}) , where a is a positive divisor of n .
c) Find all ideals in \mathbf{Z}_{24} .
- 3.13.** Find all ring homomorphisms from \mathbf{R} to \mathbf{Z} . (Hint: Use Proposition (3.10) and Exercise 3.2.(c) above.)
- † **3.14. a)** Explicitly give a ring homomorphism from \mathbf{Z}_4 to \mathbf{Z}_2 that takes $[1]_4$ to $[1]_2$.
b) Can you give a ring homomorphism from \mathbf{Z}_2 to \mathbf{Z}_4 that takes $[1]_2$ to $[1]_4$?
- † **3.15.** Show that $\varphi(a) = 3a$ is a group homomorphism from $(\mathbf{Z}, +)$ to itself but is not a ring homomorphism from $(\mathbf{Z}, +, \cdot)$ to itself.
- 3.16.** An ideal I of a ring R is called a *proper* ideal if $I \neq R$ and $I \neq (0)$. An ideal M is called a *maximal* ideal of R if $M \neq R$ and if M is not contained in any other proper ideal of R .

- a) Show that the maximal ideals of \mathbf{Z} are exactly the ideals (p) , where p is prime. (Hint: If $a \in \mathbf{Z}$ and $a \notin (p)$, then a and p are relatively prime.)
- b) Show that a maximal ideal of a commutative ring R with identity is a prime ideal (see Exercise 3.5) of R . (Hint: If $a \notin M$, consider $M + (a)$.)
- c) Suppose R is a commutative ring with identity. Show that M is a maximal ideal of R if and only if R/M is a field.

3.17. Let R be an integral domain and let F denote the quotient field of R , which was defined in Exercise 1.18. Show that the map $\varphi : R \rightarrow F$ defined by $\varphi(a) = a/1$ is a 1-1 ring homomorphism.

4. The Chinese Remainder Theorem

We begin this section with a “trick.” Think of a positive integer less than 100. Divide your integer by 4 and call the remainder a . Divide your integer by 25 and call the remainder b . Then your integer is the last two digits of $25a + 76b$.

Of course, this is not really magic. Rather, it is a simple application of the Chinese Remainder Theorem. The elements of this theorem appeared in the work of the Chinese mathematician Sun Tzu and, while the date of his writing is very uncertain, he lived sometime before A.D. 500. A modern and general form of the theorem was given by Euler in 1734. An interesting account of the history of this theorem is presented in Davis and Hersh [8].

We will present two forms of the theorem. The first is rather theoretical, although we will use it to give a proof of the formula for $\phi(n)$ that was stated in Lemma (2.8). The second version of the theorem is much more concrete. We will also briefly discuss an application of the theorem to computer arithmetic.

(4.1) THEOREM (Chinese Remainder Theorem). Suppose $m_1, m_2, \dots, m_k \in \mathbf{P}$ with $(m_i, m_j) = 1$ if $i \neq j$ (we call the m_i 's pairwise relatively prime). Put $m = m_1 m_2 \cdots m_k$. Then we have an isomorphism of rings

$$\mathbf{Z}_m \cong \mathbf{Z}_{m_1} \times \mathbf{Z}_{m_2} \times \cdots \times \mathbf{Z}_{m_k}.$$

PROOF. First, we must explain how to put a ring structure on a Cartesian product of rings. The obvious thing to do is to define addition and multiplication coordinatewise. In our specific example, if $a = ([a_1]_{m_1}, \dots, [a_k]_{m_k})$

and $b = ([b_1]_{m_1}, \dots, [b_k]_{m_k})$ are two elements of $\mathbf{Z}_{m_1} \times \dots \times \mathbf{Z}_{m_k}$, then we define

$$\begin{aligned} a + b &= ([a_1 + b_1]_{m_1}, \dots, [a_k + b_k]_{m_k}) \\ ab &= ([a_1 b_1]_{m_1}, \dots, [a_k b_k]_{m_k}). \end{aligned}$$

We leave it to the reader to see that $\mathbf{Z}_{m_1} \times \dots \times \mathbf{Z}_{m_k}$ is a ring under these operations.

Now we define a map

$$\psi : \mathbf{Z} \longrightarrow \mathbf{Z}_{m_1} \times \dots \times \mathbf{Z}_{m_k}$$

by

$$\psi(x) = ([x]_{m_1}, \dots, [x]_{m_k}).$$

It is easy to see that ψ is a ring homomorphism because of the way we defined addition and multiplication on $\mathbf{Z}_{m_1} \times \dots \times \mathbf{Z}_{m_k}$. What is the kernel of ψ ? We have that $\psi(x) = ([0]_{m_1}, \dots, [0]_{m_k})$ if and only if $m_i | x$ for $i = 1, 2, \dots, k$. But since the m_i 's are pairwise relatively prime, it follows from Exercise I.2.17 that all the m_i 's divide x if and only if $m = m_1 \cdots m_k$ divides x . Therefore, the kernel of ψ is the ideal (m) of \mathbf{Z} . Then, as in Theorem (3.11), ψ induces a 1-1 homomorphism

$$\bar{\psi} : \mathbf{Z}/(m) = \mathbf{Z}_m \longrightarrow \mathbf{Z}_{m_1} \times \dots \times \mathbf{Z}_{m_k}.$$

But since the domain and target rings both have exactly m elements, $\bar{\psi}$ must be onto as well as 1-1 and is thus an isomorphism. ■

(4.2) APPLICATION. As an application, we prove Lemma (2.8). Suppose

$$n = p_1^{r_1} p_2^{r_2} \cdots p_m^{r_m}$$

is the prime factorization of n . Since $(p_i^{r_i}, p_j^{r_j}) = 1$ if $i \neq j$, we may apply the Chinese Remainder Theorem to conclude that

$$\begin{aligned} \bar{\psi} : \mathbf{Z}_n &\longrightarrow \mathbf{Z}_{p_1^{r_1}} \times \dots \times \mathbf{Z}_{p_m^{r_m}} \\ [a]_n &\longmapsto ([a]_{p_1^{r_1}}, \dots, [a]_{p_m^{r_m}}) \end{aligned}$$

is an isomorphism. Recall that $\phi(n)$ is the number of units in the ring \mathbf{Z}_n . We will now use three facts that we leave as exercises: (1) if p is a prime, then $\phi(p^r) = p^r - p^{r-1}$, (2) $b = (b_1, \dots, b_m) \in R_1 \times \dots \times R_m$ is a unit if and only if b_i is a unit in R_i for $i = 1, 2, \dots, m$, and (3) if $\psi : R \rightarrow S$ is a ring

isomorphism, then $r \in R$ is a unit in R if and only if $\psi(r)$ is a unit in S . The first two of these facts say that the number of units in

$$\mathbb{Z}_{p_1^{r_1}} \times \cdots \times \mathbb{Z}_{p_m^{r_m}}$$

is

$$(p_1^{r_1} - p_1^{r_1-1})(p_2^{r_2} - p_2^{r_2-1}) \cdots (p_m^{r_m} - p_m^{r_m-1})$$

and the third fact implies that this must equal $\phi(n)$, thus establishing Lemma (2.8).

Now for the much more concrete version and proof of the theorem.

(4.3) THEOREM (Chinese Remainder Theorem). Suppose $m_1, m_2, \dots, m_k \in \mathbb{P}$ with $(m_i, m_j) = 1$ if $i \neq j$. Put $m = m_1 m_2 \cdots m_k$. Then given any integers a_1, a_2, \dots, a_k the system of congruences

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$

$$\vdots$$

$$x \equiv a_k \pmod{m_k}$$

has (integer) solutions and any two solutions are congruent mod m .

PROOF. For $i = 1, 2, \dots, k$, put

$$c_i = \left(\frac{m}{m_i} \right)^{\phi(m_i)},$$

where ϕ is the Euler phi function. Then since $(m_i, m_j) = 1$ if $i \neq j$, it follows from Proposition (I.2.18) that m/m_i is a multiple of m_j and so $c_i \equiv 0 \pmod{m_j}$ for $i \neq j$. Also, it follows from Corollary (2.9) (Euler's Theorem), and the fact that m_i is relatively prime to m/m_i , that $c_i \equiv 1 \pmod{m_i}$. Now let x_0 be the remainder when $c_1 a_1 + c_2 a_2 + \cdots + c_k a_k$ is divided by m . Then it is not hard to see that x_0 satisfies the given system of congruences. For example, using the fact that $c_1 \equiv 1 \pmod{m_1}$ and $c_i \equiv 0 \pmod{m_1}$, for $i = 2, 3, \dots, k$, one sees that $c_1 a_1 + c_2 a_2 + \cdots + c_k a_k$ is congruent to $a_1 \pmod{m_1}$.

If another integer x_1 satisfied this same system of congruences, then we would have $m_i | (x_0 - x_1)$ for $i = 1, 2, \dots, k$. It then follows from Exercise I.2.17 that $m | (x_0 - x_1)$, so that x_0 and x_1 must be congruent mod m . ■

We note that in finding the c_i 's and the expression $c_1 a_1 + \cdots + c_k a_k$ it is always good enough to work "modulo m ." That is, whenever we reach

an integer larger than m , we can replace it by its remainder upon division by m .

Theorems (4.1) and (4.3) relate to each other in the following way. Solving the system of congruences in Theorem (4.3) is the same as finding a preimage of the m -tuple $([a_1]_{m_1}, \dots, [a_k]_{m_k})$ under the homomorphism ψ in the proof of Theorem (4.1) (i.e., an integer x such that

$$\psi(x) = ([a_1]_{m_1}, \dots, [a_k]_{m_k}).$$

The fact that such a solution always exists says that ψ is a surjective mapping. The fact that any two solutions to this system of congruences must be congruent mod m says exactly that the kernel of ψ is (m) .

► (4.4) EXAMPLES

1) Returning to our “trick” at the beginning of this section, we can let $m_1 = 4$, $m_2 = 25$, and $m = 100$. For c_1 we can take $(100/4)^2 = 225$, but, reducing mod m , we take $c_1 = 25$. For c_2 we could take $(100/25)^{20} = 4^{20} \pmod{100}$, but let’s stop and think for a moment. We want c_2 to be an integer that is congruent to 0 mod 4 and to 1 mod 25. So we try 26, 51, and then we see that 76 works. If x is the “unknown” positive integer less than 100 and if $x \equiv a \pmod{4}$ and $x \equiv b \pmod{25}$, then x is the remainder when $25a + 76b$ is divided by 100 (i.e., the last two digits of $25a + 76b$).

2) We find the two smallest positive integers having remainder 1, 2, and 3 when divided by 2, 3, and 5, respectively. For c_1 , we want a multiple of 3 and 5 that is congruent to 1 mod 2; so we may take $c_1 = 15$. For c_2 , we want a multiple of 2 and 5 that is congruent to 1 mod 3; so we may take $c_2 = 10$. For c_3 , we want an integer that is a multiple of 2 and 3 and that is congruent to 1 mod 5; so we may take $c_3 = 6$. The integer between 0 and $30 = 2 \cdot 3 \cdot 5$ that satisfies the given conditions is the remainder when

$$15 \cdot 1 + 10 \cdot 2 + 6 \cdot 3 = 53$$

is divided by 30. Thus 23 is the smallest positive integer satisfying the given congruences and the next smallest integer is $23+30=53$.

3) We find the smallest positive integer that has remainders 1, 2, 3, and 4 when divided by 2, 3, 4, and 5, respectively. This problem is different from the previous one, and in fact may not even have a solution, since our “moduli” 2, 3, 4, and 5 are not pairwise relatively prime. We will approach this by first solving the set of congruences

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ x &\equiv 3 \pmod{4} \\ x &\equiv 4 \pmod{5}, \end{aligned}$$

which do involve relatively prime moduli. Here, we may take $c_1 = 40$, $c_2 = 45$, and $c_3 = 36$, and we find that the solution to the set of three congruences above is

$$x \equiv 40 \cdot 2 + 45 \cdot 3 + 36 \cdot 4 \equiv 59 \pmod{60}.$$

Now we look at the remaining congruence, $x \equiv 1 \pmod{2}$, which just says that x is odd. Therefore, the solution to our problem is 59. But note that if the remaining congruence had been $x \equiv 0 \pmod{2}$, then there would have been no solution to the system of four congruences. ◀

The Chinese Remainder Theorem has applications to computer arithmetic. The idea is that one can perform arithmetic on very large numbers by reducing to (more) computations modulo comparatively smaller numbers. This has obvious advantages on machines that are "highly parallel" and can also cut down on the number of multiple-precision operations and carries that must be performed, increasing accuracy and speed. A good reference for a discussion of these applications is D.E. Knuth [18].

We will work through an example to illustrate how to perform arithmetic with respect to several different moduli.

► **(4.5) EXAMPLE.** Suppose we wish to perform the following calculation:

$$N = 13 \cdot 19 + 22 \cdot 16 + 36 \cdot 25 + 18 \cdot 38$$

and assume that we know beforehand that the answer is less than 4620. (In practice, one would have a much bigger number than 4620 here.) The prime factorization of 4620 is $2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$ and the Chinese Remainder Theorem tells us that the map

$$\begin{aligned} \bar{\psi}: \mathbf{Z}_{4620} &\longrightarrow \mathbf{Z}_4 \times \mathbf{Z}_3 \times \mathbf{Z}_5 \times \mathbf{Z}_7 \times \mathbf{Z}_{11} \\ [a]_{4620} &\longmapsto ([a]_4, [a]_3, [a]_5, [a]_7, [a]_{11}) \end{aligned}$$

is an isomorphism. This allows us to do most of our calculations using single-digit numbers. First we find the image of each of our numbers under this isomorphism. We have

$$\begin{aligned} \bar{\psi}(13) &= ([1]_4, [1]_3, [3]_5, [6]_7, [2]_{11}) \\ \bar{\psi}(19) &= ([3]_4, [1]_3, [4]_5, [5]_7, [8]_{11}) \\ \bar{\psi}(22) &= ([2]_4, [1]_3, [2]_5, [1]_7, [0]_{11}) \\ \bar{\psi}(16) &= ([0]_4, [1]_3, [1]_5, [2]_7, [5]_{11}) \\ \bar{\psi}(36) &= ([0]_4, [0]_3, [1]_5, [1]_7, [3]_{11}) \\ \bar{\psi}(25) &= ([1]_4, [1]_3, [0]_5, [4]_7, [3]_{11}) \\ \bar{\psi}(18) &= ([2]_4, [0]_3, [3]_5, [4]_7, [7]_{11}) \\ \bar{\psi}(38) &= ([2]_4, [2]_3, [3]_5, [3]_7, [5]_{11}). \end{aligned}$$

We now find $\bar{\psi}(N)$ by performing coordinatewise addition and multiplication. We have

$$\begin{aligned}\bar{\psi}(N) &= \bar{\psi}(13)\bar{\psi}(19) + \bar{\psi}(22)\bar{\psi}(16) + \bar{\psi}(36)\bar{\psi}(25) + \bar{\psi}(18)\bar{\psi}(38) \\ &= ([3]_4, [1]_3, [2]_5, [2]_7, [5]_{11}) + ([0]_4, [1]_3, [2]_5, [2]_7, [0]_{11}) \\ &\quad + ([0]_4, [0]_3, [0]_5, [4]_7, [9]_{11}) + ([0]_4, [0]_3, [4]_5, [5]_7, [2]_{11}) \\ &= ([3]_4, [2]_3, [3]_5, [6]_7, [5]_{11}).\end{aligned}$$

Now we must find $\bar{\psi}^{-1}$ of this 5-tuple. In the notation of Theorem (4.3), if we put $m_1 = 4, m_2 = 3, m_3 = 5, m_4 = 7$, and $m_5 = 11$, then we leave it to the reader to see that $c_1 = 3465, c_2 = 1540, c_3 = 3696, c_4 = 2640$, and $c_5 = 2520$. Then to find N we compute

$$3465 \cdot 3 + 1540 \cdot 2 + 3696 \cdot 3 + 2640 \cdot 6 + 2520 \cdot 5$$

mod 4620 and find that $N = 2183$. Of course, one must imagine a similar example with a number much larger than 4620 and with many, many more computations than were involved here to see how this might be advantageous in performing arithmetic on a computer. ◀

For a discussion of the use of the Chinese Remainder Theorem in finding exact solutions to systems of linear equations, we refer the reader to Mackiw [22].

EXERCISES

- † 4.1. Show that if p is prime and $r \in \mathbf{P}$, then $\phi(p^r) = p^r - p^{r-1}$. (Hint: Count the positive integers less than or equal to p^r that are *not* relatively prime to p^r .)
- † 4.2. Suppose R_1, R_2, \dots, R_k are rings with identity and

$$R = R_1 \times R_2 \times \cdots \times R_k$$

is the product ring with coordinatewise addition and multiplication. Show that $b = (b_1, b_2, \dots, b_k)$ is a unit in R if and only if each b_i is a unit in R_i for $i = 1, 2, \dots, k$.

- † 4.3. Suppose $\psi: R \rightarrow S$ is an isomorphism of rings with identity. Show that $r \in R$ is a unit in R if and only if $\psi(r)$ is a unit in S .
- † 4.4. Find the two smallest positive integers that have remainders of 1, 1, and 3 when divided by 2, 3, and 5, respectively.

† 4.5. Find the two smallest positive integers that have remainders of 2, 3, and 4 when divided by 3, 5, and 7, respectively.

4.6. Solve the following system of congruences:

$$x \equiv 1 \pmod{3}$$

$$x \equiv 3 \pmod{4}$$

$$x \equiv 2 \pmod{7}.$$

† 4.7. If such an integer exists, find the smallest positive integer that has remainders of 2, 3, and 4 when divided by 3, 4, and 6, respectively.

† 4.8. If such an integer exists, find the smallest positive integer that has remainders of 2, 3, and 5 when divided by 3, 4, and 6, respectively.

4.9. Solve the following system of congruences:

$$x \equiv 1 \pmod{2}$$

$$x \equiv 2 \pmod{3}$$

$$x \equiv 4 \pmod{5}$$

$$x \equiv 5 \pmod{6}.$$

† 4.10. Mimic Example (4.5) to show how you could use the Chinese Remainder Theorem to compute $18 \cdot 48 + 54 \cdot 22 + 32 \cdot 60$, given that the answer is less than 4620.

5. Public-Key Encryption

The British mathematician G. H. Hardy (1877–1947) believed that no results in number theory would have practical applications. Recently, however, such results have been applied in cryptography, the study of secret messages. In this section, we will describe what is known as a public-key cryptosystem.

Suppose there is a group of people who want to send secret messages to each other. For example, the head office of a company may wish to communicate with many branch offices and may want to do so secretly so that no competitors can learn about their new products or ideas. Or the Pentagon may wish to communicate with its fleet of submarines and would obviously need to maintain privacy. One way to do this would be to use couriers to distribute to each of the members of the group a “key” that sets

up a correspondence between meaningful messages and nonsense messages. For example, the command to “sail to the north pole” might correspond via the key to the nonsense message “Mary had a little lamb.” The real message is known as the *plaintext* and the encrypted (“coded”) message is known as the *ciphertext*. There is probably no way that an eavesdropper could figure out the plaintext from the ciphertext “Mary had a little lamb.” However, once the message was sent, if an eavesdropper intercepted it and then noted that the submarine that received the message sailed to the north pole, the key would be “broken” and that message could not securely be transmitted again. This type of key is like a “one-time pad,” in which letters are encrypted in a purely random manner. There is also the danger that if the same key is distributed to many users, then if one key is stolen the entire network will be insecure. If a separate key must be distributed to every user, then complications and confusion might follow.

A public-key cryptosystem is quite different from a one-time pad. In this system, each user *publicly* tells how a message to be sent to them should be encrypted. Of course, each user does not tell how they will decrypt the ciphertext. At first, this may sound absurd — if one tells how to encrypt a message, then can’t the process just be reversed to decrypt the ciphertext? The point here is that one starts with an encrypting function that is fairly easy to compute, but whose inverse is extremely difficult to compute. Such a function is called a “one-way function.” If the inverse of the function can be fairly easily computed once one knows some extra information, then the function is called a “trapdoor one-way function.”

- (5.1) **EXAMPLE.** This example is not really a one-way function, but it gives an idea of where we are headed. It is simple to compute the value of a given polynomial function $f(x)$ at a given number r . But consider the “opposite” problem of solving the equation $f(x) = s$, where s is some number. This is much harder, although good algorithms exist (e.g., Newton’s method) for finding approximate solutions. ◀

The idea for a public-key cryptosystem was developed in the mid-1970’s by electrical engineers at Stanford and at M.I.T. We will present the RSA public-key cryptosystem, which was presented in a paper by Rivest, Shamir, and Adelman published in 1978 [28]. The trapdoor one-way function to be used here rests on two assumptions:

1. There exist efficient algorithms for determining very large (say about 100 digits) prime numbers. The interested reader may consult the *Scientific American* article by Pomerance [27]. (We note that the method in Example (2.12) for showing that an integer is not prime is employed in some of these algorithms.)

2. There do *not* exist efficient algorithms for finding the prime factorization of a very large (say about 200 digits) integer, even if the integer is the product of just two distinct primes. One may be able to say that an integer is not prime without being able to produce an actual factor. We note that it has not been proven that efficient factoring algorithms cannot exist. Perhaps some new ideas will give rise to better algorithms. Indeed, H. W. Lenstra [20] has recently achieved improvements in factoring large numbers by using the theory of elliptic curves over finite fields.

Before describing the RSA encryption procedure, we need to assume that our messages are integers. We can do this in a simpleminded way, say by letting $A = 01, B = 02, \dots, Z = 26$, and a space correspond to 00. For example, the word MATH would now correspond to the integer 13012008. One could use a more sophisticated method such as ASCII code. While the correspondence we have set up between letters and integers is a method of encrypting, it is certainly primitive and could easily be broken. It is similar to the "cryptoquotes" that appear in many newspapers. (It is generally easy to see which number corresponds to "E", since "E" is the most common letter in English, and "A" and "I" are also usually easy to identify.)

Now for the description of the RSA encrypting and decrypting procedures. Each user of the network uses prime-finding algorithms to find two distinct large (say about 100 digits) primes p and q . Then put $n = pq$. Let t denote the least common multiple of $p-1$ and $q-1$. (Alternatively, one could take $t = (p-1)(q-1) = \phi(n)$ here and everything below would still work. Indeed, this was what was done in [28].) Each user then selects a positive integer $e, 1 < e < t$, that is relatively prime to his t .

(5.2) RSA ENCRYPTION. The user publicly announces his integers n and e . If someone wishes to send a message (which we are now assuming is an integer) to this user, he first breaks up the message into blocks of integers such that each block is less than n . Given such a block M , he encrypts this block as

$$M^e \bmod n$$

and this (i.e., the remainder when M^e is divided by n) is what is sent to the user.

- **(5.3) EXAMPLE.** Of course, in the examples we will present in this section to illustrate these ideas, we will not use a 200-digit integer for n . I announce my integers to be $n = 2911$ and $e = 11$. (The reader can obtain the prime factorization of n without too much difficulty; ~~we~~ will give it below.) Suppose someone wants to send me the message "NO." This word is first converted into the integer $M = 1415$ by our simple system above.

The sender then computes $(1415)^{11} \bmod 2911$. This may seem like a very difficult computation itself, but it is not too bad if we proceed as follows. We compute $(1415)^2 \bmod 2911$, then we square this to obtain $(1415)^4 \bmod 2911$, then square again to obtain $(1415)^8 \bmod 2911$. Now we use the fact that $11=8+2+1$ and so

$$(1415)^{11} = (1415)^{8+2+1} \equiv (1415)^8 (1415)^2 (1415) \bmod 2911.$$

In detail, we obtain

$$1415^2 = 2002225 \equiv 2368 \bmod 2911$$

$$1415^4 \equiv 2368^2 \equiv 5607424 \equiv 838 \bmod 2911$$

$$1415^8 \equiv 838^2 \equiv 702244 \equiv 693 \bmod 2911.$$

Hence we have

$$\begin{aligned} (1415)^{11} &\equiv (693)(2368)(1415) \bmod 2911 \\ &\equiv (2131)(1415) \bmod 2911 \\ &\equiv 2480 \bmod 2911. \end{aligned}$$

Thus the message "NO" is encrypted as 2480 and that is what is sent to me. This took a little time with a calculator, but of course a computer would do it very quickly. ◀

Now for the decryption. Since e was chosen to be relatively prime to $p-1$ and to $q-1$, it is relatively prime to t , their least common multiple. Thus, by Proposition (2.6), \bar{e} is a unit in \mathbf{Z}_t . The key to decryption is to find the multiplicative inverse of \bar{e} in this ring.

(5.4) THEOREM (RSA Decryption). Find the integer d between 0 and t that satisfies

$$ed \equiv 1 \bmod t.$$

Then the original message M is found by computing

$$(M^e)^d \bmod n.$$

PROOF. We must show that $M \equiv M^{ed} \bmod n$. Note that once we know this, then M can be recovered as the remainder upon dividing M^{ed} by n , using the fact that M is less than n .

Since t is a common multiple of $p-1$ and $q-1$, there exist integers u and v such that $t = u(p-1)$ and $t = v(q-1)$. Since $ed \equiv 1 \bmod t$, there

exists an integer w such that $ed = 1 + wt$. Suppose $(M, p) = 1$. Then we have

$$\begin{aligned} M^{ed} &= M^{1+wt} = M \cdot M^{wt} \\ &= M \cdot (M^{p-1})^{uw} \equiv M \pmod{p} \end{aligned}$$

since $M^{p-1} \equiv 1 \pmod{p}$ by Corollary (2.11). On the other hand, if $p|M$, then $M^{ed} \equiv M \pmod{p}$ since both sides are zero. Thus we always have $M^{ed} \equiv M \pmod{p}$. A completely similar argument shows that we also have $M^{ed} \equiv M \pmod{q}$.

Now, p and q are relatively prime, since they are distinct primes. It follows then from the Chinese Remainder Theorem (4.3) that M is the unique integer that is at least 0 and less than $n = pq$ and that is congruent to $M^{ed} \pmod{p}$ and to $M^{ed} \pmod{q}$. Since $M^{ed} \equiv M^{ed} \pmod{p}$ and \pmod{q} as well, we may conclude from Theorem (4.3) that $M \equiv M^{ed} \pmod{n}$. ■

- (5.5) **EXAMPLE.** Continuing the example we began above, I have received the message 2480. Recall that I announced $n = 2911$ and $e = 11$. I have the extra information that $n = (41)(71)$; i.e., $p = 41$ and $q = 71$. Therefore, $p - 1 = 40$ and $q - 1 = 70$ and so their least common multiple t equals 280. I must now solve for the multiplicative inverse of $\overline{11}$ in the ring \mathbf{Z}_{280} . This is similar to Exercise 2.15. We use the Euclidean Algorithm and then work our way back up the steps of the algorithm to obtain $1 = 51 \cdot 11 - 2 \cdot 280$. Thus the multiplicative inverse of $\overline{11}$ in \mathbf{Z}_{280} is $\overline{51}$, so we take $d = 51$.

To decrypt the message 2480, we must now compute $(M^e)^d = 2480^{51} \pmod{2911}$. Again, we do this by repeatedly squaring and using the fact that $51 = 32 + 16 + 2 + 1$. In detail, we have

$$\begin{aligned} 2480^2 &\equiv 6150400 \equiv 2368 \pmod{2911} \\ 2480^4 &\equiv 2368^2 \equiv 838 \pmod{2911} \\ 2480^8 &\equiv 838^2 \equiv 693 \pmod{2911} \\ 2480^{16} &\equiv 693^2 \equiv 480249 \equiv 2845 \equiv -66 \pmod{2911} \\ 2480^{32} &\equiv (-66)^2 \equiv 1445 \pmod{2911}. \end{aligned}$$

(We note that it was just an accident that the original message 1415 and the encrypted message 2480 were both roots of the polynomial $X^2 - \overline{2368}$ over \mathbf{Z}_{2911} .) Now we compute

$$\begin{aligned} 2480^{51} &= (2480)^{32+16+2+1} \\ &\equiv (1445)(2845)(2368)(2480) \pmod{2911} \\ &\equiv (693)(2368)(2480) \pmod{2911} \\ &\equiv (2131)(2480) \pmod{2911} \\ &\equiv 1415 \pmod{2911}. \end{aligned}$$

Therefore, we find that the message is 1415, which translates to “NO” under our simple correspondence between letters and two-digit integers. The reader can certainly imagine that a computer can perform these computations quite rapidly, and, in fact, a so-called “RSA chip” has been manufactured. ◀

The fact that each user publicly announces how to send an encrypted message to him has many advantages, but it also seemingly has a big disadvantage. Namely, if everyone knows how to send a secret message to me, then how do I know that the person who “signs” the message is really the one who sent it. For example, how do I know that it is really the Pentagon that is telling me to sail around in circles if everyone knows how to send me that message? Happily, a public-key cryptosystem can take care of the problem of “authenticating” signatures.

(5.6) RSA SIGNATURE AUTHENTICATION. Suppose that User 1 sends a message to User 2 and wishes to “sign” the message in a way that will convince User 2 that he, User 1, really sent it. Suppose User 1 has announced the integers n_1 and e_1 and that his secret decryption exponent is d_1 . Suppose User 2 has announced the integers n_2 and e_2 and that his secret decryption exponent is d_2 . Finally, suppose User 1 wishes to sign his message with the phrase S , which we assume is an integer that is less than the minimum of n_1 and n_2 . Then for this signature portion of the transmission, User 1 does the following. First, he computes $S^{d_1} \bmod n_1$. Let S_1 denote the remainder when S^{d_1} is divided by n_1 . Then User 1 sends to User 2 the integer $S_1^{e_2} \bmod n_2$; that is, he treats S_1 as if it were a message and encrypts it according to RSA encryption (5.2).

For this signature portion of the message, User 2 proceeds as follows. First, he computes, as usual, the number $(S_1^{e_2})^{d_2} \bmod n_2$. As we showed above in (5.4), the remainder when we divide $(S_1^{e_2})^{d_2}$ by n_2 is the integer S_1 . The new wrinkle when working with the signature portion of the message is that now User 2 computes $S_1^{e_1} \bmod n_1$. (Remember, e_1 and n_1 are public knowledge.) Note that

$$S_1^{e_1} = (S^{d_1})^{e_1} \equiv S \bmod n_1.$$

Thus User 2 recovers the signature S by finding the remainder when $S_1^{e_1}$ is divided by n_1 . To review, the steps were

$$\begin{aligned} S &\mapsto S_1 \equiv S^{d_1} \bmod n_1 \mapsto S_1^{e_2} \bmod n_2 \\ &\mapsto (S_1^{e_2})^{d_2} \equiv S_1 \bmod n_2 \mapsto S_1^{e_1} \equiv S \bmod n_1. \end{aligned}$$

Why is User 2 so sure that this signature came from User 1? Because User 1 is the only person who knows d_1 (User 1’s secret decryption exponent) and so is the only person who could have transformed S into S_1 .

There is still a problem. An eavesdropper may listen to the communication and learn the integer $S_1^{e_2} \bmod n_2$. This eavesdropper could then “forge” a message from User 1 to User 2. Thus User 1 should vary his signature in some agreed upon fashion with each transmission; for example, a portion of the current plaintext might be included as part of the signature.

- **(5.7) EXAMPLE.** Suppose that I am User 1 and have announced, as before, $n_1 = 2911$ and $e_1 = 11$, with my secret decryption exponent being $d_1 = 51$. Suppose I am sending my signature to User 2 who has announced $n_2 = 3007$ and $e_2 = 13$ and whose secret decryption exponent is d_2 . (For fun you might want to find d_2 .) Let’s assume my signature is “RL.” First, I convert this, using our simple scheme above, into the integer $S = 1812$. Next, I compute S_1 , the remainder when S^{51} is divided by 2911 and find that $S_1 = 730$. Now I must encrypt 730 for transmission to User 2. Therefore, I compute the remainder when $(730)^{13}$ is divided by 3007, obtaining 2979. User 2 receives the ciphertext 2979 and computes the remainder when $(2979)^{d_2}$ is divided by 3007, obtaining 730. Since User 2 knows that this is the signature portion of the message, he now computes the remainder when $(730)^{11}$ is divided by 2911. (Remember, I have publicly announced 2911 and 11.) This computation yields 1812, which converts to the message “RL.” Thus User 2 knows that the message really came from me. ◀

There have been other proposals for public-key cryptosystems. Some have been based on the so-called “knapsack” problem. If one is given a finite number of integers, then it is a simple matter to find their sum. But suppose one is told a sum (a very large integer, say 62 digits long) and is given a very large collection of integers (say 100 60-digit integers) and is asked to find a subset of these integers such that the integers in this subset add up to the given sum. This is a very difficult problem. However, two attempts by Merkle and Hellman to base a public-key cryptosystem on this problem failed when Shamir and Brickell found algorithms to “crack” the cipher. We note again that there is no proof that the RSA cryptosystem is “unbreakable.”

Finally, we note that we have assumed throughout our discussion that a message that is transmitted is received without any errors. The problem of reliably transferring information, that is, of detecting and correcting errors in transmission, is considered in Chapter V, where we present an introduction to Algebraic Coding Theory.

EXERCISES

- † 5.1. Suppose $n = pq$ is the product of two distinct primes.

- a) Show that $\phi(n) = (p-1)(q-1)$.
 - b) Suppose $(e, \phi(n)) = 1$ and let d be the integer between 0 and $\phi(n)$ such that $ed \equiv 1 \pmod{\phi(n)}$. Show that $M^{ed} \equiv M \pmod{n}$ for all integers M .
- † 5.2. Suppose $n = 377 = (13)(29)$. Then $\phi(n) = 336$, while the least common multiple of 12 and 28 is $t = 84$. Put $e = 5$.
- a) Find d_1 such that $ed_1 \equiv 1 \pmod{84}$.
 - b) Find d_2 such that $ed_2 \equiv 1 \pmod{336}$.
 - c) For $M = 3$, show explicitly that $M^{ed_1} \equiv M^{ed_2} \pmod{377}$.
- † 5.3. Suppose $n = pq$ is the product of two distinct primes.
- a) Show that if one could find $\phi(n)$, then one could determine p and q . (Hint: Use the facts that $\phi(n) = (p-1)(q-1)$ and that $(p-q)^2 = (p+q)^2 - 4n$.)
 - b) Explain why we cannot take the modulus n to be a prime p in setting up an RSA encryption procedure.
- † 5.4. Suppose $n = 33$ and $e = 7$.
- a) Find the RSA decryption exponent d .
 - b) Decrypt the ciphertext "14."
 - c) Decrypt the ciphertext "2827." (Break this up into two blocks.)
- † 5.5. Suppose $n = 77$ and $d = 13$. Decrypt the ciphertext "5737" (assuming that we are using the correspondence A=01, B=02, etc.).
- † 5.6. Suppose $n = 15$ and $e = 3$. Find an integer $M > 1$ such that $M^e \equiv M \pmod{n}$ (i.e., RSA encryption leaves M unchanged).
- 5.7. Suppose $n = 2911$ and $e = 11$. Encrypt the following messages as in Example (5.3).
- a) "OK"
 - b) "HELP" (Break this up into two blocks.)
- 5.8. Suppose $n = 2911$, $e = 11$, $d = 51$. Decrypt the ciphertext "2402" (assuming that we are using the correspondence A=01, B=02, etc.).
- † 5.9. Suppose you are User 1 and you have announced $n_1 = 77, e_1 = 7$. Suppose User 2 has announced $n_2 = 91, e_2 = 5$.
- a) Find your "secret" decryption exponent d_1 .
 - b) If your signature is "Z," what ciphertext would you send to User 2 as the signature portion of your message?
 - c) Show how User 2 would determine that this message really came from you.

6. Rings of Polynomials

In this section, we consider rings of polynomials in one indeterminate with coefficients in a ring R . To introduce the concept of an "indeterminate," we begin with what may seem like a strange definition of a polynomial. One advantage is that this definition only uses terms with which we are familiar. In a short time, we will introduce the more conventional notation for a polynomial. However, be prepared for some careful thinking ahead; you are probably not used to drawing a distinction between a polynomial and a polynomial function, but these concepts, while closely related, are different.

(6.1) DEFINITION. Suppose $(R, +, \cdot)$ is a ring. A *polynomial with coefficients in R* is an infinite sequence

$$(a_0, a_1, a_2, \dots)$$

where $a_i \in R$ for all i and where only a finite number of the a_i are not equal to 0. Two polynomials with coefficients in R

$$(a_0, a_1, a_2, \dots) \text{ and } (b_0, b_1, b_2, \dots)$$

are said to be equal if $a_i = b_i$ for all i .

(6.2) DEFINITIONS. Suppose (a_0, a_1, \dots) is a polynomial with coefficients in R such that $a_n \neq 0$ and $a_i = 0$ for all $i > n$. Then n is called the *degree* of this polynomial. If the polynomial $(a_0, a_1, \dots, a_n, 0, 0, \dots)$ has degree n , then a_n is called the *leading coefficient*. If R is a ring with identity, then a polynomial is called *monic* if its leading coefficient is 1.

We will assign $-\infty$ as the degree of the polynomial $0 = (0, 0, \dots)$ and we make the convention that $-\infty + (-\infty) = -\infty$ and $-\infty + n = -\infty$ for any $n \in \mathbb{Z}$. This allows us to state certain formulas below without treating the 0 polynomial as an exceptional case. We note that many other texts do not assign a degree to the 0 polynomial.

We now define the usual operations of addition and multiplication of polynomials.

(6.3) DEFINITION. Given two polynomials

$$f = (a_0, a_1, \dots, a_i, \dots) \text{ and } g = (b_0, b_1, \dots, b_i, \dots)$$

we define their sum to be the polynomial

$$f + g = (a_0 + b_0, a_1 + b_1, \dots, a_i + b_i, \dots)$$

and their product to be the polynomial

$$fg = (c_0, c_1, \dots, c_i, \dots)$$

where

$$c_i = \sum_{j=0}^i a_j b_{i-j} \text{ for } i = 0, 1, \dots$$

(6.4) NOTATION and DEFINITIONS. We will identify a ring element $a \in R$ with the polynomial $(a, 0, 0, \dots)$. (This amounts to noting that R is isomorphic to the ring of polynomials over R of degree ≤ 0 .) Denote the polynomial $(0, 1, 0, 0, \dots)$ by X . The symbol X is called an “indeterminate.” With the definitions of addition and multiplication above, a polynomial of degree n

$$f = (a_0, a_1, \dots, a_n, 0, 0, \dots)$$

may (and will) be written

$$f(X) = a_0 + a_1X + \dots + a_nX^n$$

and will be called “a polynomial in X over R .” The set of all polynomials in X over R is denoted $R[X]$. We will denote the degree of a polynomial $f(X) \in R[X]$ by $\deg(f(X))$, or simply $\deg(f)$.

We leave to the reader the straightforward (though tedious) checking required to prove the following proposition.

(6.5) PROPOSITION. $(R[X], +, \cdot)$ is a ring, called the ring of polynomials in X over R .

Note that a polynomial here is not the same as a polynomial function and an indeterminate X is not the same as a variable x . To understand what we mean, consider the polynomial $X^2 + X \in \mathbf{Z}_2[X]$. Obviously, this polynomial is not equal to the zero polynomial. However, the polynomial function $f(x) = x^2 + x$ on the field \mathbf{Z}_2 is equal to the zero function since $f(0) = f(1) = 0$. (Two functions f and g are equal on a set S if $f(x) = g(x)$ for all $x \in S$.) To help to keep this distinction straight, we will always denote an indeterminate by a capital letter and a variable by a small letter. Of course, every polynomial with coefficients in R

$$f(X) = a_0 + a_1X + \dots + a_nX^n$$

does give rise to a corresponding polynomial function

$$f(x) = a_0 + a_1x + \cdots + a_nx^n$$

on R , but as we have seen above two different polynomials over R may give rise to the same function on R .

(6.6) NOTATION. Given a polynomial $f(X) = a_0 + a_1X + \cdots + a_nX^n$ over R and an element $a \in R$, by $f(a)$ we mean the value of the corresponding polynomial function at a ; namely, the ring element $a_0 + a_1a + \cdots + a_na^n$.

(6.7) DEFINITION. Suppose $f(X)$ is a polynomial over R . An element $a \in R$ is called a *root* of $f(X)$ if $f(a) = 0$.

In dealing with polynomials in this generality, some unusual phenomena can occur. For example, the square of the polynomial $\bar{2}X$ in the ring $\mathbf{Z}_4[X]$ is 0 (since $\bar{2} \cdot \bar{2} = 0$ in \mathbf{Z}_4), so a polynomial ring may have nonzero nilpotents or zero divisors. However, we now show that if R is an integral domain, then so is $R[X]$.

(6.8) PROPOSITION. Suppose R is an integral domain and suppose $f(X), g(X) \in R[X]$. Then $\deg(fg) = \deg(f) + \deg(g)$.

PROOF. First, note that if $f(X) = 0$ or $g(X) = 0$, then $f(X)g(X) = 0$ and the proposition is true (since $-\infty + n = -\infty$). Now, suppose $\deg(f) = n$ and $\deg(g) = m$. If a_n is the leading coefficient of $f(X)$ and b_m is the leading coefficient of $g(X)$, then $a_nb_m \neq 0$ since R is an integral domain. Thus the leading coefficient of $f(X)g(X)$ is a_nb_m , which is the coefficient of X^{n+m} . Therefore the degree of $f(X)g(X)$ is $n + m$. ■

(6.9) COROLLARY. If R is an integral domain, then so is $R[X]$.

PROOF. It is easy to see that if R is commutative and has an identity, then these properties will hold in $R[X]$ also. Now suppose that $f(X), g(X) \in R[X]$ and $f(X)g(X) = 0$. Then by Proposition (6.8), either $f(X) = 0$ or $g(X) = 0$. Thus, $R[X]$ has no zero divisors except 0. ■

(6.10) DEFINITION. Suppose $f(X), g(X) \in R[X]$. We say that $f(X)$ *divides* $g(X)$, and we write $f(X) | g(X)$, if there exists a polynomial $h(X) \in R[X]$ such that $g(X) = f(X)h(X)$.

We will now restrict ourselves to considering polynomials¹⁸⁴ over a field F . The central theme of the rest of this section is that there are many similarities between the rings $F[X]$ and \mathbf{Z} . In many of the results to come

the proofs are very similar to the proofs of the analogous results about \mathbf{Z} , with the degree of a polynomial playing the role analogous to the absolute value of an integer.

To begin establishing these similarities, we show the crucial fact that there is a Division Algorithm in $F[X]$. This is really just "long division" of polynomials, which you are accustomed to carrying out over \mathbf{Q} or over \mathbf{R} .

(6.11) THEOREM (Division Algorithm for Polynomials). Suppose $f(X), g(X)$ are polynomials in $F[X]$ with $f(X) \neq 0$. Then there exist unique polynomials $q(X)$ and $r(X) \in F[X]$ such that

$$g(X) = q(X)f(X) + r(X) \text{ with } \deg(r) < \deg(f).$$

PROOF. There are two things to prove: the existence of an equation as in the theorem and the specified uniqueness of $q(X)$ and $r(X)$. We first establish the existence of such an equation. Put

$$S = \{g(X) - t(X)f(X) : t(X) \in F[X]\}.$$

(S is thus a subset of $F[X]$ and is obviously nonempty.) If $0 \in S$, then the existence is established. If not, then let $r(X)$ be an element of S of minimum degree and suppose $r(X) = g(X) - q(X)f(X)$. We must show that $\deg(r) < \deg(f)$. Suppose $\deg(r) = m$ and $\deg(f) = n$ and suppose

$$\begin{aligned} f(X) &= a_0 + \cdots + a_n X^n \\ r(X) &= b_0 + \cdots + b_m X^m. \end{aligned}$$

If $m \geq n$, then put

$$r_1(X) = r(X) - \frac{b_m}{a_n} X^{m-n} f(X).$$

(Note that $a_n \neq 0$ since f has degree n and so a_n^{-1} exists since F is a field.) Then the coefficient of X^m in $r_1(X)$ is 0, so $\deg(r_1) < \deg(r)$. Also,

$$\begin{aligned} r_1(X) &= g(X) - q(X)f(X) - \frac{b_m}{a_n} X^{m-n} f(X) \\ &= g(X) - \left(q(X) + \frac{b_m}{a_n} X^{m-n} \right) f(X). \end{aligned}$$

Therefore, $r_1(X) \in S$. But that contradicts the fact that $r(X)$ was an element of S of minimum degree. This contradiction shows that the assumption that $m \geq n$ must be false, so $\deg(r) < \deg(f)$.

Now for the uniqueness part of the proof. Suppose there also exists $\tilde{q}(X)$ and $\tilde{r}(X)$ such that

$$g(X) = \tilde{q}(X)f(X) + \tilde{r}(X) \text{ with } \deg(\tilde{r}) < \deg(f).$$

Then we have

$$0 = (\tilde{q}(X) - q(X))f(X) + (\tilde{r}(X) - r(X)).$$

Therefore, $\deg((\tilde{q} - q)f) = \deg(\tilde{r} - r)$. But the degree of $\tilde{r}(X) - r(X)$ is at most equal to the maximum of the degrees of $\tilde{r}(X)$ and $r(X)$, and this maximum is less than the degree of $f(X)$. Since $\deg((\tilde{q} - q)f) \geq \deg(f)$ if $\tilde{q}(X) - q(X) \neq 0$, we conclude that $\tilde{q}(X) - q(X)$ and $\tilde{r}(X) - r(X)$ must both be the zero polynomial. Hence we have $\tilde{q}(X) = q(X)$ and $\tilde{r}(X) = r(X)$. ■

Long division of polynomials over a finite field proceeds just as long division over \mathbf{Q} or over \mathbf{R} , except that we must keep in mind the addition and multiplication tables of the finite field.

- (6.12) **EXAMPLE.** We divide the polynomial $X^4 + \bar{4}X^2 + 1$ by the polynomial $X^2 + \bar{3}X + \bar{2}$ over \mathbf{Z}_5 (i.e., we view these polynomials as being elements of $\mathbf{Z}_5[X]$). We have

$$\begin{array}{r} X^2 + \bar{2}X + 1 \\ X^2 + \bar{3}X + \bar{2} \overline{) X^4 + \quad \quad 4X^2 + \quad \quad 1} \\ \underline{X^4 + \bar{3}X^3 + \bar{2}X^2} \\ \bar{2}X^3 + \bar{2}X^2 + \\ \underline{\bar{2}X^3 + X^2 + \bar{4}X} \\ X^2 + X + 1 \\ \underline{X^2 + \bar{3}X + \bar{2}} \\ \bar{3}X + \bar{4} \end{array}$$

Looking at one of the computations involved above, we note that over \mathbf{Z}_5 , we have $X - \bar{3}X = X + \bar{2}X = \bar{3}X$. The long division shows that, in terms of the Division Algorithm, we have

$$X^4 + \bar{4}X^2 + 1 = (X^2 + \bar{2}X + 1)(X^2 + \bar{3}X + \bar{2}) + \bar{3}X + \bar{4} \quad \blacktriangleleft$$

As in \mathbf{Z} , the Division Algorithm has many nice consequences.

(6.13) COROLLARY (Factor Theorem). Suppose $f(X) \in F[X]$ and $a \in F$. Then $f(a) = 0$ if and only if $(X - a) \mid f(X)$.

PROOF. By the Division Algorithm, there exist $q(X), r(X) \in F[X]$ such that

$$f(X) = q(X)(X - a) + r(X) \text{ with } \deg(r) < 1.$$

Now, if $f(a) = 0$, then $r(a) = 0$. Since $\deg(r) < 1$ this means that $r(X) = 0$, hence that $(X - a) \mid f(X)$. Conversely, if $(X - a) \mid f(X)$, then $r(X) = 0$ and $f(a) = q(a) \cdot (a - a) = 0$. ■

(6.14) COROLLARY. If $f(X) \in F[X]$ has degree n , then $f(X)$ has at most n roots in F .

PROOF. We prove this by induction on $\deg(f)$. If $\deg(f) = 0$, then $f(X)$ is a constant, nonzero polynomial and hence has no roots in F . Assume the Corollary is true for polynomials of degree k and suppose $\deg(f) = k + 1$. By way of contradiction, suppose $f(X)$ has $k + 2$ distinct roots $a_1, a_2, \dots, a_{k+2} \in F$. Then $(X - a_{k+2}) \mid f(X)$, and we may write $f(X) = (X - a_{k+2})h(X)$ for some $h(X) \in F[X]$ of degree k . Then, by Corollary (6.13), $(a_i - a_{k+2})h(a_i) = 0$ so we have $h(a_i) = 0$ for $i = 1, 2, \dots, k + 1$. But this says that $h(X)$ is a polynomial of degree k with $k + 1$ distinct roots, which contradicts the induction hypothesis. ■

(6.15) COROLLARY. $F[X]$ is a principal ideal domain.

PROOF. Suppose I is an ideal of $F[X]$. If $I = (0)$, then we are done. If $I \neq (0)$, then let $g(X)$ be a polynomial in I of minimum nonnegative degree. We claim that $I = (g(X))$. To see this, suppose $h(X) \in I$ and write $h(X) = q(X)g(X) + r(X)$, where $\deg(r) < \deg(g)$. Then $r(X) \in I$ since $r(X) = h(X) - q(X)g(X)$ and $h(X)$ and $g(X)$ belong to I . But since $\deg(r) < \deg(g)$ and $g(X)$ was an element of I of minimum nonnegative degree, we conclude that $r(X) = 0$. Therefore $h(X) = q(X)g(X)$ and I is generated by $g(X)$. ■

(6.16) DEFINITION. A nonconstant polynomial $p(X) \in F[X]$ is said to be *irreducible over F* if whenever $p(X) = f(X)g(X)$, with $f(X), g(X) \in F[X]$, then either $\deg(f) = 0$ or $\deg(g) = 0$. A nonconstant polynomial that is not irreducible is called *reducible*.

Thus, a polynomial is irreducible if it cannot be written as the product of two polynomials of lesser degree. Irreducible polynomials play the same

role in $F[X]$ that primes play in \mathbf{Z} . Some texts consider constant polynomials to be irreducible, but we do not (in analogy with the fact that 1 is not considered a prime integer).

- **(6.17) EXAMPLE.** Whether or not a polynomial $f(X)$ is irreducible over a field F depends on both $f(X)$ and F . For example, the polynomial $X^2 + 1$ is irreducible over \mathbf{R} , but is reducible over \mathbf{C} . For another example, the polynomial $X^2 + X + 1$ is irreducible over \mathbf{Z}_2 , but is reducible over \mathbf{Z}_3 (we have $X^2 + X + 1 = (X + \bar{2})^2$ over \mathbf{Z}_3). ◀

One difference between irreducible polynomials and prime integers is that irreducible polynomials can divide each other. For example, the polynomials $p(X) = X + \bar{2}$ and $q(X) = \bar{2}X + 1$ are both irreducible over \mathbf{Z}_3 , but $q(X) = \bar{2}p(X)$ and $p(X) = \bar{2}q(X)$. In general, if $q(X)$ is an irreducible polynomial and if $p(X)$ has positive degree and $p(X) \mid q(X)$, then, by the fact that $q(X)$ is irreducible, we must have $q(X) = cp(X)$ where c has degree 0; that is, c must be a nonzero element of F , hence a unit. It follows that $p(X) = c^{-1}q(X)$. Note that $p(X)$ is then also irreducible, since any divisor of $p(X)$ is also a divisor of $q(X)$.

In analogy to the Fundamental Theorem of Arithmetic (I.2.21), we have the following result for polynomials over a field.

(6.18) THEOREM. Any nonconstant polynomial in $F[X]$ can be written as the product of irreducible polynomials. Further, if

$$f(X) = p_1(X)^{s_1} p_2(X)^{s_2} \cdots p_r(X)^{s_r} \text{ and } f(X) = q_1(X)^{t_1} q_2(X)^{t_2} \cdots q_l(X)^{t_l}$$

are two factorizations of $f(X)$ into the product of powers of distinct irreducible polynomials, then $l = r$, and there is a permutation σ of $\{1, 2, \dots, r\}$ such that for each $i = 1, 2, \dots, r$ there is a nonzero element $a_i \in F$ such that $p_i(X) = a_i q_{\sigma(i)}(X)$ and $s_i = t_{\sigma(i)}$ (i.e., the irreducible factors are unique up to order and nonzero constant multiples).

PROOF. The proof of the existence of such a factorization proceeds by induction on the degree of the polynomial. First, note that if $f(X)$ is of degree 1, then $f(X)$ is itself irreducible. Now, assume that the theorem is true for all polynomials of degree $< k$ and suppose $f(X)$ has degree k . If $f(X)$ is irreducible, then we are done. If not, then we may write $f(X) = g(X)h(X)$, where $\deg(g) < k$ and $\deg(h) < k$. By our induction hypothesis, we may then express $g(X)$ and $h(X)$ as the product of irreducible polynomials. The product of these two expressions then gives $f(X)$ as the product of irreducible polynomials. The proof of the “uniqueness” of such a factorization is similar to the proof of uniqueness in the Fundamental Theorem of Arithmetic and will be outlined in several exercises at the end of this section. ■

Note that if $f(X)$ is of degree > 1 and has a root in F , then $f(X)$ is reducible over F by the Factor Theorem (6.13). The converse of this is true if $\deg(f)=2$ or 3 .

(6.19) PROPOSITION. Suppose $f(X) \in F[X]$ has degree 2 or 3. Then $f(X)$ is irreducible if and only if it has no roots in F .

PROOF. We must show that if $f(X)$ is of degree 2 or 3 and has no roots, then it is irreducible. By way of contradiction, suppose $f(X)$ is reducible. Then we may write $f(X) = g(X)h(X)$ with the degree of both $g(X)$ and $h(X)$ being positive. Since $\deg(f)=2$ or 3 , it follows that either $\deg(g)=1$ or $\deg(h)=1$. But a polynomial of degree 1 must be of the form $c(X - a)$ for some $a, c \in F$. Hence $f(X)$ has a factor of the form $(X - a)$ and so a is a root of $f(X)$. This contradicts the assumption that $f(X)$ has no roots in F and finishes the proof. ■

► **(6.20) EXAMPLE.** Note that a polynomial of degree 4 may be reducible over F even if it has no roots in F . For example, the polynomial $X^4 + 2X^2 + 1$ has no roots in \mathbf{R} , but it is reducible since it equals $(X^2 + 1)^2$. ◀

By analogy with the definition of greatest common divisor of two integers that we gave in I.2, we now define the greatest common divisor of two polynomials.

(6.21) DEFINITION. Suppose $f(X), g(X) \in F[X]$. A monic polynomial $d(X) \in F[X]$ is said to be a *greatest common divisor (GCD)* of $f(X)$ and $g(X)$ if:

- 1) $d(X) | f(X)$ and $d(X) | g(X)$ and
- 2) whenever $h(X) | f(X)$ and $h(X) | g(X)$, for some $h(X) \in F[X]$, then $h(X) | d(X)$.

The Euclidean Algorithm for polynomials, Theorem (6.23) below, will demonstrate the existence of GCDs. We leave it as an exercise to show that there is a unique greatest common divisor of any two polynomials in $F[X]$. We will denote the greatest common divisor of $f(X)$ and $g(X)$ by $(f(X), g(X))$.

(6.22) DEFINITION. The polynomials $f(X)$ and $g(X)$ are said to be *relatively prime* if $(f(X), g(X)) = 1$.

Just as the Euclidean Algorithm followed from the Division Algorithm in I.2, so too do we get a Euclidean Algorithm for polynomials from the Division Algorithm for polynomials. Since the proof of the next theorem is

so similar to the proof of the Euclidean Algorithm for integers (Theorem (I.2.11)), we will omit it.

(6.23) THEOREM (Euclidean Algorithm for Polynomials). Let $f(X)$ and $g(X)$ be polynomials in $F[X]$ with $\deg(g) \geq \deg(f) \geq 0$. Applying the Division Algorithm repeatedly, we have

$$g(X) = q_1(X)f(X) + r_1(X) \text{ with } \deg(r_1) < \deg(f)$$

$$f(X) = q_2(X)r_1(X) + r_2(X) \text{ with } \deg(r_2) < \deg(r_1)$$

$$\vdots$$

$$r_{n-2}(X) = q_n(X)r_{n-1}(X) + r_n(X) \text{ with } \deg(r_n) < \deg(r_{n-1})$$

$$r_{n-1}(X) = q_{n+1}(X)r_n(X).$$

If c_k is the leading coefficient of $r_n(X)$, then $\frac{1}{c_k}r_n(X)$ is the greatest common divisor of $f(X)$ and $g(X)$. Further, there exist polynomials $s(X), t(X) \in F[X]$ such that

$$\frac{1}{c_k}r_n(X) = s(X)f(X) + t(X)g(X).$$

(6.24) COROLLARY. Suppose $f(X)$ and $g(X) \in F[X]$. Then $f(X)$ and $g(X)$ are relatively prime if and only if there exist $s(X), t(X) \in F[X]$ such that $1 = s(X)f(X) + t(X)g(X)$.

PROOF. Exercise 6.22.

► **(6.25) EXAMPLE.** We return to the two polynomials of Example (6.12) and compute their GCD. Recall that all our computations here are over \mathbb{Z}_5 . We have

$$X^4 + \bar{4}X^2 + 1 = (X^2 + \bar{2}X + 1)(X^2 + \bar{3}X + \bar{2}) + \bar{3}X + \bar{4}$$

$$X^2 + \bar{3}X + \bar{2} = \bar{2}X(\bar{3}X + \bar{4}) + \bar{2}$$

$$\bar{3}X + \bar{4} = \bar{2}(\bar{4}X + \bar{2}).$$

Therefore, the GCD of our two polynomials is 1 (recall that the GCD must be monic). To express 1 in the form $s(X)f(X) + t(X)g(X)$, we begin with the next-to-last equation above, multiply it through by $\bar{2}^{-1} = \bar{3}$ and “run backward” through the steps of the algorithm. We get

$$1 = \bar{3}(X^2 + \bar{3}X + \bar{2}) - X(\bar{3}X + \bar{4}) = \bar{3}(X^2 + \bar{3}X + \bar{2}) + \bar{4}X(\bar{3}X + \bar{4}).$$

Now we substitute for $\bar{3}X + \bar{4}$ using the first equation in the algorithm above. We obtain

$$\begin{aligned} 1 &= \bar{3}(X^2 + \bar{3}X + \bar{2}) + \bar{4}X[X^4 + \bar{4}X^2 + 1 - (X^2 + \bar{2}X + 1)(X^2 + \bar{3}X + \bar{2})] \\ &= \bar{3}(X^2 + \bar{3}X + \bar{2}) + \bar{4}X[X^4 + \bar{4}X^2 + 1 + (\bar{4}X^2 + \bar{3}X + \bar{4})(X^2 + \bar{3}X + \bar{2})] \\ &= (X^3 + \bar{2}X^2 + X + \bar{3})(X^2 + \bar{3}X + \bar{2}) + \bar{4}X(X^4 + \bar{4}X^2 + 1). \quad \blacktriangleleft \end{aligned}$$

EXERCISES

- 6.1. Suppose F is a field.
- Suppose $d_1(X)$ and $d_2(X)$ are monic polynomials in $F[X]$. Show that if $d_1(X)|d_2(X)$ and $d_2(X)|d_1(X)$, then $d_1(X) = d_2(X)$.
 - Prove that any two nonzero polynomials in $F[X]$ can have only one greatest common divisor.
- 6.2. What are the units in $F[X]$, where F is a field?
- † 6.3. Let $f(X) = X^2 + \bar{2}X + 1$ and $g(X) = \bar{2}X^3 + X + 1$. Find $q(X)$ and $r(X) \in F[X]$ to satisfy the conditions in the Division Algorithm if $F =$
- \mathbf{Z}_3
 - \mathbf{Z}_5 .
- † 6.4. Find all irreducible monic quadratic polynomials in $\mathbf{Z}_5[X]$.
- † 6.5. Find a formula for the number of irreducible monic quadratic polynomials in $\mathbf{Z}_p[X]$, where p is a prime. (Hint: Count the number of reducible monic quadratics.)
- 6.6. Show that $X^2 + 1$ is irreducible over \mathbf{Q} but is reducible over \mathbf{Z}_5 .
- † 6.7. Find all roots of $X^2 + \bar{1}\bar{1}$ in \mathbf{Z}_{12} .
- 6.8. Find all nilpotent polynomials of degree 1 in $\mathbf{Z}_8[X]$.
- 6.9. Find all degree 1 zero divisors in $\mathbf{Z}_6[X]$.
- † 6.10. Find the greatest common divisor of $f(X) = X^3 + X^2 + X$ and $g(X) = X^4 + \bar{2}X^2 + \bar{2}X + 1$ in the ring $\mathbf{Z}_3[X]$ and express the GCD in the form $s(X)f(X) + t(X)g(X)$.
- 6.11. Find the greatest common divisor of $f(X) = \bar{2}X^3 + X + 1$ and $g(X) = X^2 + \bar{2}X + 1$ in the ring $\mathbf{Z}_5[X]$ and express the GCD in the form $s(X)f(X) + t(X)g(X)$.
- 6.12. Find all roots of $X^2 + X + 1$ in the field \mathbf{Z}_7 .
- † 6.13. Suppose $p(X), f(X) \in F[X]$, with $p(X)$ irreducible. Show that either $p(X)|f(X)$ or $(p(X), f(X)) = 1$.
- 6.14. Suppose $p(X), f(X)$, and $g(X)$ are polynomials in $F[X]$. Show that if $p(X)|f(X)g(X)$ and $(p(X), f(X)) = 1$, then $p(X)|g(X)$. (Hint: See the proof of Proposition (I.2.18).)

- 6.15. Suppose $p(X)$ is an irreducible polynomial in $F[X]$ and suppose that $p(X)|f(X)g(X)$. Show that $p(X)|f(X)$ or $p(X)|g(X)$. (Hint: Use Exercises 6.13 and 6.14.)
- 6.16. Extend the result in Exercise 6.15 by showing that if $p(X)$ is irreducible and if $p(X)|f_1(X)f_2(X)\cdots f_n(X)$, then $p(X)|f_i(X)$ for some $i = 1, 2, \dots, n$.
- 6.17. Use Exercise 6.16 and induction on the degree of $f(X)$ to prove the "uniqueness" part of Theorem (6.18).
- 6.18. (Rational Root Theorem) Suppose $f(X) = a_0 + a_1X + \cdots + a_nX^n \in \mathbf{Z}[X]$ and suppose $m/n \in \mathbf{Q}$, with $(m, n) = 1$, is a root of $f(X)$. Show that $m|a_0$ and $n|a_n$.
- 6.19. Let I be the set of all polynomials in $\mathbf{Z}[X]$ with even constant term.
 a) Show that I is an ideal of $\mathbf{Z}[X]$.
 b) Show that I is not a principal ideal. (Hint: Consider 2 and X .)
- 6.20. Show that $X^4 + 3X^2 + 2$ is reducible in $\mathbf{R}[X]$ but has no roots in \mathbf{R} .
- 6.21. Prove Theorem (6.23).
- 6.22. Prove Corollary (6.24).
- 6.23. Let $f(X)$ be a nonzero polynomial in $\mathbf{Z}[X]$. The *content* of $f(X)$ is the GCD of its coefficients. One says that $f(X)$ is *primitive* if its content is 1. Prove that the product of two primitive polynomials is a primitive polynomial. (Hint: If not, then there exists some prime p that divides each coefficient of the product, but p does not divide some coefficient of each of the two primitive polynomials.)
- 6.24. (Gauss' Lemma) Suppose $f(X) \in \mathbf{Z}[X]$ is primitive. Prove that if $f(X)$ is reducible in $\mathbf{Q}[X]$, then it is reducible in $\mathbf{Z}[X]$. (Hint: Assume $f(X)$ is reducible in $\mathbf{Q}[X]$, clear denominators, and use the previous exercise.)
- 6.25. (Eisenstein's Criterion) Suppose $f(X) = a_0 + a_1X + \cdots + a_nX^n \in \mathbf{Z}[X]$. Suppose that for some prime p ,

$$p \nmid a_n, p|a_{n-1}, p|a_{n-2}, \dots, p|a_0, p^2 \nmid a_0.$$

Show that $f(X)$ is irreducible in $\mathbf{Q}[X]$. (Hint: By taking out any common factors of the coefficients, we may assume $f(X)$ is primitive. Now assume $f(X)$ is reducible in $\mathbf{Q}[X]$. Use the previous exercise and the fact that if $p|ab$, then $p|a$ or $p|b$.)

7. Creating Roots and Extending Fields

A “shortcoming” of the real numbers is that not every polynomial with real coefficients has a real root. A well-known example is the polynomial $X^2 + 1$. To remedy this situation, we create an “imaginary” number i to be a root of $X^2 + 1$ (so $i^2 = -1$). The field of complex numbers is the smallest field containing \mathbf{R} and the new number i , and it is an amazing fact (the Fundamental Theorem of Algebra) that now every polynomial with complex coefficients has a complex root, and, in fact, splits completely into a product of linear polynomials in $\mathbf{C}[X]$.

In this section, we want to consider this situation in more generality. Specifically, we will start with an irreducible polynomial with coefficients in a field F and we will “create” a root of this polynomial in a larger field. The key is the following result, which is analogous to Corollary (2.3).

(7.1) PROPOSITION. Suppose $f(X) \in F[X]$. Then the quotient ring $F[X]/(f(X))$ is a field if and only if $f(X)$ is irreducible over F .

PROOF. First, suppose $f(X)$ is reducible over F . Then there exist nonzero polynomials $g(X), h(X) \in F[X]$ such that $\deg(g)$ and $\deg(h)$ are less than $\deg(f)$ and $f(X) = g(X)h(X)$. Since $g(X)$ and $h(X)$ have lesser degree than $f(X)$, they cannot lie in $(f(X))$, so their images, \bar{g} and \bar{h} , in $F[X]/(f(X))$ are nonzero. But $\bar{g}\bar{h} = \bar{f} = 0$, so $F[X]/(f(X))$ has nonzero zero divisors and is therefore not a field.

Conversely, assume $f(X)$ is irreducible over F . We must show that every nonzero element of $F[X]/(f(X))$ has an inverse. Suppose $\bar{g} \in F[X]/(f(X))$ is nonzero and let $g(X) \in F[X]$ be any nonzero polynomial in the coset \bar{g} . Then $g(X) \notin (f(X))$, so we have that $f(X)$ does not divide $g(X)$. Since $f(X)$ is irreducible, this implies, by Exercise 6.13, that the GCD of $f(X)$ and $g(X)$ is 1. Hence, by Corollary (6.24), there exist $s(X), t(X) \in F[X]$ such that $1 = s(X)f(X) + t(X)g(X)$. Taking the image of both sides under the natural homomorphism from $F[X]$ to $F[X]/(f(X))$, we see that $1 = \bar{t}\bar{g}$. Hence \bar{t} is the multiplicative inverse of \bar{g} in $F[X]/(f(X))$. ■

(7.2) COROLLARY. Suppose $f(X) \in F[X]$ is a polynomial of degree ≥ 1 that has no roots in F . Then there exists a field F' such that $F \subseteq F'$ and such that $f(X)$ has a root in F' .

PROOF. Let $p(X)$ be an irreducible factor of $f(X)$ and put

$$F' = F[X]/(p(X)).$$

By Proposition (7.1), F' is a field. Let $\pi : F[X] \rightarrow F'$ be the natural ring homomorphism as in Example (3.9). Recall that we have identified the

elements of F with the polynomials of degree ≤ 0 in $F[X]$. We claim that the restriction of π to F is a 1-1 mapping. Indeed, suppose $a_1, a_2 \in F \subseteq F[X]$ and $\pi(a_1) = \pi(a_2)$. Then, by the definition of equivalence mod $(p(X))$, we have that $p(X)|(a_2 - a_1)$. But $\deg(p) \geq 1$ and $\deg(a_2 - a_1) \leq 0$, so we may conclude that $a_2 - a_1 = 0$, or $a_1 = a_2$. This shows that π maps F isomorphically onto a field (namely, $\pi(F)$) contained in F' . This allows us to identify an element of the image $\pi(F)$ with its preimage in F ; i.e., if $a \in F$, then we can (and will) denote its image in F' by a , instead of by \bar{a} . This identification allows us to consider F to be a field contained in F' and to view $f(X)$ as having coefficients in F' .

Now suppose $p(X) = b_0 + b_1X + \cdots + b_nX^n$ and put $\alpha = \pi(X)$. Then

$$0 = \pi(p(X)) = b_0 + b_1\alpha + \cdots + b_n\alpha^n.$$

Thus, $\alpha = \pi(X) \in F'$ is a root of $p(X)$ and hence a root of $f(X)$. ■

► (7.3) EXAMPLES

1) Returning to the discussion at the start of this section, let $p(X) = X^2 + 1$ and let $F = \mathbf{R}$. Then $p(X)$ is irreducible over \mathbf{R} and to create a root of $p(X)$ we form the field

$$F' = \mathbf{R}[X]/(X^2 + 1).$$

Instead of denoting the image of X in this quotient by α , let's denote it by i . Then $i \in F'$ is a root of $X^2 + 1$. Given any polynomial $g(X) \in \mathbf{R}[X]$, we may apply the Division Algorithm to write

$$g(X) = q(X)(X^2 + 1) + r(X)$$

with $r(X) = a + bX$ for some $a, b \in \mathbf{R}$. Then $\bar{g} = \pi(g) = \bar{r} = a + bi$. Thus the field F' is exactly the field \mathbf{C} of complex numbers.

2) Let $F = \mathbf{Z}_2$ and let $p(X) = X^2 + X + 1$. Since $p(0) = p(1) = 1$, the polynomial $p(X)$ has no roots in \mathbf{Z}_2 and is thus irreducible by Proposition (6.19) (since $\deg(p) = 2$). We create a root of $p(X)$ by forming the quotient field

$$F' = \mathbf{Z}_2[X]/(X^2 + X + 1).$$

Let α denote the image of X in F' . As in the first example, since $\deg(p) = 2$, we see by applying the Division Algorithm that every element in F' is of the form $a + b\alpha$ with $a, b \in \mathbf{Z}_2$. Thus F' has exactly four elements: $0, 1, \alpha, 1 + \alpha$. Note that since $\alpha^2 + \alpha + 1 = 0$, we have that $\alpha^2 = -(1 + \alpha) = 1 + \alpha$

(since $-1 = 1$ in \mathbf{Z}_2). F' is a field of characteristic 2 with four elements (cf. Exercise 2.16). The addition and multiplication tables for F' are

+	0	1	α	$1 + \alpha$
0	0	1	α	$1 + \alpha$
1	1	0	$1 + \alpha$	α
α	α	$1 + \alpha$	0	1
$1 + \alpha$	$1 + \alpha$	α	1	0

.	0	1	α	$1 + \alpha$
0	0	0	0	0
1	0	1	α	$1 + \alpha$
α	0	α	$1 + \alpha$	1
$1 + \alpha$	0	$1 + \alpha$	1	α

What is α ? It is not a number with which we are familiar. It is a new, “created” number in the same sense that the imaginary number i is created. ◀

The preceding example shows that there exists a field with exactly four elements. The main question that we wish to answer in the next section is

For which n does there exist a field with exactly n elements?

Of course, we already know that if p is a prime, then \mathbf{Z}_p is a field with exactly p elements. Can there exist a field with exactly six elements? The answer is no, as we will now show.

A *subfield* of F' is a subset of F' that forms a field under the operations in F' . Note that if F is a subfield of F' , then we may view F' as a vector space over F . Indeed, the necessary algebraic properties of addition of “vectors” (elements of F') and scalar multiplication (which amounts here to multiplying an element of F' by an element of the subfield F) follow immediately from the ring axioms and the fact that F is a subfield of F' .

(7.4) DEFINITION. Suppose F is a subfield of F' . Then the *degree* of F' over F , denoted $[F' : F]$, is the dimension of F' as an F -vector space.

(7.5) PROPOSITION. If F is a finite field, then F must have p^n elements for some prime p and some $n \in \mathbf{P}$.

PROOF. Suppose F is a finite field. Then F must have characteristic p for some prime p by Exercise 2.13. (Recall that this means that $pa = 0$ for all

$a \in F$.) But this is just what is needed for F to be a vector space over \mathbf{Z}_p ; indeed, we claim that the map

$$\begin{aligned}\varphi: \mathbf{Z}_p &\longrightarrow F \\ \bar{m} &\longmapsto m \cdot 1\end{aligned}$$

is a 1-1 ring homomorphism, which would show that we may identify \mathbf{Z}_p with a subfield of F . The fact that F has characteristic p shows that φ is well-defined and then the laws of coefficients in F show that this map is a ring homomorphism. In order to see that it is 1-1, we compute its kernel. Suppose $\varphi(\bar{m}) = 0$. This means that $m1 = 0$ in F . But by Exercise 2.14, this implies that $p|m$ and so $\bar{m} = 0$. Thus the kernel of φ consists only of 0 and φ is 1-1 by Proposition (III.6.10).

Now that we have identified \mathbf{Z}_p with a subfield of F , let $n = [F : \mathbf{Z}_p]$ be the dimension of F as a \mathbf{Z}_p -vector space and let $v_1, v_2, \dots, v_n \in F$ be a basis. Then each element of F can be written in a unique way as a linear combination

$$\bar{a}_1 v_1 + \bar{a}_2 v_2 + \cdots + \bar{a}_n v_n$$

where $\bar{a}_1, \dots, \bar{a}_n \in \mathbf{Z}_p$. Since there are p possibilities for each of these coefficients, the number of elements in F is p^n . ■

So the question we must answer is: For any prime p and any positive integer n , does there exist a field with exactly p^n elements? The answer to this question will be shown in the next section to be yes. By virtue of the following results, one approach to answering this is to look for an irreducible polynomial of degree n in $\mathbf{Z}_p[X]$.

(7.6) PROPOSITION. Suppose $f(X) \in F[X]$ is a polynomial of degree n . Then $R = F[X]/(f(X))$ is a vector space of dimension n over F and a basis for R as a vector space over F is $1, \alpha, \alpha^2, \dots, \alpha^{n-1}$, where α is the image of X in R .

PROOF. The ring axioms, and the fact that we have identified elements of F with polynomials in R of degree ≤ 0 , show that R is a vector space over F . Suppose

$$f(X) = a_0 + a_1 X + \cdots + a_n X^n.$$

Recall from linear algebra that we must show that the set

$$\{1, \alpha, \alpha^2, \dots, \alpha^{n-1}\}$$

196

is linearly independent and generates R as a vector space over F .

To see that these elements generate R , suppose $\bar{g} \in R$ and let $g(X) \in F[X]$ be any element in the coset \bar{g} . Then, as in Example (7.3), apply the Division Algorithm for polynomials to write

$$g(X) = q(X)f(X) + r(X) \text{ with } r(X) = b_0 + b_1X + \cdots + b_{n-1}X^{n-1}.$$

Then

$$\bar{g} = b_0 + b_1\alpha + \cdots + b_{n-1}\alpha^{n-1}.$$

This shows that $1, \alpha, \dots, \alpha^{n-1}$ generate the vector space R over the field of scalars F .

Finally, to prove linear independence, suppose

$$c_0 + c_1\alpha + \cdots + c_{n-1}\alpha^{n-1} = 0$$

where $c_0, c_1, \dots, c_{n-1} \in F$. We must show that $c_0 = c_1 = \cdots = c_{n-1} = 0$. Consider the polynomial

$$h(X) = c_0 + c_1X + \cdots + c_{n-1}X^{n-1} \in F[X].$$

Then $\bar{h} = 0$, so $f(X) | h(X)$. But since $\deg(h) < \deg(f)$, we conclude that $h(X) = 0$; i.e., $c_0 = c_1 = \cdots = c_{n-1} = 0$. ■

(7.7) COROLLARY. If $f(X)$ is an irreducible polynomial of degree n in $\mathbb{Z}_p[X]$, then the number of elements in the field $F' = \mathbb{Z}_p[X]/(f(X))$ is p^n .

► **(7.8) EXAMPLE.** The polynomial $X^3 + X + 1$ is irreducible over \mathbb{Z}_2 (why?). The field $\mathbb{Z}_2[X]/(X^3 + X + 1)$ has the eight elements

$$0, 1, \alpha, \alpha^2, 1 + \alpha, 1 + \alpha^2, \alpha + \alpha^2, \text{ and } 1 + \alpha + \alpha^2. \blacktriangleleft$$

We actually want to see that there is a unique (up to isomorphism) field with p^n elements for any prime p and positive integer n . To see this, we will introduce the concept of a splitting field for a polynomial.

(7.9) DEFINITION. Suppose $F \subseteq K$ are fields and $\alpha_1, \alpha_2, \dots, \alpha_n \in K$. Then we denote by $F(\alpha_1, \alpha_2, \dots, \alpha_n)$ the smallest subfield of K that contains F and $\alpha_1, \alpha_2, \dots, \alpha_n$. One calls $F(\alpha_1, \alpha_2, \dots, \alpha_n)$ the field obtained by *adjoining* $\alpha_1, \alpha_2, \dots, \alpha_n$ to F .

(7.10) DEFINITION. Suppose $f(X) \in F[X]$ is a polynomial of degree n . A field $K \supseteq F$ is called a *splitting field over F for $f(X)$* if the following two properties hold:

- 1) in $K[X]$, $f(X)$ factors into the product of a constant and n (not necessarily distinct) linear polynomials

$$f(X) = a(X - \alpha_1)(X - \alpha_2) \cdots (X - \alpha_n).$$

- 2) $K = F(\alpha_1, \alpha_2, \dots, \alpha_n)$.

In other words, a splitting field for $f(X)$ is a minimal field in which $f(X)$ has $\deg(f)$ roots (where a root α is counted as many times as the linear polynomial $X - \alpha$ appears in the factorization in (1) above).

► (7.11) EXAMPLES

- 1) \mathbf{C} is a splitting field over \mathbf{R} for $X^2 + 1$.
 2) Consider the polynomial $X^3 - 2$ in $\mathbf{Q}[X]$. Since this is of degree 3 and has no roots in \mathbf{Q} , it is irreducible. To create a root, we form the quotient

$$\mathbf{Q}[X]/(X^3 - 2).$$

If we call α_1 the image of X in this quotient, then $\alpha_1^3 = 2$. We may identify α_1 with the real number $\sqrt[3]{2}$. In the polynomial ring $\mathbf{Q}(\alpha_1)[X]$, our polynomial $X^3 - 2 = X^3 - \alpha_1^3$ factors into

$$(X - \alpha_1)(X^2 + \alpha_1 X + \alpha_1^2).$$

The quadratic factor above is irreducible over $\mathbf{Q}(\alpha_1)$ since its roots, by the quadratic formula, are

$$\alpha_2 = \frac{(-1 + \sqrt{-3})\alpha_1}{2}, \quad \alpha_3 = \frac{(-1 - \sqrt{-3})\alpha_1}{2}.$$

The quotient

$$K = \mathbf{Q}(\alpha_1)[X]/(X^2 + \alpha_1 X + \alpha_1^2)$$

is the field

$$K = \mathbf{Q}(\alpha_1, \alpha_2, \alpha_3) \cong \mathbf{Q}(\sqrt[3]{2}, \sqrt{-3})$$

and over K our polynomial splits into the product of three linear polynomials. Note that K “sits” between \mathbf{Q} and \mathbf{C} but not between either \mathbf{Q} and \mathbf{R} or between \mathbf{R} and \mathbf{C} . We leave it as an exercise for the reader to see that K is a 6-dimensional \mathbf{Q} -vector space.

3) Again, consider the irreducible polynomial $f(X) = X^3 + X + 1$ in $\mathbf{Z}_2[X]$. To create a root of this polynomial we form the quotient⁹⁸

$$K = \mathbf{Z}_2(\alpha) = \mathbf{Z}_2[X]/(X^3 + X + 1)$$

as in Example (7.8). Factoring $f(X)$ in the ring $K[X]$ yields

$$X^3 + X + 1 = (X + \alpha)(X^2 + \alpha X + 1 + \alpha^2).$$

(To see this, divide $X^3 + X + 1$ by $X - \alpha = X + \alpha$.) Unfortunately, we do not have the quadratic formula at our disposal in the finite field case to help to solve quadratic polynomials. But, some checking shows that α^2 and $\alpha + \alpha^2$ are roots of $X^2 + \alpha X + 1 + \alpha^2$ (when you check this remember that $\alpha^3 = 1 + \alpha$ in K and that K has characteristic 2). Hence, $f(X)$ actually factors as the product

$$(X + \alpha)(X + \alpha^2)(X + \alpha + \alpha^2)$$

in $K[X]$. Therefore, K is already a splitting field for $f(X)$. ◀

(7.12) THEOREM. A splitting field over F for $f(X)$ always exists.

PROOF. We describe a (theoretical) algorithm for constructing a splitting field.

Step 1. Factor $f(X)$ into the product of irreducible polynomials in $F[X]$. If each irreducible polynomial in this product has degree 1, then F is our splitting field. If not, then let $p_1(X)$ be an irreducible factor of $f(X)$ of degree > 1 and let K_1 denote the field $F[X]/(p_1(X))$. Set $m = 1$.

Step 2. Factor $f(X)$ into the product of irreducible polynomials in $K_m[X]$. (Note that $f(X)$ has at least m factors of degree 1 in $K_m[X]$.) If each irreducible polynomial in this product has degree 1, then K_m is a splitting field over F for $f(X)$, so stop. If not, then let $p_{m+1}(X)$ be an irreducible factor of $f(X)$ of degree > 1 and let K_{m+1} denote the field $K_m[X]/(p_{m+1}(X))$.

Step 3. Replace m by $m + 1$ and go to Step 2.

This algorithm must terminate in at most $\deg(f) - 1$ passes through Step 2, since with each pass we obtain at least one more degree 1 polynomial in the expression for $f(X)$ as a product of irreducible polynomials. ■

We called the above algorithm “theoretical” because it is a very difficult practical problem to factor a polynomial into a product of irreducible polynomials (just as it is very difficult to factor a large integer into a product of primes).

A proof of the next theorem would take us too far afield, so we will omit the proof. The interested reader may consult Lang [19] for a proof.

(7.13) THEOREM. Any two splitting fields over F for $f(X)$ are isomorphic via an isomorphism that is the identity when restricted to F .

We will illustrate the possible problem and solution involved in Theorem (7.13) with an example.

- **(7.14) EXAMPLE.** Consider the polynomial $f(X) = X^8 - X$. We claim that the field

$$K_1 = \mathbf{Z}_2[X]/(X^3 + X + 1)$$

of Examples (7.8)' and (7.11.3) is a splitting field over \mathbf{Z}_2 for $f(X)$. Recall that K_1 consists of the eight elements $0, 1, \alpha, \alpha^2, 1 + \alpha, 1 + \alpha^2, \alpha + \alpha^2$, and $1 + \alpha + \alpha^2$, where $\alpha^3 = 1 + \alpha$ and $-a = a$ for all $a \in K_1$. We leave it to the reader to check that each element of K_1 is a root of $X^8 - X (= X^8 + X)$. Hence in $K_1[X]$, the polynomial $X^8 - X$ splits into a product of linear factors. Furthermore, it is not hard to see that, since $[K_1 : \mathbf{Z}_2] = 3$, K_1 is the smallest field containing \mathbf{Z}_2 and α and α^2 . Therefore, K_1 is a splitting field over \mathbf{Z}_2 for $f(X)$.

Now consider the field

$$K_2 = \mathbf{Z}_2[X]/(X^3 + X^2 + 1).$$

Letting β denote the image of X in K_2 , we see that K_2 consists of the eight elements $0, 1, \beta, \beta^2, 1 + \beta, 1 + \beta^2, \beta + \beta^2$, and $1 + \beta + \beta^2$, where $\beta^3 = 1 + \beta^2$ (so β is not the same as α). Again, we leave it to the reader to check (using the facts that K_2 is of characteristic 2 and $\beta^3 = 1 + \beta^2$) that each of the eight elements of K_2 is a root of $X^8 - X$. It follows as above that K_2 is a (different) splitting field over \mathbf{Z}_2 for $f(X)$.

Even though K_1 and K_2 are different, they must be isomorphic by Theorem (7.13) above. What is the isomorphism? It's not the map that sends α to β ! That map is not a homomorphism. Rather, it's the ring homomorphism that sends α to $1 + \beta$. (The key is that $(1 + \beta)^3 = 1 + (1 + \beta) = \beta$.) By constructing addition and multiplication tables for these two fields, the reader will be able to see that they are isomorphic via this map. ◀

EXERCISES

† 7.1. How many elements are there in each of the following fields:

- $\mathbf{Z}_3[X]/(X^3 + \bar{2}X + 1)$.
- $\mathbf{Z}_5[X]/(X^2 + X + \bar{2})$.
- $\mathbf{Z}_5[X]/(X^3 + \bar{3}X + \bar{2})$.

- † 7.2. Put $K = \mathbf{Z}_3[X]/(X^2 + 1)$. Let α denote the image of X in K . List the nine elements in K . Also, find the multiplicative inverse of each of the following elements of K :
- α .
 - $\alpha + 1$.
 - $2\alpha + 1$.

- † 7.3. Find the multiplicative inverse of $\alpha^3 + \alpha + 1$ in the field

$$K = \mathbf{Z}_3[X]/(X^5 + 2X + 1)$$

where α denotes the image of X in K . (Hint: In $\mathbf{Z}_3[X]$, express 1 in the form $s(X)(X^5 + 2X + 1) + t(X)(X^3 + X + 1)$.)

- 7.4. Construct a field with exactly 49 elements.
- † 7.5. Show that the multiplicative group of nonzero elements in the field $\mathbf{Z}_3[X]/(X^2 + 1)$ is a cyclic group of order 8.
- 7.6. Show that the six elements $1, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}, \sqrt{-3}, 2^{\frac{1}{3}}\sqrt{-3}, 2^{\frac{2}{3}}\sqrt{-3}$ form a basis for $\mathbf{Q}(2^{\frac{1}{3}}, \sqrt{-3})$ as a vector space over \mathbf{Q} .
- 7.7. Suppose $F \subseteq K \subseteq L$ are fields with $[L : K] = m$ and $[K : F] = n$. Show that $[L : F] = mn$. (Hint: If $\{\beta_1, \dots, \beta_m\}$ is a basis for L over K and $\{\alpha_1, \dots, \alpha_n\}$ is a basis for K over F , then show that $\{\alpha_i \beta_j\}$ is a basis for L over F .)
- 7.8. Suppose $f(X)$ is a polynomial of degree n in $F[X]$ and let K be a splitting field over F of $f(X)$. Show that $[K : F] \leq n!$. (Hint: Use induction on n , Proposition (7.6), the proof of Theorem (7.12), and the previous exercise.)
- † 7.9. Suppose $F \subseteq K$ are fields. An element $\alpha \in K$ is said to be *algebraic over F* if α is a root of some nonzero polynomial in $F[X]$. If α is not algebraic over F , then it is called *transcendental over F* . Some major results of the nineteenth century proved that π and e are transcendental over \mathbf{Q} . If α is algebraic over F , then a monic polynomial $f(X) \in F[X]$ of minimum degree such that $f(\alpha) = 0$ is called a *minimal polynomial for α over F* .
- Suppose α is algebraic over F and let $f(X)$ be a minimal polynomial for α over F . If $g(X) \in F[X]$ and if $g(\alpha) = 0$, then show that $f(X) | g(X)$.
 - Prove that an algebraic element has a unique minimal polynomial.

- c) Prove that the minimal polynomial of an algebraic element is an irreducible polynomial.
- d) Suppose α is algebraic over F and $p(X)$ is its minimal polynomial over F . Prove that $F(\alpha) \cong F[X]/(p(X))$. (Hint: Consider the ring homomorphism $\varphi : F[X] \rightarrow F(\alpha)$ given by $\varphi(f(X)) = f(\alpha)$.)
- e) Suppose that $[K : F]$, the dimension of K as an F -vector space, is finite. Prove that every element in K is algebraic over F . (Hint: If $\alpha \in K$, consider $1, \alpha, \alpha^2, \dots$)

7.10. a) Show that

$$K_1 = \mathbf{Z}_3[X]/(X^2 + 1)$$

is a splitting field over \mathbf{Z}_3 for $X^9 - X$.

b) Show that

$$K_2 = \mathbf{Z}_3[X]/(X^2 + X + \bar{2})$$

is also a splitting field over \mathbf{Z}_3 of $X^9 - X$.

- c) Produce an isomorphism from K_1 to K_2 that is the identity when restricted to \mathbf{Z}_3 .

7.11. Let K be a field. An *automorphism* of K is a 1-1 ring homomorphism from K onto itself. Show that the set of all automorphisms of K forms a group under composition.

7.12. Let K denote the field of characteristic 2 with four elements in Example (7.3.2). Prove that the mapping $\varphi(x) = x^2$ is an automorphism of K .

7.13. Let K be a field and let G be a subgroup of the group of automorphisms of K . Put $F = \{y \in K : \varphi(y) = y \text{ for all } \varphi \in G\}$. Prove that F is a field. (F is called the *fixed field* of G .)

7.14. Let K be a field and suppose F is a subfield of K . Let $G(K, F)$ denote the set of all automorphisms of K that leave every element of F fixed (i.e., an automorphism φ of K is in $G(K, F)$ if and only if $\varphi(a) = a$ for all $a \in F$). Prove that $G(K, F)$ is a subgroup of the group of automorphisms of K . The group $G(K, F)$ is called the *Galois group* of K over F .

7.15. If $f(X) \in F[X]$, then the Galois group of the splitting field of $f(X)$ over F is called the *Galois group* of $f(X)$ over F . Find the Galois group of $X^2 + X + 1$ over \mathbf{Z}_2 .

8. Finite Fields

In the previous section we saw that the number of elements in a finite field must be p^n for some prime p . The main result of this section will be that there exists a field with exactly p^n elements for any prime p and positive integer n , and that any two fields with exactly p^n elements are isomorphic. To accomplish this, we will show that a splitting field over \mathbf{Z}_p for the polynomial $X^{p^n} - X$ has exactly p^n elements. We will show that $X^{p^n} - X$ has p^n distinct roots and that these roots form a field.

(8.1) DEFINITION. If $f(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0 \in F[X]$, then we define the *derivative of $f(X)$* , denoted $f'(X)$, to be the polynomial

$$f'(X) = n a_n X^{n-1} + (n-1) a_{n-1} X^{n-2} + \cdots + 2 a_2 X + a_1.$$

Of course, this is just the usual formula for the derivative of a polynomial function of a real variable. However, note that some strange phenomena can occur here. For example, if $f(X) = X^2 \in \mathbf{Z}_2[X]$, then $f'(X) = 0$, since $2 \cdot 1 = 0$ in \mathbf{Z}_2 .

We leave as an exercise the proof of the following lemma, which says that the usual rule for finding the derivative of a product is valid.

(8.2) LEMMA. Suppose $f(X), g(X) \in F[X]$. Then

$$(f(X)g(X))' = f'(X)g(X) + f(X)g'(X).$$

(8.3) DEFINITION. Suppose $f(X) \in F[X]$ and suppose $a \in F$ is a root of $f(X)$. We say that a is a *root of multiplicity m of $f(X)$* if $(X-a)^m | f(X)$ and $(X-a)^{m+1} \nmid f(X)$.

A root of multiplicity 1 is called a *simple* root, while a root of multiplicity greater than 1 is called a *multiple* root. The next result shows how we can use the derivative to determine when a polynomial has a multiple root.

(8.4) PROPOSITION. Suppose $f(X) \in F[X]$ and $a \in F$. Then $(X-a)^2$ divides $f(X)$ if and only if $f(a) = 0$ and $f'(a) = 0$.

PROOF. Suppose $(X-a)^2 | f(X)$ and write $f(X) = q(X)(X-a)^2$. Clearly, $f(a) = 0$. Now, we have

$$f'(X) = 2q(X)(X-a) + q'(X)(X-a)^2.$$

Hence $f'(a) = 0$ as well.

Conversely, suppose $f(a) = f'(a) = 0$. By the Factor Theorem (6.13), we know that there exists $q(X)$ such that $f(X) = q(X)(X - a)$. Then we have

$$f'(X) = q'(X)(X - a) + q(X).$$

Then $q(a) = f'(a) - q'(a)(a - a) = 0$, so there exists $s(X)$ such that $q(X) = s(X)(X - a)$. It follows that $f(X) = s(X)(X - a)^2$, which completes the proof. ■

(8.5) PROPOSITION. Suppose F is a field of characteristic p . If $X^t - 1$ has a multiple root in F , then $p|t$.

PROOF. Suppose $a \in F$ is a multiple root of $f(X) = X^t - 1$. Then $a \neq 0$ since $f(0) = -1 \neq 0$. Note that $(X^t - 1)' = tX^{t-1}$. By Proposition (8.4), we have $f'(a) = ta^{t-1} = 0$. Since $a \neq 0$, we conclude that $t \cdot 1 = 0$, so t must be a multiple of p , the characteristic of F , by Exercise 2.14. ■

(8.6) COROLLARY. For any positive integer n , $X^{p^n-1} - 1$ has no multiple roots in any field of characteristic p .

PROOF. It is obvious that p does not divide $p^n - 1$ for any prime p and positive integer n . ■

We need one more lemma before our main theorem.

(8.7) LEMMA. Suppose F is a field of characteristic p and suppose $a, b \in F$. Then $(a + b)^{p^n} = a^{p^n} + b^{p^n}$ for all $n \geq 0$.

PROOF. By the Binomial Theorem (which is valid in any commutative ring), we have

$$(a + b)^{p^n} = a^{p^n} + p^n a^{p^n-1} b + \cdots + \binom{p^n}{i} a^{p^n-i} b^i + \cdots + b^{p^n}.$$

But since the coefficient of every term on the right side of this equation except the first and the last is a multiple of p and F has characteristic p , the right side reduces to $a^{p^n} + b^{p^n}$. ■

(8.8) THEOREM. For any prime p and any positive integer n , there exists a field with exactly p^n elements. Any two fields with exactly p^n elements are isomorphic via an isomorphism that is the identity when restricted to the subfield \mathbf{Z}_p .

PROOF. Let F be a splitting field over \mathbf{Z}_p of the polynomial $f(X) = X^{p^n} - X$. Since $f(X) = X(X^{p^n-1} - 1)$ and $X^{p^n-1} - 1$ has $p^n - 1$ distinct (nonzero) roots by Corollary (8.6), we know that $f(X)$ has p^n distinct roots in F . If we can show that these p^n roots themselves form a field, then by the minimality property of a splitting field, F must consist of just these p^n elements.

Let S denote the set of p^n roots of $f(X)$ in F . First we see that S is a subgroup of $(F, +)$. Indeed, suppose that $a, b \in S$. Then

$$\begin{aligned} f(a+b) &= (a+b)^{p^n} - (a+b) \\ &= a^{p^n} + b^{p^n} - a - b \text{ by Lemma (8.7)} \\ &= a^{p^n} - a + b^{p^n} - b = 0 \text{ (since } a, b \in S). \end{aligned}$$

Therefore, $a+b \in S$ and since $(F, +)$ is finite, this suffices (by Proposition (III.3.11)) to show that S is a subgroup of $(F, +)$.

Let F^* denote the nonzero elements of F . We next will see that the nonzero elements of S form a subgroup of (F^*, \cdot) . Suppose a and b are nonzero elements of S . Note that ab is then nonzero, since $S \subseteq F$. Then

$$\begin{aligned} f(ab) &= (ab)^{p^n} - ab \\ &= a^{p^n} b^{p^n} - ab = ab - ab = 0. \end{aligned}$$

Thus, we have $ab \in S$ and so the nonzero elements of S form a subgroup of (F^*, \cdot) . This shows that S is a subring of F and also that S itself is a field. By the minimality property of a splitting field, we conclude that $F = S$, so F has exactly p^n elements.

Now suppose F' is another field with exactly p^n elements. Since the order of the group of nonzero elements of F' is $p^n - 1$, we know from Corollary (III.4.10) that $a^{p^n-1} = 1$ for every nonzero $a \in F'$. It follows that each of the p^n elements of F' is a root of $X^{p^n} - X$. Therefore, F' is also a splitting field over \mathbf{Z}_p of $X^{p^n} - X$ and by Theorem (7.13), F' is isomorphic to F via an isomorphism that is the identity when restricted to \mathbf{Z}_p . ■

Since two isomorphic fields may be identified, one usually speaks of “the” splitting field of a polynomial. Hence one speaks of “the” finite field with exactly p^n elements, and this field is often denoted $GF(p^n)$ and is called the *Galois field with p^n elements*.

► (8.9) EXAMPLES

1) Suppose we wish to construct $GF(8)$. Since $8 = 2^3$, this field has characteristic 2, so we form the splitting field over \mathbf{Z}_2 of the polynomial $X^8 - X$. To form this splitting field, we need to factor $X^8 - X$ into irreducible factors in $\mathbf{Z}_2[X]$. One gets

$$X^8 - X = X(X+1)(X^3+X+1)(X^3+X^2+1).$$

By the algorithm in Theorem (7.12) for finding a splitting field, we could now form

$$\mathbf{Z}_2[X]/(X^3+X+1) \text{ or } \mathbf{Z}_2[X]/(X^3+X^2+1).$$

Each of these is a splitting field over \mathbf{Z}_2 for $X^8 - X$. These are the fields K_1 and K_2 of Example (7.14).

2) We now construct $GF(16)$. This is the splitting field over \mathbf{Z}_2 of the polynomial $X^{16} - X$. Factoring this polynomial into irreducible factors in $\mathbf{Z}_2[X]$ is not so easy. It is much easier to find an irreducible polynomial $p(X)$ of degree 4 over \mathbf{Z}_2 . By Corollary (7.7), the quotient ring $\mathbf{Z}_2[X]/(p(X))$ will then be a field with 16 elements (which must be “the” splitting field for $X^{16} - X$ over \mathbf{Z}_2). For $p(X)$, we will take $X^4 + X + 1$. (We leave it to the reader to check that this polynomial is not divisible by any irreducible polynomial over \mathbf{Z}_2 of smaller degree.) Then $GF(16)$ is (isomorphic to)

$$\mathbf{Z}_2[X]/(X^4 + X + 1).$$

Let α denote the image of X in this quotient ring. Then we have $\alpha^4 + \alpha + 1 = 0$. By Proposition (7.6), the 16 elements in $GF(16)$ may be written $0, 1, \alpha, \alpha^2, \alpha^3, 1 + \alpha, 1 + \alpha^2, 1 + \alpha^3, \alpha + \alpha^2, \alpha + \alpha^3, \alpha^2 + \alpha^3, 1 + \alpha + \alpha^2, 1 + \alpha^2 + \alpha^3, 1 + \alpha + \alpha^3, \alpha + \alpha^2 + \alpha^3$, and $1 + \alpha + \alpha^2 + \alpha^3$. ◀

Before proceeding to applications of finite fields in the next section, we prove a few more results concerning the structure of finite fields.

(8.10) THEOREM. Suppose p is a prime. Then $GF(p^m)$ is a subfield of $GF(p^n)$ if and only if $m|n$.

PROOF. Suppose $GF(p^m)$ is a subfield of $GF(p^n)$. Then $GF(p^n)$ is a vector space of dimension d , for some $d \in \mathbf{P}$, over $GF(p^m)$. Therefore, the number of elements in $GF(p^n)$, p^n , equals $(p^m)^d = p^{md}$. Thus, we have $n = md$ and $m|n$.

Conversely, assume $n = md$ for some $d \in \mathbf{P}$. Then

206

$$p^n - 1 = (p^m)^d - 1^d = (p^m - 1) [(p^m)^{d-1} + (p^m)^{d-2} + \cdots + 1],$$

so $(p^m - 1)|(p^n - 1)$. A similar argument then shows that

$$(X^{p^m-1} - 1)|(X^{p^n-1} - 1).$$

Therefore, we have that $(X^{p^m} - X)|(X^{p^n} - X)$. It follows that every root of $X^{p^m} - X$ is a root of $X^{p^n} - X$ and hence that $X^{p^m} - X$ must split into a product of linear factors in the splitting field of $X^{p^n} - X$. But this implies that $GF(p^m)$, the splitting field of $X^{p^m} - X$, is contained in $GF(p^n)$, the splitting field of $X^{p^n} - X$. ■

► **(8.11) EXAMPLE.** What are the subfields of $GF(16)$? Since $16 = 2^4$, the subfields of $GF(16)$ are $GF(2)$ and $GF(4)$. Note that $GF(8)$ is not a subfield of $GF(16)$ since $3 \nmid 4$. ◀

(8.12) THEOREM. The multiplicative group consisting of the nonzero elements in $GF(p^n)$ is a cyclic group of order $p^n - 1$.

PROOF. Let G denote the multiplicative group of nonzero elements in $GF(p^n)$. The point of the theorem is that G is *cyclic*. Note that it follows from Corollary (6.14) that the equation $x^k = 1$ can have at most k solutions in G for any $k \in \mathbf{P}$. The fact that G is cyclic then is a consequence of the following lemma from group theory.

(8.13) LEMMA. Suppose (G, \cdot) is a finite Abelian group such that the equation $x^k = e$, where e is the identity of G , has at most k solutions in G for every $k \in \mathbf{P}$. Then G is cyclic.

PROOF. Put $n = |G|$. Let $a \in G$ be an element of maximum order m . Then $m \leq n$ and we wish to see that $m = n$.

First, suppose that the order of every element of G divides m . Then $g^m = e$ for every $g \in G$, so the equation $x^m = e$ has n solutions. Therefore, $n \leq m$ and we can conclude that $n = m$.

Now suppose that there exists an element $b \in G$ such that $l = o(b) \nmid m$. (We will show that this case does not actually occur.) Then there exists a prime p such that

$$l = p^r l', m = p^s m', p \nmid l', p \nmid m', \text{ and } r > s \geq 0.$$

(That is, some prime must appear to a higher power in the prime factorization of l than it appears in the prime factorization of m .) Now put $g = b^{l'}$ and $h = a^{p^s}$. Then, using the facts that $o(b) = l$ and $o(a) = m$, it is not hard to see that $o(g) = p^r$ and $o(h) = m'$. Since $p \nmid m'$, it follows from Exercise III.3.12 that $o(gh) = p^r m' > p^s m' = m$. But this contradicts the fact that

m was the maximum order of any element of G , so there cannot exist an element whose order does not divide m . The proof is now complete. ■

(8.14) DEFINITION. An element $\gamma \in GF(p^n)$ is called *primitive* if γ is a generator of the multiplicative group of nonzero elements in $GF(p^n)$.

(8.15) COROLLARY. $GF(p^n) = \mathbf{Z}_p(\gamma)$ where γ is any primitive element in $GF(p^n)$.

PROOF. Since $\mathbf{Z}_p(\gamma)$ is the smallest field containing \mathbf{Z}_p and γ , we have that $\mathbf{Z}_p(\gamma)$ contains all powers of γ . Hence $\mathbf{Z}_p(\gamma) = GF(p^n)$ since any nonzero element in $GF(p^n)$ is a power of γ . ■

- **(8.16) EXAMPLE.** We claim that the element α in the field $GF(16)$ in Example (8.9.2) is a primitive element. Obviously, the elements $1, \alpha, \alpha^2$, and α^3 are powers of α . We may use the relation $\alpha^4 = \alpha + 1$ and the fact that this field has characteristic 2 (so the sum of every element with itself is 0) to perform the following calculations:

$$\begin{aligned}\alpha^4 &= \alpha + 1, \alpha^5 = \alpha^2 + \alpha, \alpha^6 = \alpha^3 + \alpha^2, \alpha^7 = \alpha^3 + \alpha + 1, \\ \alpha^8 &= \alpha^2 + 1, \alpha^9 = \alpha^3 + \alpha, \alpha^{10} = \alpha^2 + \alpha + 1, \alpha^{11} = \alpha^3 + \alpha^2 + \alpha, \\ \alpha^{12} &= \alpha^3 + \alpha^2 + \alpha + 1, \alpha^{13} = \alpha^3 + \alpha^2 + 1, \alpha^{14} = \alpha^3 + 1.\end{aligned}$$

This shows that α generates the cyclic group of nonzero elements in this field. ◀

EXERCISES

- † 8.1. List the orders of all fields with fewer than 20 elements.
- † 8.2. Determine all subfields of $GF(64)$.
- 8.3. Suppose F is a field and $f(X), g(X) \in F[X]$.
- Prove that $(f(X) + g(X))' = f'(X) + g'(X)$.
 - Prove that $(f(X)g(X))' = f'(X)g(X) + f(X)g'(X)$. (Hint: One method is to use part (a) and induction on the degree of f .)
- † 8.4. Show that every nonzero element in $GF(8)$, except the element 1, is a primitive element.
- 8.5. Let G be the group of nonzero elements in $GF(16)$ (see Example (8.16)).

- a) Find the order of each element of G .
- b) Find the primitive elements in $GF(16)$.

† 8.6. Find all the primitive elements in $\mathbf{Z}_3[X]/(X^2 + X + \bar{2})$.

8.7. Let α denote the image of X in the field $F = \mathbf{Z}_3[X]/(X^2 + 1)$.

- a) Show that α is not a primitive element, even though $F = \mathbf{Z}_3(\alpha)$.
- b) Find a primitive element in F .

8.8. Show that there are $\phi(p^n - 1)$ primitive elements in $GF(p^n)$, where ϕ is the Euler phi function. (Hint: compare with Example (III.3.13.2).)

8.9. Use Corollary (8.15), Proposition (7.1), and Exercise 7.9.(d) to show that there exists an irreducible polynomial of degree n in $\mathbf{Z}_p[X]$ for any prime p and any positive integer n .

8.10. Let F be a finite field with p^n elements. Suppose γ is a primitive element in F . Given any nonzero element $\beta \in F$, we know we may write $\beta = \gamma^m$ for some positive integer $m < p^n - 1$. This exponent m is called the *logarithm of β* (with base γ). Find the logarithm with base α of each element in the field K_1 of Example (7.14). (The difficulty of finding logarithms in finite fields with many elements prompted Diffie and Hellman [9] to propose a public-key cryptosystem (cf. §5) based on logarithms.)

8.11. Suppose F is a field of characteristic p . Suppose $f(X) \in F[X]$ and $a \in F$. Show that $f(a^p) = (f(a))^p$.

8.12. Suppose F is a field of characteristic p . Show that the map $\psi : F \rightarrow F$ given by $\psi(a) = a^p$ is a ring isomorphism from F onto itself (i.e., an automorphism of F) that is the identity when restricted to the subfield \mathbf{Z}_p . This mapping is called the *Frobenius automorphism*.

8.13. Suppose $f(X)$ is a monic, irreducible polynomial in $GF(p)[X]$. Let α denote the image of X in the field $F = GF(p)[X]/(f(X)) (\cong GF(p^n))$. Then $f(X)$ is called a *primitive polynomial over $GF(p)$* if α is a primitive element in F .

- a) Show that $X^2 + X + 1$ and $X^3 + X + 1$ are primitive polynomials over $GF(2)$.
- b) Example (8.16) shows that the polynomial $X^4 + X + 1$ is a primitive polynomial over $GF(2)$. There are two other irreducible polynomials of degree 4 over $GF(2)$. Find these two polynomials and decide whether or not each is primitive. (One is and one isn't.)
- c) Find a primitive polynomial of degree 2 over $GF(3)$.

- d) Use Exercise 7.9 to show that a polynomial of degree n is primitive over $GF(p)$ if and only if it is the minimal polynomial of a primitive element in $GF(p^n)$. Conclude that for every prime p and every positive integer n there exists a primitive polynomial of degree n over $GF(p)$.

9. Applications of Finite Fields to Combinatorial Design

In this section we will consider two applications of finite fields to the design of experiments. We begin with an agricultural problem.

(9.1) PROBLEM. An agricultural researcher has three different strains of wheat and three different fertilizers. The researcher would like to see which combination of wheat and fertilizer gives the best yield. How should the researcher design an experiment on a field to determine this?

After giving this a little thought, it is clear that one should divide the field into nine equal plots so that one can plant each strain of wheat with each type of fertilizer. However, to design the best possible experiment, one should minimize any effect due to one part of the field being more or less fertile than another part of the field. To do this, one can divide the field into three rows and three columns and try to plant each strain of wheat exactly once in each row and once in each column and to use each type of fertilizer exactly once in each row and once in each column, while still achieving all nine possible combinations of wheat and fertilizer. Is this possible? As we shall see, this amounts to asking whether we can find two orthogonal 3×3 Latin squares.

(9.2) DEFINITION. An $n \times n$ Latin square is an $n \times n$ matrix of symbols (usually taken to be the integers $1, 2, \dots, n$) such that each symbol occurs exactly once in each row and once in each column of the matrix.

► **(9.3) EXAMPLE.** Some examples of Latin squares are

$$M = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}, B = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}.$$

Also, Proposition III.2.8 shows that the multiplication table for a finite group is a Latin square. ◀

(9.4) DEFINITION. Two $n \times n$ Latin squares $A = [a_{ij}]$ and $B = [b_{ij}]$ are said to be *orthogonal* if the n^2 ordered pairs

$$(a_{ij}, b_{ij}) \text{ where } i, j = 1, 2, \dots, n$$

are distinct (equivalently, if these ordered pairs consist of all the possible n^2 ordered pairs with first coordinate from A and second coordinate from B). A set $\{A_1, A_2, \dots, A_m\}$ of Latin squares is called *orthogonal* if A_i and A_j are orthogonal for $i \neq j$ and $i, j \in \{1, 2, \dots, m\}$.

► **(9.5) EXAMPLE.** The Latin squares A and B of Example (9.3) are orthogonal. The Latin squares

$$\begin{bmatrix} 3 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 3 & 2 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{bmatrix}$$

are not orthogonal since $(a_{12}, b_{12}) = (a_{21}, b_{21}) = (2, 3)$. ◀

The two orthogonal 3×3 Latin squares A and B of (9.3) give a solution to Problem (9.1). If we form a matrix of ordered pairs with the first coordinate of the ordered pair from A and the second coordinate from B , we obtain

$$\begin{bmatrix} (1, 1) & (2, 2) & (3, 3) \\ (2, 3) & (3, 1) & (1, 2) \\ (3, 2) & (1, 3) & (2, 1) \end{bmatrix}.$$

These ordered pairs describe how the agricultural researcher can plant the wheat and apply the fertilizer. The first coordinate in each ordered pair gives the strain of wheat and the second coordinate gives the type of fertilizer in each of the nine plots into which the field is divided. Note that each strain of wheat and each type of fertilizer appears exactly once in each row and in each column and all nine possible ordered pairs occur.

We may have made the solution of (9.1) seem too easy. Consider another problem.

(9.6) THE 36 OFFICER PROBLEM. Euler considered the following problem. There are six ranks of officers. One officer of each rank is chosen from each of six different regiments. Can these 36 officers be lined up in rows and columns so that in each row and in each column there is exactly one officer of each rank and exactly one officer from each regiment? A solution

to this problem is equivalent to finding two orthogonal 6×6 Latin squares. Euler conjectured in 1782 that the answer to the above question is *no*. It was not until 1901 that a rigorous proof was given by G. Tarry that there do not exist two orthogonal 6×6 Latin squares.

On the other hand, it has been shown (see Roberts [29]) that there exists a pair of orthogonal $n \times n$ Latin squares for all $n > 2$ except $n = 6$. However, many problems concerning Latin squares are still open; we will mention some below.

We will see how to construct orthogonal $n \times n$ Latin squares if $n = p^r$, where p is prime, by using $GF(p^r)$. First, we want to see that an orthogonal set of $n \times n$ Latin squares can contain at most $n - 1$ Latin squares.

If $A = [a_{ij}]$ is an $n \times n$ Latin square (where each $a_{ij} \in \{1, 2, \dots, n\}$) and $\sigma \in S_n$ is a permutation of $\{1, 2, \dots, n\}$, then by $\sigma(A)$ we mean the Latin square $[\sigma(a_{ij})]$. (We leave it as an exercise to show that this is in fact a Latin square.)

(9.7) LEMMA. Suppose A and B are $n \times n$ Latin squares and suppose σ and τ are permutations in S_n . Then A and B are orthogonal if and only if $\sigma(A)$ and $\tau(B)$ are orthogonal.

PROOF. We have that

$$(\sigma(a_{ij}), \tau(b_{ij})) = (\sigma(a_{kl}), \tau(b_{kl}))$$

if and only if $\sigma(a_{ij}) = \sigma(a_{kl})$ and $\tau(b_{ij}) = \tau(b_{kl})$. But since σ and τ are 1-1 mappings, this is true if and only if $a_{ij} = a_{kl}$ and $b_{ij} = b_{kl}$. Thus the n^2 ordered pairs $(\sigma(a_{ij}), \tau(b_{ij}))$ are distinct if and only if the ordered pairs (a_{ij}, b_{ij}) are distinct. ■

(9.8) PROPOSITION. If $\{A_1, A_2, \dots, A_k\}$ is an orthogonal set of $n \times n$ Latin squares, then $k < n$. Thus there can exist at most $n - 1$ pairwise orthogonal $n \times n$ Latin squares.

PROOF. By applying permutations, if necessary, to each of A_1, A_2, \dots, A_k we may assume that the first row of each of these Latin squares is $1\ 2\ \dots\ n$. (Note that the resulting Latin squares still form an orthogonal set by Lemma (9.7).) Consider the (2,1) entry in each square (i.e., the entry in the second row and first column). Since the (1,1) entry of each square is 1, the (2,1) entry of each square cannot be 1 (or else 1 would appear twice in the first column). Since A_i and A_j are orthogonal if $i \neq j$ and since the (1, m) entry of each square is m for $m = 1, 2, \dots, n$, the (2,1) entries of A_i and A_j must be different if $i \neq j$. Therefore, there are only $n - 1$ possibilities (namely $2, 3, \dots, n$) for the (2,1) entries of A_1, A_2, \dots, A_k , so $k < n$. ■

We now show that if $n = p^r$, then we can construct a set of $n - 1$ pairwise orthogonal $n \times n$ Latin squares using $GF(p^r)$. It is an open problem to determine all n such that there exist $n - 1$ pairwise orthogonal $n \times n$ Latin squares. Also open is the more general problem of determining for each n the maximum number of pairwise orthogonal $n \times n$ Latin squares that actually exist (of course, this maximum number is at most $n - 1$). The next theorem shows that for $n = 2, 3, 4, 5, 7, 8$, and 9 , there exist $n - 1$ pairwise orthogonal $n \times n$ Latin squares and, as noted above, there do not exist even two 6×6 orthogonal Latin squares. It is an open problem to determine the maximum number of pairwise orthogonal 10×10 Latin squares.

(9.9) THEOREM. Let p be a prime and suppose $n = p^r$ for some $r \in \mathbf{P}$. Then there exists an orthogonal set of $n - 1$ $n \times n$ Latin squares.

PROOF. Let

$$\alpha_1 = 1, \alpha_2, \dots, \alpha_{n-1}, \alpha_n = 0$$

denote the elements of $GF(p^r)$. For $k = 1, 2, \dots, n - 1$, define

$$A_k = [a_{ij}^{(k)}]$$

by

$$a_{ij}^{(k)} = \alpha_k \alpha_i + \alpha_j.$$

First, we will see that each A_k is a Latin square. Suppose $a_{ij}^{(k)} = a_{il}^{(k)}$. Then $\alpha_k \alpha_i + \alpha_j = \alpha_k \alpha_i + \alpha_l$, so we have $\alpha_j = \alpha_l$, which implies $j = l$. Thus no element occurs more than once in any row. Now suppose $a_{ij}^{(k)} = a_{lj}^{(k)}$. Then $\alpha_k \alpha_i + \alpha_j = \alpha_k \alpha_l + \alpha_j$, so we have $\alpha_k \alpha_i = \alpha_k \alpha_l$. Since $k \neq n$, we have $\alpha_k \neq 0$, so we may invoke the cancellation law to conclude that $\alpha_i = \alpha_l$ and, hence, that $i = l$. Thus no element occurs more than once in any column and A_k is a Latin square.

Finally, we must show that A_k is orthogonal to A_l if $k \neq l$. Suppose $(a_{ij}^{(k)}, a_{ij}^{(l)}) = (a_{pq}^{(k)}, a_{pq}^{(l)})$. Then we have

$$(9.10) \quad \begin{aligned} \alpha_k \alpha_i + \alpha_j &= \alpha_k \alpha_p + \alpha_q \text{ and} \\ \alpha_l \alpha_i + \alpha_j &= \alpha_l \alpha_p + \alpha_q. \end{aligned}$$

Subtracting the first equation from the second, we see that $\alpha_i(\alpha_k - \alpha_l) = \alpha_p(\alpha_k - \alpha_l)$. But $(\alpha_k - \alpha_l) \neq 0$, so we conclude that $\alpha_i = \alpha_p$ and so $i = p$. The first equation of (9.10) then becomes $\alpha_k \alpha_i + \alpha_j = \alpha_k \alpha_i + \alpha_q$. It follows that $\alpha_j = \alpha_q$ and so $j = q$. Therefore A_k and A_l are orthogonal. ■

- **(9.11) EXAMPLE.** We will use the addition and multiplication tables for $GF(4)$ given in Example (7.3.2) to construct three orthogonal 4×4 Latin squares. We denote the elements of $GF(4)$ by

$$\alpha_1 = 1, \alpha_2 = \alpha, \alpha_3 = 1 + \alpha, \alpha_4 = 0.$$

The (i, j) -entry of the first Latin square A_1 is given by

$$a_{ij}^{(1)} = \alpha_i + \alpha_j.$$

We leave it to the reader to verify that

$$A_1 = \begin{bmatrix} \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 \\ \alpha_3 & \alpha_4 & \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_1 & \alpha_4 & \alpha_3 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{bmatrix}.$$

Replacing α_i by i , we may write

$$A_1 = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 \\ 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}.$$

The (i, j) -entry of A_2 is given by

$$a_{ij}^{(2)} = \alpha\alpha_i + \alpha_j.$$

Computing these entries and replacing α_i by i , we find that

$$A_2 = \begin{bmatrix} 3 & 4 & 1 & 2 \\ 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}.$$

Similarly, the reader may verify that A_3 is given by

$$A_3 = \begin{bmatrix} 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 \\ 1 & 2 & 3 & 4 \end{bmatrix}.$$

We now turn to a somewhat different type of design problem. Here is another "classical" problem.

(9.12) THE KIRKMAN SCHOOLGIRL PROBLEM. In 1847, Rev. Thomas Kirkman considered (and solved) the following problem. A teacher takes her class of 15 girls on a daily walk. The students walk in five rows of three each. The problem is to devise a schedule such that in a week (seven consecutive days) each student walks in a row with each other student exactly once. A solution to this problem is an example of a block design. (For a solution, see Brualdi [6].)

(9.13) DEFINITION. Suppose V is a set with v elements (sometimes called varieties). A collection of subsets $\{B_1, B_2, \dots, B_b\}$ (called blocks) of V is called a (*balanced incomplete*) *block design* with parameters (v, b, r, k, λ) if:

- 1) Each block B_i contains exactly k elements.
- 2) Each element of V occurs in exactly r blocks.
- 3) Each (unordered) pair of elements of V occurs in exactly λ blocks.

A block design with $k = 3$ and $\lambda = 1$ is called a *Steiner triple system*.

To solve the Kirkman schoolgirl problem, what is required is a block design (each block representing a row of three girls on one of the seven days) with $v = 15, b = 35, r = 7, k = 3$, and $\lambda = 1$, such that the 35 blocks may be partitioned into 7 sets of 5 blocks each with each girl occurring once in each of the 7 sets. Thus, the block design needed to solve the Kirkman schoolgirl problem is an example of a Steiner triple system.

We will show here that for certain values of v, b , and k a Steiner triple system may be constructed using a finite field. (Unfortunately, we cannot solve the schoolgirl problem in this way.)

(9.14) THEOREM. Suppose p is a prime and n is a positive integer such that $p^n = 6t + 1$ for some $t \in \mathbf{P}$. Let α be a primitive element in $GF(p^n) = \{\alpha_1, \alpha_2, \dots, \alpha_{p^n}\}$. Then a Steiner triple system with $V = GF(p^n)$, $b = tp^n$, and $r = 3t$ is given by

$$\{B_{01}, \dots, B_{0p^n}, B_{11}, \dots, B_{(t-1)p^n}\}$$

where

$$B_{ij} = \{\alpha^i + \alpha_j, \alpha^{2t+i} + \alpha_j, \alpha^{4t+i} + \alpha_j\}$$

for $i = 0, 1, \dots, t-1$ and $j = 1, 2, \dots, p^n$.

PROOF. First, we need to see that each block has three elements in it. Since the order of α (in the multiplicative group of nonzero elements of $GF(p^n)$)

is $6t$, it follows that α^i, α^{2t+i} , and α^{4t+i} must be different for $i < 2t$. Thus B_{ij} contains three distinct elements for each i and j .

Next, we must see that each element of $GF(p^n)$ occurs exactly $3t$ times in our blocks. If we fix an i , each element of $GF(p^n)$ occurs exactly three times in the blocks $B_{i1}, B_{i2}, \dots, B_{ip^n}$. To see this, note that the elements $\alpha^i + \alpha_j$, for $j = 1, 2, \dots, p^n$, are just the elements in the row of the addition table of $GF(p^n)$ corresponding to α^i . Hence, each element of $GF(p^n)$ occurs exactly once among the elements $\alpha^i + \alpha_j, j = 1, 2, \dots, p^n$. Similarly, each element occurs exactly once among the $\alpha^{2t+i} + \alpha_j$ and once among the $\alpha^{4t+i} + \alpha_j$.

Now we must show that each pair of elements of $GF(p^n)$ occurs in exactly one of the blocks. Since each element $\beta \in GF(p^n)$ occurs in $3t$ blocks and each of these blocks contains two elements of $GF(p^n)$ different from β and since there are $6t$ elements in $GF(p^n)$ different from β , it suffices to show that no pair of elements occurs in more than one block.

The key is to consider the difference of a pair of elements in a block. Note that the difference of a pair of elements in B_{ij} is

$$\pm \alpha^i(\alpha^{2t} - 1) \text{ or } \pm \alpha^{2t+i}(\alpha^{2t} - 1) \text{ or } \pm \alpha^i(\alpha^{4t} - 1).$$

Since $\alpha^{2t} \neq 1$, we may write $\alpha^{2t} - 1 = \alpha^s$ for some $s, 0 \leq s < 6t$. Also, since α has order $6t$, note that $(\alpha^{3t})^2 = 1$, so we see that $\alpha^{3t} = -1$ in $GF(p^n)$. Further, note that $\alpha^{4t} = (\alpha^{2t})^{-1}$, so we have

$$\begin{aligned} \alpha^{4t} - 1 &= \frac{1}{\alpha^{2t}} - 1 = -1(\alpha^{2t} - 1)/\alpha^{2t} \\ &= \alpha^{s+3t}/\alpha^{2t} = \alpha^{s+t}. \end{aligned}$$

It then follows that the difference of a pair of elements in B_{ij} may be written in the form

$$\alpha^{i+s+ct} \text{ where } c \in \{0, 1, \dots, 5\}.$$

Now suppose that for some i , the same pair of elements occurs in B_{ij} and in B_{ik} for $j \neq k$. Since we clearly have

$$\alpha^i + \alpha_j \neq \alpha^i + \alpha_k, \alpha^{2t+i} + \alpha_i \neq \alpha^{2t+i} + \alpha_k, \alpha^{4t+i} + \alpha_j \neq \alpha^{4t+i} + \alpha_k,$$

it follows that the difference of the pair of elements must be of the form α^{i+s+ct} and of the form α^{i+s+dt} for different c and d in $\{0, 1, \dots, 5\}$. Hence we have $\alpha^{(c-d)t} = 1$ with $c - d \neq 0$. But since α has order $6t$, this implies that $6 \mid (c - d)$, which is impossible (since $c, d \in \{0, 1, \dots, 5\}$). 216

Finally, suppose that the same pair of elements occurs in block B_{ij} and in block B_{kl} , where $k \neq i$. Then the difference of the pair of elements is

of the form α^{i+s+ct} and of the form α^{k+s+dt} . Since the order of α is $6t$, it follows that $i + ct = k + dt + 6et$ for some $e \in \mathbf{Z}$. Hence $i \equiv_t k$, which is impossible since i and k are distinct elements of $\{0, 1, \dots, t-1\}$. The proof is now finally complete. ■

- (9.15) **EXAMPLE.** Seven friends buy three season tickets to a season of seven football games. They wish to set up a schedule so that each friend attends three games and every friend attends a game with each of the other six friends. Luckily, one of the friends has read and understood the previous theorem and realizes that what is required is a Steiner triple system. Taking $V = GF(7)$ (so $t = 1$) and noting that $\bar{3}$ is a primitive element, this friend constructs the following block design:

$$\begin{aligned} B_{01} &= \{1, \bar{2}, \bar{4}\} \\ B_{02} &= \{\bar{2}, \bar{3}, \bar{5}\} \\ B_{03} &= \{\bar{3}, \bar{4}, \bar{6}\} \\ B_{04} &= \{\bar{4}, \bar{5}, 0\} \\ B_{05} &= \{\bar{5}, \bar{6}, 1\} \\ B_{06} &= \{\bar{6}, 0, \bar{2}\} \\ B_{07} &= \{0, 1, \bar{3}\}. \end{aligned}$$

This Steiner triple system gives an example of the desired schedule after one indexes the set of seven friends by the elements of $GF(7)$. ◀

EXERCISES

- 9.1. Show that if A is an $n \times n$ Latin square and $\sigma \in S_n$, then $\sigma(A)$ is a Latin square.
- 9.2. If A is a Latin square, then is the transpose A^t a Latin square also?
- 9.3. Construct four pairwise orthogonal 5×5 Latin squares.
- † 9.4. Construct two orthogonal 7×7 Latin squares.
- 9.5. Construct two orthogonal 8×8 Latin squares.
- † 9.6. Given a block design with parameters (v, b, r, k, λ) , show that $vr = bk$.
- 9.7. Given a block design with parameters (v, b, r, k, λ) , prove that $\lambda(v-1) = r(k-1)$. (Hint: There are $v(v-1)/2$ distinct pairs of elements from V .)

Define a $v(v-1)/2 \times b$ matrix $M = [m_{ij}]$ by $m_{ij} = 1$ if the i^{th} pair belongs to the j^{th} block and $m_{ij} = 0$ otherwise. Count the 1's in M by counting by rows and by counting by columns.)

- 9.8. Use the previous two exercises to show that v must be congruent to either 1 or 3 modulo 6 in a Steiner triple system.
- 9.9. In a Steiner triple system with $v = 9$, find the values for b and r .
- † 9.10. Thirteen friends buy 3 season tickets for a season of 26 operas. Devise a schedule so that every friend attends 6 operas and attends one opera with each of the other 12 friends.
- 9.11. Devise a Steiner triple system if $v = 19$.
- 9.12. Devise a Steiner triple system if $v = 25$. (Hint: If we take $GF(25) = \mathbb{Z}_5[X]/(X^2 + X + 2)$ and if we let α denote the image of X in this quotient, then α is a primitive element.)

CHAPTER V

Algebraic Coding Theory

When most people hear the word “code,” they think of spies and secret, hard-to-decipher codes, like the material we considered in IV.5. That is not what this chapter is about. The coding theory we consider here is concerned with the reliable and efficient transfer of information. For example, one might want to transfer a file from one computer to another or from a disk to the computer’s memory. Obviously, we want to do this with complete accuracy, for if the receiving computer misconstrues one “1” as a “0,” for example, then the program may not run properly. If the file is being transferred via a modem, then there is a good chance that “noise” on the phone lines may cause the message to contain errors when it is received. We also want to do this efficiently so that a large file can be transferred relatively quickly. We will not be too concerned with the efficiency of coding schemes here, but we will consider the problem of detecting and correcting errors. This theory is often called the theory of error-correcting codes. This area has its roots in the work in the 1940’s done by Claude Shannon and Richard Hamming.

In the first section, we consider the case of a message consisting of two binary digits as a simple example. Next, we introduce the notion of a linear code. The subsequent two sections deal with correcting errors. In the fifth section, we consider an important class of codes called Hamming codes. In the final section, we consider polynomial codes. Here, our knowledge of polynomials and finite fields will be put to use.

1. An Example

We consider the problem of transferring a message consisting of two binary digits from a sender to a receiver over a channel that is subject to "noise." (Of course, in real applications one would be interested in sending longer blocks of digits.) The two digits in our message may represent a letter, or a number, or a musical note, or perhaps the color or brightness that a certain dot, or pixel, should have in a picture. The sender and receiver may be computers and the channel may be a phone line, or the message might go from a compact disc to the compact disc player via a laser beam, or the sender may be a spacecraft passing by Jupiter that is sending a digitized picture back through space (with all its background "noise") to Earth. In any event, we have to worry that the message that is received may not be exactly the message that was sent. How can we send the information so that errors may be detected and perhaps corrected? For the sake of this discussion, we will assume for the rest of this section that not more than two errors will occur in the transfer of a small number of digits (say not more than six digits).

► (1.1) EXAMPLE: Two-Times Repetition Code.

One simple idea is to send each message twice. Thus, if we wish to send the message 01, then we would actually transmit 0101. We would say that we have *encoded* 01 as the codeword 0101. Suppose, in this case, that the receiver actually receives 0001. Since the receiver knows that we are sending each message twice, it is obvious that an error has occurred. But the receiver has no way of knowing if the correct message is 00 or 01. The same thing would be true whenever one error occurs in the transmission of any of the four possible two-digit binary messages (00, 01, 10, 11). We say that this two-times repetition code can *detect* one error, but it cannot *correct* the error.

Note that if two errors occur in our transmission, then the receiver might receive 0000 when we actually transmitted 0101. Thus, if two errors occur, then the receiver cannot be sure that the message was received accurately. This code cannot detect every possible occurrence of two errors. ◀

► (1.2) EXAMPLE: Overall Parity Check Code.

The previous code could detect, but not correct, one error and could not even detect two errors all of the time. It required that we transmit four digits for every two-digit message. With a little thought, we can accomplish the same thing by transmitting only three digits for every two-digit message. Recall that a string of binary digits is called *even* if it contains an even number of 1's and is *odd* if it contains an odd number of 1's. What we will

do is to extend our message to three digits by appending a 1 if our message is odd and a 0 if our message is even. Thus if we wish to send 01, then what we will actually transmit is 011. (Again, we would say that 01 is *encoded* as 011.)

Suppose one error occurs in the transmission of 011. Then the receiver gets either 111, 001, or 010. But it is then easy to see that an error must have occurred; indeed, 11 and 00 are even strings, so the last parity check digit should be a 0, and 01 is an odd string, so the last parity check digit should be a 1. A similar thing happens with any of our other three two-digit messages. Thus this “overall parity check” code can also detect a single error, and we need only transmit three digits, instead of four digits as in the two-times repetition code. Note that this code cannot correct one error. For example, if 111 is received, then although the receiver realizes an error has occurred, he (she, it) cannot know if the correct message was 011, 101, or 110.

Just as with the two-times repetition code, this overall parity check code cannot even detect two errors. For example, the message 01 would be encoded and sent as 011, but if two errors occur, then 000 might, for example, be received. Now the receiver does not know whether an error has occurred. ◀

► (1.3) EXAMPLE: Three-Times Repetition Code.

Now suppose that we send every message three times. So, for example, we would now encode 01 as the codeword 010101. This three-times repetition code can not only detect a single error, it can *correct* a single error. Indeed, if the above transmission were received as, say, 011101, then the receiver could figure out that the codeword that was sent must have been 010101. (Remember, we are assuming no more than two errors can occur.) It is not hard to see that no matter which of the four codewords (000000, 010101, 101010, or 111111) are transmitted, the receiver will be able to decode the message accurately if only one error occurs. For this reason, this code is called a *1-error-correcting code*.

Now suppose two errors occur in transmission. If the receiver receives 110111, then he knows that an error has occurred. However, he cannot know if the message that was sent was 010101 or 111111 (since he does not know if one or two errors occurred). It is intuitively clear that the receiver’s best guess at the actual transmission would be 111111, thus assuming that only one error occurred. This may or may not be an accurate decoding. We would say that this code can detect, but not correct, two errors.

Note that sometimes this code could actually correct two errors. For example, if 110011 were received, then the receiver could be sure that the codeword sent was 111111 (again, assuming no more than two errors can occur). ◀

► (1.4) EXAMPLE: A (5,2) Linear Code.

For our final code in this section, we consider a way to achieve the same error correction and detection properties as the three-times repetition code by using codewords that are only five, and not six, digits long. For now, we will just pull this code “out of the air.” In the next two sections, we will see how to construct such a code systematically.

We encode our four two-digit messages as follows:

$$\begin{aligned} 00 &\longmapsto 00000 \\ 01 &\longmapsto 01101 \\ 10 &\longmapsto 10011 \\ 11 &\longmapsto 11110. \end{aligned}$$

We will see in the next section that this code is an example of what is called a (5,2) linear binary code.

We claim that if a single error occurs in the transmission of one of these codewords, then the receiver can always accurately decode the message. The key point is that any codeword differs from any other codeword in at least three places. If one error occurs in transmission, then the received message will differ from a codeword in precisely one place. Since any two codewords differ in at least three places, there will be a *unique* codeword that differs in only one place from the received message. The receiver decodes the message to be that unique codeword. For example, if 11101 were received, the receiver would decode this as 01101 since these two strings differ in only one place. (Note that 11101 differs in two places from 11110, in three places from 10011, and in four places from 00000.)

Now suppose two errors occur. Then the received message cannot be one of the codewords (again, since any two of our codewords differ in at least three places). Thus, this code also detects two errors. However, not all messages with two errors will be accurately decoded. For example, if 11000 were received, the receiver would not know if the actual codeword transmitted was 00000 or 11110. ◀

To summarize, we have seen that to obtain an error-detecting or error-correcting code, we must encode our messages by adding extra digits. These extra digits are called *redundancy digits*. The digits in our original messages (i.e., 00, 01, 10, and 11 in the above examples) are called *information digits*.

We have seen that if we add more redundancy digits, then we can achieve better error-correction properties. But we have also seen that the simplest way of adding redundancy digits may not be the most efficient way. For example, the (5,2) code above adds only three redundancy digits to our

two information digits and is a 1-error-correcting code, so it is more efficient than the three-times repetition code, which required four redundancy digits.

EXERCISES

- 1.1. Assume that no more than two errors occur in transmission and that the three-times repetition code of Example (1.3) is being used. Decode (i.e., determine the codeword that was sent) each of the following messages.
- a) 010001
 - b) 111011
 - c) 000100
- † 1.2. Can a four-times repetition code always correct two errors? Can a five-times repetition code always correct two errors?
- † 1.3. Assume that not more than one error occurs in transmission and that the code of Example (1.4) is being used. Decode each of the following messages.
- a) 10111
 - b) 01101
 - c) 10110
 - d) 01111
- † 1.4. Consider the following way to encode two-digit messages:

$$\begin{aligned} 00 &\mapsto 00000 \\ 01 &\mapsto 01101 \\ 10 &\mapsto 10011 \\ 11 &\mapsto 10001. \end{aligned}$$

Is this a 1-error-correcting code? Explain.

- 1.5. Consider the following way to encode two-digit messages:

$$\begin{aligned} 00 &\mapsto 00000 \\ 01 &\mapsto 01101 \\ 10 &\mapsto 11010 \\ 11 &\mapsto 10111. \end{aligned}$$

Is this a 1-error-correcting code? Explain.

† 1.6. Let's "extend" the code in Example (1.4) to one in which the codewords are each six digits long by appending an "overall parity check" to each of the codewords in (1.4). This means that we append a 0 to the codeword if there are an even number of 1's in it, and we append a 1 to the codeword if there are an odd number of 1's in it. The new code will then be

$$00 \mapsto 000000$$

$$01 \mapsto 011011$$

$$10 \mapsto 100111$$

$$11 \mapsto 111100.$$

Is the resulting code still 1-error-correcting? Can this new code correct two errors?

2. Linear Codes

A *binary codeword of length n* is just a string of n binary digits. A *binary (block) code of length n* is a set of binary codewords of length n . In order to prove nice results about codes, we will need to assume that the code satisfies some algebraic properties.

First, we will identify a string of n binary digits with an ordered n -tuple. Thus the set of all binary strings of length n will be identified with the set

$$GF(2)^n = \underbrace{GF(2) \times \cdots \times GF(2)}_{n \text{ times}}.$$

(Recall that $GF(2)$ is the finite field with two elements. We could also denote this set of ordered n -tuples by \mathbf{B}^n or by \mathbf{Z}_2^n , but we want to emphasize the field structure.)

Next, we recall that if F is a field, then F^n is a vector space of dimension n over F . In a vector space, one can add vectors and multiply a vector by a scalar (field element). (Now that you know what a group is, you probably realize that the axioms for a vector space say, in part, that a vector space forms a commutative group under addition. However, scalar multiplication is not an operation on V ; i.e., we do not multiply two vectors and get another vector, so a vector space is different from a ring.) If (a_1, a_2, \dots, a_n) and (b_1, b_2, \dots, b_n) are in F^n and if $c \in F$, then one defines

$$\begin{aligned}(a_1, \dots, a_n) + (b_1, \dots, b_n) &= (a_1 + b_1, \dots, a_n + b_n) \\ c(a_1, \dots, a_n) &= (ca_1, \dots, ca_n)\end{aligned}$$

where the operations of addition and multiplication on the right sides of these equations are the operations in F . It is easy to verify that these definitions satisfy the axioms for a vector space.

Recall that a subset of a vector space V is a *subspace* of V if the subset itself forms a vector space under the operations in V . This is equivalent to saying that the subset is a subgroup of $(V, +)$ and is closed under scalar multiplication. We can now give the definition of a linear code. All our codes will be *block codes*, which means that all codewords in the code have the same length. There are other types of codes (e.g., Huffman codes) in which the length of the codewords is not constant.

(2.1) DEFINITIONS. A *linear binary code of length n* is a subspace of $GF(2)^n$. More generally, a *linear q -ary code of length n* is a subspace of $GF(q)^n$. If the linear q -ary code has dimension k as a subspace of $GF(q)^n$, then it is referred to as an (n, k) code over $GF(q)$. A 3-ary linear code is usually called a *ternary* linear code.

Here, $GF(q)$ denotes the finite field with q elements, and we recall from IV.8 that such a field exists if and only if $q = p^m$ for some prime p and some $m \in \mathbf{P}$. Since a vector space forms a group under addition, we note that every linear q -ary code of length n is a subgroup of $GF(q)^n$.

If C is an (n, k) code over $GF(q)$, then the rational number k/n is called the *rate* of C . If the rate of C is close to 1, then C is an efficient code, since the number of redundancy digits ($= n - k$) will be relatively small compared to the number of information digits ($= k$).

Note that for a nonempty subset of $GF(2)^n$ to form a subspace it is sufficient that it be closed under addition. For then the subset will be a subgroup by Proposition (III.3.11) and thus, in particular, will contain the zero vector. Hence the subset will be closed under scalar multiplication as well, since the only scalars in this case are 0 and 1. Thus a set of binary codewords of length n is a linear code if and only if it is a subgroup (under addition) of $GF(2)^n$. For this reason, some authors refer to binary linear codes as *group codes*.

(2.2) NOTATION. We will denote vectors in $GF(q)^n$ by \hat{x}, \hat{y} , etc. We will view these vectors in four different ways: as ordered n -tuples, as strings of length n , as $1 \times n$ row vectors, and as $n \times 1$ column vectors. The context should make clear which interpretation we are using. For example, if A is a $k \times n$ matrix and we write $A\hat{x}$, then we are viewing \hat{x} as a column vector and we are multiplying $k \times n$ and $n \times 1$ matrices. Also, we will let $\hat{0}_n$ denote the zero vector in $GF(q)^n$.

At this point, we'll review a little more linear algebra. Recall that if

v_1, v_2, \dots, v_m are vectors in a vector space V , then the subspace generated by these vectors, denoted $\langle v_1, \dots, v_m \rangle$, is defined to be the set of all linear combinations of v_1, v_2, \dots, v_m ; i.e.,

$$\langle v_1, \dots, v_m \rangle = \{a_1 v_1 + a_2 v_2 + \dots + a_m v_m : a_i \in F, i = 1, 2, \dots, m\}.$$

Recall also that a set of vectors $\{v_1, \dots, v_k\}$ forms a basis for a vector space if every vector in the space can be written in a unique way as a linear combination of v_1, \dots, v_k . Thus, if the vectors v_1, \dots, v_m are linearly independent (which means there is no nontrivial linear combination of them equal to the zero vector), then they form a basis for $\langle v_1, \dots, v_m \rangle$, and so the dimension of $\langle v_1, \dots, v_m \rangle$ would be m . (Recall that the dimension of a vector space is the cardinality of a basis.)

► (2.3) EXAMPLES

1) The two-times repetition code in Example (1.1) is a binary linear code. The codewords in that code are 0000, 0101, 1010, and 1111. It is easy to see that this set is a subgroup of $GF(2)^4$. Remember that we add coordinate-wise with addition as in $GF(2)$, so that, for example, $1111 + 0101 = 1010$. Indeed, this code is the subspace of $GF(2)^4$ generated by 0101 and 1010, so this is a $(4, 2)$ binary code.

2) The code in Example (1.4) is indeed a $(5, 2)$ code. It's generated by 10011 and 01101. (Note that $11110 = 10011 + 01101$.)

3) For an example of a ternary linear code, consider the subspace of $GF(3)^3$ generated by the vectors $\hat{c}_1 = 121$ and $\hat{c}_2 = 112$. (In this chapter, we will write the elements of $GF(3)$ as 0, 1, and 2, rather than $\bar{0}, \bar{1}$, and $\bar{2}$.) How many codewords are in this code? Note that these two vectors are linearly independent. (For two vectors, this just means that one is not a scalar multiple of the other.) Hence, each codeword may be written uniquely in the form $a_1 \hat{c}_1 + a_2 \hat{c}_2$ where $a_1, a_2 \in GF(3)$. Thus, there are $3 \cdot 3 = 9$ codewords in this code. They are

$$\begin{aligned} &000, 121, 112, 200(= 121 + 112), 212(= 2 \cdot 121), 221(= 2 \cdot 112), \\ &012(= 121 + 2 \cdot 112), 021(= 2 \cdot 121 + 112), \text{ and } 100(= 2 \cdot 121 + 2 \cdot 112). \blacktriangleleft \end{aligned}$$

(2.4) DEFINITION. Suppose C is an (n, k) code over $GF(q)^n$. A $k \times n$ matrix G with entries in $GF(q)$ is called a *generator matrix* of C if the rows of G form a basis of the subspace C .

► **(2.5) EXAMPLE.** Suppose C is the $(5, 2)$ linear binary code of Example (1.4). One generator matrix of C would be

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \quad 226$$

There are other bases for this subspace, and thus there are other generator matrices. Another one is

$$G_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \blacktriangleleft$$

Of course, the rows of a generator matrix are linearly independent, so a generator matrix of an (n, k) code has rank k . Recall that there are three types of elementary row operations on a matrix; namely, transposition of two rows, multiplication of a row by a nonzero scalar, and adding a nonzero multiple of one row to another row. In performing elementary row operations on a matrix, one does not change the subspace spanned by the rows. Hence, if G is a generator matrix of a code and if we perform any sequence of elementary row operations on G , then the resulting matrix is also a generator matrix of the same code. Given any generator matrix of a code, we may perform a sequence of elementary row operations on it to obtain a generator matrix that is in reduced row echelon form. This implies that the first nonzero entry in each row is a 1 (called the "leading 1" of that row) and in each column that contains a leading 1, that 1 is the only nonzero entry in that column. (Also, the leading 1's appear further to the right as we go from one row down to the next.) For example, the matrix G_1 in Example (2.5) is in reduced row echelon form, but the matrix G_2 in that example is not because of the 1 in the second column of the first row. Note that we can get from G_2 to G_1 by adding the second row to the first row. (Remember that over $GF(2)$ adding and subtracting are the same thing.)

It is pretty clear that if we apply the same permutation to each codeword in a code, for example switching the i^{th} and j^{th} digit of each codeword, then the resulting code is not very different from the original one. In fact, we have the following notion for when two codes are equivalent.

(2.6) DEFINITION. Two (n, k) codes C_1 and C_2 over $GF(q)^n$ are called *equivalent* if there is a generator matrix G_1 of C_1 and a generator matrix G_2 of C_2 such that G_2 may be obtained from G_1 by a sequence of elementary column operations of the following types: (i) transposition of two columns and (ii) multiplication of a column by a nonzero scalar.

(2.7) LEMMA. Any (n, k) code is equivalent to one that has a generator matrix of the form

$$\begin{bmatrix} I_k & \vdots & A \end{bmatrix},$$

where I_k denotes the $k \times k$ identity and A is a $k \times (n - k)$ matrix.

PROOF. Any code has a generator matrix in reduced row echelon form. By applying a sequence of column transpositions to such a matrix, we can make sure that the leading 1's occur in the first k columns, thus obtaining a matrix of the form $\begin{bmatrix} I_k & A \end{bmatrix}$. ■

(2.8) DEFINITION. A generator matrix of the form $\begin{bmatrix} I_k & A \end{bmatrix}$ is said to be in *standard form*.

► (2.9) EXAMPLE. Let C be the ternary (i.e., over $GF(3)$) $(6, 3)$ code with generator matrix

$$G = \begin{bmatrix} 2 & 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 2 & 2 \end{bmatrix}.$$

We can obtain a reduced row echelon matrix by performing the following sequence of elementary row operations on G . (Remember that the operations below are addition and multiplication in $GF(3)$.)

G	multiply 1st row $\xrightarrow{\quad}$ by 2	$\begin{bmatrix} 1 & 2 & 0 & 2 & 2 & 1 \\ 2 & 1 & 1 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 2 & 2 \end{bmatrix}$
	add 1st row to $\xrightarrow{\quad}$ 2nd row	$\begin{bmatrix} 1 & 2 & 0 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 2 & 2 \end{bmatrix}$
	add $2 \times$ 2nd row $\xrightarrow{\quad}$ to 3rd row	$\begin{bmatrix} 1 & 2 & 0 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 & 1 & 1 \end{bmatrix}$
	multiply 3rd row $\xrightarrow{\quad}$ by 2	$\begin{bmatrix} 1 & 2 & 0 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 & 2 \end{bmatrix}$

$$\begin{array}{l}
 \text{add } 2 \times 3\text{rd row} \rightarrow \text{to 2nd row} \\
 \text{add 3rd row to 1st row}
 \end{array}
 \begin{bmatrix}
 1 & 2 & 0 & 2 & 2 & 1 \\
 0 & 0 & 1 & 0 & 2 & 2 \\
 0 & 0 & 0 & 1 & 2 & 2 \\
 1 & 2 & 0 & 0 & 1 & 1 \\
 0 & 0 & 1 & 0 & 2 & 2 \\
 0 & 0 & 0 & 1 & 2 & 2
 \end{bmatrix}.$$

The last matrix above is in reduced row echelon form (and is another generator matrix for the code C), but it is not a standard form generator matrix. To obtain a standard form matrix, we can switch the 2nd and 3rd columns and then the 3rd and 4th columns to obtain

$$G' = \begin{bmatrix} 1 & 0 & 0 & 2 & 1 & 1 \\ 0 & 1 & 0 & 0 & 2 & 2 \\ 0 & 0 & 1 & 0 & 2 & 2 \end{bmatrix}.$$

This is a standard form generator matrix for a code C' that is equivalent to C . The code given by G' is not identical to C ; i.e., the rows of G' do not generate the same subspace of $GF(3)^6$ as the rows of G . ◀

If C is an (n, k) code with generator matrix G , then we can use G to *encode* a message of length k , where by encoding a message, we mean associating a codeword to that message. All we have to do is view our message of length k as a $1 \times k$ row vector and then multiply this row vector on the right by the $k \times n$ matrix G . The resulting $1 \times n$ row vector is the codeword associated to our message.

► **(2.10) EXAMPLE.** Suppose C is the $(5, 2)$ binary code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

as in Example (1.4). If we wish to send the message x_1x_2 , where each of x_1 and x_2 is either 0 or 1, then we encode this message as

$$[x_1 \ x_2] \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} = [x_1 \ x_2 \ x_2 \ x_1 \ (x_1 + x_2)].$$

For example, the message 11 is encoded as the codeword 11110. ◀

Note that if C is an (n, k) code and if G is a generator matrix for C that is in standard form, then the first k digits of the codeword associated to a message $x_1x_2 \cdots x_k$ will be $x_1x_2 \cdots x_k$ (i.e., the message itself). This is because the first k columns of G form the $k \times k$ identity matrix.

There is another useful matrix, called a parity check matrix, that is associated to a linear code. Suppose C is an (n, k) code over $GF(q)$ and let G be a generator matrix for C . Then G has rank k . Recall from linear algebra that the null space of G , which is the subspace of $GF(q)^n$ consisting of all solutions to the homogeneous system of linear equations $G\hat{x} = \hat{0}_k$, has dimension $n - k$.

(2.11) DEFINITION. A *parity check matrix* for C is an $(n - k) \times n$ matrix H whose rows form a basis for the null space of a generator matrix for C .

We note that some texts define the parity check matrix to be the transpose of what we are calling the parity check matrix.

► **(2.12) EXAMPLE.** Suppose C is the $(5, 2)$ binary linear code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

To find a parity check matrix for C , we solve the homogeneous system of linear equations

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$$

Hence we must solve

$$x_1 + x_4 + x_5 = 0$$

$$x_2 + x_3 + x_5 = 0.$$

The space of solutions to this homogeneous system of equations has dimension 3 for we may let x_3, x_4 , and x_5 take arbitrary values, say $x_3 = s, x_4 = t, x_5 = u$, and then take $x_1 = -t - u$ and $x_2 = -s - u$. (Of course, over $GF(2)$ the only possible values for s, t , and u are 0 and 1, and subtraction is the same as addition.) We get a basis for the solution space by letting $s = 1, t = 0, u = 0$; then $s = 0, t = 1, u = 0$; then $s = 0, t = 0, u = 1$. This is the procedure we would follow over any field, not just over $GF(2)$. With

$s = 1, t = 0, u = 0$, we have $x_1 = 0, x_2 = -1 = 1, x_3 = 1, x_4 = 0, x_5 = 0$. Thus our first basis vector for the null space of G is 01100. In the same way, one finds that the other two basis vectors for the null space of G are 10010 and 11001. Thus a parity check matrix for C is

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \blacktriangleleft$$

(2.13) LEMMA. Suppose G is a $k \times n$ matrix and H is an $(n - k) \times n$ matrix. Then the vectors that form the rows of H lie in the null space of G if and only if $GH^t = 0$.

PROOF. Let $\hat{x}_i = (x_{i1}, \dots, x_{in})$ denote the i^{th} row of H . Then \hat{x}_i is in the null space of G if and only if

$$G \begin{bmatrix} x_{i1} \\ \vdots \\ x_{in} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}.$$

But the matrix product on the left of the above equation is exactly the i^{th} column of GH^t . ■

Suppose that G is a generator matrix and H is a parity check matrix for an (n, k) code. Note that since $GH^t = 0$, we have $0 = (GH^t)^t = HG^t$. Since G has rank k and H has rank $n - k$, it follows that the rows of G are a basis for the null space of H . By applying the previous Lemma to this “dual” situation, we obtain the following result, which will be important for the next section.

(2.14) PROPOSITION. Suppose C is an (n, k) code over $GF(q)$ and H is a parity check matrix for C . Then a vector $\hat{c} \in GF(q)^n$ is a codeword in C if and only if $H\hat{c} = 0$.

PROOF. $H\hat{c} = 0$ means that \hat{c} is in the null space of H . By our remarks above, this holds if and only if \hat{c} is in the row space of a generator matrix for C and that means that \hat{c} is a codeword in C . ■

Proposition (2.14) explains the name “parity check” matrix. Indeed, by this proposition, a vector \hat{x} is a codeword in C if and only if it satisfies

the homogeneous system of equations $H\hat{x} = 0$, where H is a parity check matrix for C . These homogeneous equations are often called "parity check" equations. This is a very useful way to tell if a string is a codeword; note that even if one knows a generator matrix for the code, it may not be easy to see whether or not a given string is a linear combination of the rows of this generator matrix.

- **(2.15) EXAMPLE.** Let C be the $(5, 2)$ binary code of Example (1.4). We saw in Example (2.12) that a parity check matrix for C is

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Thus $\hat{x} = x_1x_2x_3x_4x_5$ is a codeword in C if and only if

$$H \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Thus the parity check equations that \hat{x} must satisfy are

$$\begin{aligned} x_2 + x_3 &= 0 \\ x_1 + x_4 &= 0 \\ x_1 + x_2 + x_5 &= 0. \end{aligned}$$

So, for example, the vector 01001 is not a codeword in C since it fails to satisfy the first of these parity check equations. ◀

If our generator matrix for C is in standard form, then we can immediately write down a parity check matrix for C by using the following result.

(2.16) PROPOSITION. Suppose $G = \begin{bmatrix} I_k & A \end{bmatrix}$ is a standard form generator matrix for an (n, k) code C . (So I_k denotes the $k \times k$ identity matrix and A is a $k \times (n - k)$ matrix.) Then a parity check matrix for C is

$$H = \begin{bmatrix} -A^t & \vdots & I_{n-k} \end{bmatrix}. \quad 232$$

PROOF. A straightforward computation shows that $GH^t = 0$, hence every row of H is in the null space of G by Lemma (2.13). Since the last $n - k$ columns of H form the $(n - k) \times (n - k)$ identity, it follows that the rank of H is $n - k$. This is also the dimension of the null space of G , so the rows of H form a basis for the null space of G . This was the definition of a parity check matrix. ■

► (2.17) EXAMPLES

1) We could have used Proposition (2.16) to find the parity check matrix for the $(5, 2)$ binary code of Example (1.4), and we would have found the same matrix that we derived in Example (2.12).

2) Let G' be the standard form generator matrix for the ternary code C' in Example (2.9). Then a parity check matrix for C' is

$$H' = \begin{bmatrix} -2 & 0 & 0 & 1 & 0 & 0 \\ -1 & -2 & -2 & 0 & 1 & 0 \\ -1 & -2 & -2 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \blacktriangleleft$$

The same reasoning that we used in proving Proposition (2.16) can also be used to obtain the following “dual” result.

(2.18) PROPOSITION. Suppose $H = \begin{bmatrix} B & I_{n-k} \end{bmatrix}$ is an $(n - k) \times n$ matrix. Then H is a parity check matrix for an (n, k) code C with generator matrix

$$G = \begin{bmatrix} I_k & \vdots & -B^t \end{bmatrix}.$$

EXERCISES

† 2.1. Suppose C is an (n, k) code over $GF(q)$. Show that there are q^k codewords in C .

† 2.2. Let C be the $(5, 3)$ binary code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

- a) List all the codewords in C .
- b) Find a parity check matrix for C .

† 2.3. Let C be the $(4, 2)$ ternary code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 2 & 2 \\ 0 & 1 & 1 & 2 \end{bmatrix}.$$

- a) List all the codewords in C .
- b) Find a parity check matrix for C .

2.4. Let C be the $(5, 3)$ binary code with generator matrix

$$G = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Find a standard form generator matrix for a code equivalent to C .

† 2.5. Let C be the $(4, 2)$ ternary code with generator matrix

$$G = \begin{bmatrix} 2 & 0 & 1 & 1 \\ 1 & 2 & 0 & 2 \end{bmatrix}.$$

- a) Find a standard form generator matrix for C .
- b) Find a parity check matrix for C .

2.6. Let C be the binary linear code generated by 1111. Find a parity check matrix for C .

2.7. Suppose C is an (n, k) code over $GF(q)$. If H is a parity check matrix for C , then, since the rows of H are linearly independent, H is a generator matrix for an $(n, n-k)$ code over $GF(q)$, which is called the code *dual to* C and is denoted C^\perp . A code C is called *self-dual* if $C = C^\perp$. Show that the two-times repetition binary code of Example (2.3.1) is a self-dual code.

† 2.8. Suppose

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

is a parity check matrix for a binary linear code C .

- Find a generator matrix for C .
- Does the vector 111100 belong to C ?
- Does the vector 111101 belong to C ?

2.9. Suppose C is an (n, k) binary code. We can form a new code with codewords of length $n + 1$ by adding an overall parity check to each codeword of C . This means we tack on a 1 to the end of a codeword \hat{c} if there are an odd number of 1's in \hat{c} and we tack on a 0 if there are an even number of 1's in \hat{c} . This process is called *extending* or *expanding* C by adding an overall parity check.

- Show that the extension of C by adding an overall parity check is again a linear code.
- If H is a parity check matrix for C , then show that

$$\bar{H} = \begin{bmatrix} & & & 0 \\ & & & \vdots \\ & H & & \\ & & & 0 \\ 1 & \cdots & 1 & 1 \end{bmatrix}$$

is a parity check matrix for the extension of C .

† **2.10.** Suppose $\sigma \in S_n$, the n^{th} symmetric group. Define an action of S_n on $GF(q)^n$ by

$$\sigma(x_1, x_2, \dots, x_n) = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)}).$$

Let C be an (n, k) code over $GF(q)$. The *group of C* , denoted $\mathcal{G}(C)$, is defined by

$$\mathcal{G}(C) = \{\sigma \in S_n : \sigma(\hat{c}) \in C \text{ for all } \hat{c} \in C\}.$$

- Show that $\mathcal{G}(C)$ is indeed a subgroup of S_n .
- If C is the code generated by 111, then what is $\mathcal{G}(C)$?
- Find the group of the $(3, 2)$ binary code C with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

- Find the group of the $(3, 2)$ binary code C with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

3. Error Detection and Correction

Suppose an (n, k) code C is being used and a string \hat{y} is received. One way to tell if an error has occurred in transmission is to compare \hat{y} to a list of all the codewords in C . However, Proposition (2.14) gives us a much better tool for determining if an error has occurred.

(3.1) PROPOSITION. Let H denote a parity check matrix for C and suppose \hat{y} is a received string. If $H\hat{y} \neq \hat{0}_{n-k}$, then an error has occurred in transmission.

Notice that the above proposition is not an “if and only if” statement. It is possible that errors have occurred that transform one codeword into another codeword.

We saw in §1 of this chapter that the key to determining how many errors a given code can detect or correct is the number of places in which two codewords differ. This motivates the following definition.

(3.2) DEFINITION. Suppose $\hat{x}, \hat{y} \in GF(q)^n$. The (*Hamming*) distance between \hat{x} and \hat{y} , denoted $d(\hat{x}, \hat{y})$, is the number of places in which \hat{x} and \hat{y} differ.

► (3.3) EXAMPLES

- 1) If $\hat{x} = 0110$ and $\hat{y} = 1010$, then $d(\hat{x}, \hat{y}) = 2$.
- 2) If $\hat{x} = 021211$ and $\hat{y} = 012122$, then $d(\hat{x}, \hat{y}) = 5$. ◀

Notice that $d(\hat{x}, \hat{y})$ is a nonnegative integer; it is not an element of $GF(q)$.

To determine the number of errors that a code may detect or correct, we need to know the minimum distance between any two codewords.

(3.4) DEFINITION. The *minimum distance* of C , denoted $d(C)$, is defined by

$$d(C) = \min\{d(\hat{c}_1, \hat{c}_2) : \hat{c}_1, \hat{c}_2 \in C, \hat{c}_1 \neq \hat{c}_2\}.$$

If C is an (n, k) code with minimum distance d , then C is called an (n, k, d) code. The numbers n , k , and d are called the *parameters* of C .

According to Definition (3.4), if we wish to determine $d(C)$, we could check the distance between all possible pairs of codewords in C . However, the algebraic properties of a linear code make this determination easier.

(3.5) DEFINITION. Suppose $\hat{x} \in GF(q)^n$. The *weight* of \hat{x} , denoted $w(\hat{x})$, is the number of nonzero places in \hat{x} . Equivalently, $w(\hat{x}) = d(\hat{x}, \hat{0}_n)$.

Note that we have $d(\hat{x}, \hat{y}) = w(\hat{x} - \hat{y})$.

The next result says that we can determine the minimum distance of a linear code of length n by just finding the distance between each codeword and $\hat{0}_n$ (instead of between every possible pair of codewords).

(3.6) PROPOSITION. Suppose C is an (n, k) code and $C \neq \{\hat{0}_n\}$. Then $d(C) = \min\{w(\hat{c}) : \hat{c} \in C, \hat{c} \neq \hat{0}_n\}$.

PROOF. Put $m = \min\{w(\hat{c}) : \hat{c} \in C, \hat{c} \neq \hat{0}_n\}$. Since $w(\hat{c}) = d(\hat{c}, \hat{0}_n)$ and $\hat{0}_n \in C$, we clearly have $d(C) \leq m$. Now suppose \hat{c}_1 and \hat{c}_2 are codewords such that $d(\hat{c}_1, \hat{c}_2) = d(C)$. Then $\hat{c}_1 - \hat{c}_2$ is a nonzero codeword in C and $w(\hat{c}_1 - \hat{c}_2) = d(\hat{c}_1, \hat{c}_2) = d(C)$. Thus we have $m \leq d(C)$, so we may conclude that $m = d(C)$. ■

► (3.7) EXAMPLES

1) Let C be the $(5, 2)$ binary code of Example (1.4). The nonzero codewords of C are 10011, 01101, and 11110. The weights of these codewords are 3, 3, and 4, respectively. Thus $d(C) = 3$ and we would say that C is a $(5, 2, 3)$ binary code.

2) Let C' be the $(6, 3)$ ternary code of Example (2.9) generated by the codewords 100211, 010022, and 001022. The weights of these generators are 4, 3, and 3, respectively, but do not jump to the conclusion that $d(C') = 3$. Indeed, $2 \cdot 010022 + 001022 = 021000$, so that 021000 is a codeword and $d(C') \leq 2$. It is not hard to see that each nonzero codeword in this code must have weight at least 2, so we have $d(C') = 2$. The moral of this example is that one cannot tell $d(C)$ by simply finding the weights of a set of generators. ◀

Suppose C is an (n, k) code over $GF(q)$. We make the following assumptions about our transmission channel:

- 1) Each "digit" (i.e., element of $GF(q)$) has the same probability of being transmitted correctly.
- 2) If an error occurs in transmitting an element of $GF(q)$, then any of the other elements of $GF(q)$ is equally likely to be received (so, for example, if C is a ternary code and an error occurs in the transmission of 0, then it is just as likely that the receiver gets a 1 as that the receiver gets a 2).
- 3) Errors are independent of each other (so, for example, if an error occurs in the transmission of the second element in a string, then that does not affect the probability that the third element will be transmitted correctly).

We will further assume that the probability that a digit is correctly transmitted is greater than 0.5. Under these assumptions, it is more likely that fewer errors occur than more errors. So, for example, if a received string differs in only one place from a codeword \hat{c}_1 and in two places from a codeword \hat{c}_2 , then it is more likely that the message sent was \hat{c}_1 . Hence we would decode the received string as the codeword \hat{c}_1 . This is called *maximum likelihood* or *nearest neighbor* decoding. Using this method of decoding, we can now prove

(3.8) THEOREM. Suppose C is an (n, k, d) code.

- 1) C can detect t errors if $d \geq t + 1$.
- 2) C can correct t errors if $d \geq 2t + 1$.

PROOF. The proof is not difficult. For part (1), if a codeword \hat{c} is sent and at least one and at most t errors occur in the transmission, then the received string is not a codeword, since every other codeword differs from \hat{c} in at least $t + 1$ places. Thus by, say, looking at a list of all codewords we will see that the received string is not a codeword. Our code has detected an error or errors.

For part (2), if at most t errors occur in transmitting \hat{c} , then \hat{c} will be the closest (in Hamming distance) codeword to the received string \hat{y} . Indeed, we know that $d(\hat{c}, \hat{y}) \leq t$ and we know that $d(\hat{c}, \hat{c}') \geq 2t + 1$, where \hat{c}' is any other codeword. It follows from Exercise 3.1 that $d(\hat{y}, \hat{c}') \geq t + 1$ for every codeword $\hat{c}' \neq \hat{c}$. Thus, using nearest neighbor decoding, we correctly decode \hat{y} as \hat{c} . ■

As an illustration of Theorem (3.8), we can state

(3.9) COROLLARY. If $d(C) = 3$, then C is a 1-error-correcting code.

Thus to construct a 1-error-correcting code, we would just need to ensure that the weight of each nonzero codeword is at least 3. This is not quite as easy as we made it sound. For remember that we cannot determine $d(C)$ from just knowing the weights of a set of generators; we would have to consider all linear combinations of the generators as well. The next result shows that if we want to construct a code with a given minimum distance, then first we should construct a parity check matrix for the code.

(3.10) THEOREM. Suppose C is an (n, k) code over $GF(q)$ with parity check matrix H . Then $d(C) = d$ if and only if every set of $d - 1$ columns of H is a linearly independent set (of vectors in $GF(q)^{n-k}$) and some set of d columns of H is a linearly dependent set.

PROOF. Let $\hat{h}_1, \hat{h}_2, \dots, \hat{h}_n$ denote the columns of H . Each column of H is an $(n-k) \times 1$ matrix that we may view as a vector in $GF(q)^{n-k}$. Suppose $\hat{x} = x_1x_2 \cdots x_n \in GF(q)^n$. Then $\hat{x} \in C$ if and only if $H\hat{x} = \hat{0}_{n-k}$. Now, notice that

$$H\hat{x} = x_1\hat{h}_1 + x_2\hat{h}_2 + \cdots + x_n\hat{h}_n.$$

Thus, to each codeword $\hat{x} = x_1x_2 \cdots x_n$ of positive weight w there corresponds a linear dependence relation

$$x_{i_1}\hat{h}_{i_1} + x_{i_2}\hat{h}_{i_2} + \cdots + x_{i_w}\hat{h}_{i_w} = \hat{0}_{n-k}$$

among w of the columns of H , where x_{i_1}, \dots, x_{i_w} are the nonzero coordinates of the vector \hat{x} (so $\{i_1, \dots, i_w\} \subseteq \{1, \dots, n\}$). On the other hand, if there were a (nontrivial) linear dependence relation among some set of d' columns of H , then there is a codeword of positive weight at most d' ; indeed, suppose

$$a_{i_1}\hat{h}_{i_1} + a_{i_2}\hat{h}_{i_2} + \cdots + a_{i_{d'}}\hat{h}_{i_{d'}} = 0$$

with not all $a_{i_j} = 0$. Then the string \hat{x} of length n with a_{i_j} in the i_j position and 0's elsewhere is of positive weight at most d' and satisfies $H\hat{x} = 0$, hence is a codeword in C .

Now assume $d(C) = d$. Then there is a nonzero codeword of weight d , so there is a set of d columns of H that is a linearly dependent set. If some set of $d-1$ columns of H were a linearly dependent set, then there would be a nonzero codeword of weight less than d , which would contradict the fact that $d(C) = d$.

Conversely, if every set of $d-1$ columns of H is a linearly independent set, then there is no nonzero codeword of weight at most $d-1$ and if some set of d columns is a linearly dependent set, then there is a nonzero codeword of weight at most d . It follows that $d(C) = d$. ■

For example, if we want to construct an (n, k) code C over $GF(q)$ that is 1-error-correcting (so has $d(C) \geq 3$), we should construct an $(n-k) \times n$ matrix H with entries from $GF(q)$ such that any two columns of H are linearly independent; i.e., such that no column is a scalar multiple of any other column. Then C is the code with parity check matrix H .

For emphasis, we state again that Theorem (3.10) refers to a parity check matrix and not to a generator matrix. You cannot use the columns of a generator matrix to determine the minimum distance of a code.

► (3.11) EXAMPLES

1) Suppose we want to construct a $(5, 2)$ binary code that is 1-error-correcting. Using Theorem (3.10), we should first construct a 3×5 matrix of 0's and 1's with distinct nonzero columns (since the only scalars here are 0 and 1). One such matrix would be

$$H' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

However, since we are interested in finding a generator matrix for our code, it is more convenient to take our parity check matrix in the form $\begin{bmatrix} -A^t : I_{n-k} \end{bmatrix}$ as in Proposition (2.16), since then a generator matrix will be $\begin{bmatrix} I_k : A \end{bmatrix}$. So, let's take

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Then a generator matrix for the code we seek is

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

This is the code of Example (1.4).

2) Consider the situation in §1 where we wish to encode the two-digit binary messages 00, 10, 01, and 11. Suppose we want to construct a 2-error-correcting code. If we wish to use a repetition code, then we would have to use a five-times repetition code (for then the minimum weight of a nonzero codeword would be 5 and the code would be 2-error-correcting by Theorem (3.8)). This five-times repetition code would be a $(10, 2, 5)$ binary code, requiring 8 redundancy digits. We should be able to do better. Let's use Theorem (3.10) to construct an $(8, 2, 5)$ binary code. This will be 2-error-correcting (since $d(C) = 5$) and will only require 6 redundancy digits. We need to construct a 6×8 matrix of 0's and 1's such that every set of four columns is linearly independent, or, equivalently, such that no column equals the sum of at most three other columns. This matrix will be a parity check

matrix for the desired code. It is not hard to see that the matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

has this property. By Proposition (2.18), the corresponding generator matrix for an $(8, 2, 5)$ binary code is

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

The codewords in this code C are

$$00000000, 10111100, 01001111, \text{ and } 11110011.$$

Thus we see that indeed $d(C) = 5$. ◀

Theorem (3.8) is satisfying from a theoretical standpoint, but what we really need is an *effective* way to implement nearest neighbor decoding. That is, we do not want to have to make a list of all the codewords and then compute the Hamming distance from each received string to every codeword in our list. In the next section, we will make use of the group structure of a linear code to implement nearest neighbor decoding in a reasonably effective way. For now, we consider a decoding scheme we could use in connection with a 1-error-correcting code.

(3.12) A POSSIBLE DECODING SCHEME TO USE WITH A 1-ERROR-CORRECTING CODE. Suppose C is a 1-error-correcting (n, k) code and let H be a parity check matrix for C . Suppose the sender transmits the codeword \hat{c} and the receiver receives the string \hat{y} . Call $\hat{e} = \hat{y} - \hat{c}$ the error string. Now form the matrix product $H\hat{y}$. Note that

$$H\hat{y} = H(\hat{c} + \hat{e}) = H\hat{c} + H\hat{e} = \hat{0}_{n-k} + H\hat{e} = H\hat{e}.$$

There are three possibilities:

$$1) H\hat{y} = \hat{0}_{n-k}.$$

In this case, we know from Proposition (2.14) that \hat{y} is a codeword, so we presume that $\hat{e} = \hat{0}_n$ and we decode \hat{y} as \hat{y} . (It is more likely that no errors occurred than that so many errors occurred that \hat{c} was transformed into a different codeword.)

$$2) H\hat{y} \text{ is a (nonzero) scalar multiple of one of the columns of } H.$$

Note that since C is 1-error-correcting, no column of H is a scalar multiple of any other column of H . Therefore, if $H\hat{y}$ is a scalar multiple of a column of H , it is a scalar multiple of a *unique* column of H . Suppose $H\hat{y}$ is $a \cdot \hat{h}_i$, where \hat{h}_i is the i^{th} column of H and $a \in GF(q)$. Since $H\hat{y} = H\hat{e}$, it follows that (it is most likely that) \hat{e} is the string with a in the i^{th} place and 0's everywhere else. (Note that \hat{e} could equal the sum of this string with some codeword, but that is less likely.) Thus, since $\hat{c} = \hat{y} - \hat{e}$, we decode \hat{y} by subtracting a from the i^{th} element in \hat{y} .

$$3) H\hat{y} (= H\hat{e}) \text{ is not a scalar multiple of any column of } H.$$

In this case, \hat{e} has more than one nonzero digit, so more than one error must have occurred. We can then ask for a retransmission of the message. (Or we can decode the received string as one of the codewords that is nearest to it in Hamming distance, but in doing so there is a good chance that we may be decoding inaccurately.)

► (3.13) EXAMPLES

1) Suppose C is the (5, 2) binary code of Example (1.4) with parity check matrix

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

a) Suppose the string $\hat{y} = 11110$ is received. We compute

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and we conclude that \hat{y} is a codeword, so we decode the message as \hat{y} .

b) Suppose the string $\hat{y} = 01001$ is received. We compute

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

We compare this column matrix with the columns of H and we find that this is the third column of H . Therefore, we decode \hat{y} as $01001 - 00100 = 01101$. (Note that 01101 is indeed a codeword in C .)

c) Suppose the string $\hat{y} = 01010$ is received. We compute

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Looking at the matrix H we see that this column is not one of the columns of H . Therefore, we conclude that more than one error has occurred and we ask for a retransmission of the message.

2) Let C be the $(6, 3, 3)$ ternary code with parity check matrix

$$H = \begin{bmatrix} 1 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Suppose the string $\hat{y} = 011210$ is received. We compute

$$\begin{bmatrix} 1 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix},$$

which is $2 \cdot \hat{h}_4$, where \hat{h}_4 denotes the fourth column of H . Thus we decode \hat{y} as the codeword $011210 - 000200 = 011010$. (The reader should check that this is indeed a codeword in C .) ◀

EXERCISES

3.1. Suppose $\hat{x}, \hat{y}, \hat{z} \in GF(q)^n$. Show that Hamming distance satisfies the following three properties:

- (i) $d(\hat{x}, \hat{y}) = 0$ if and only if $\hat{x} = \hat{y}$.
- (ii) $d(\hat{x}, \hat{y}) = d(\hat{y}, \hat{x})$.
- (iii) $d(\hat{x}, \hat{y}) \leq d(\hat{x}, \hat{z}) + d(\hat{z}, \hat{y})$.

These three properties say that Hamming distance is a *metric* on $GF(q)^n$. (Hint for (iii): Think of changing the digits in one string to obtain another string.)

† 3.2. Let C be the binary linear code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

List all the codewords in C and find the weight of each codeword. What is $d(C)$?

3.3. Suppose \hat{x} is a binary string of weight k and \hat{y} is a binary string of weight l . Suppose that there are exactly m places in which \hat{x} and \hat{y} both have 1's. Show that $w(\hat{x} + \hat{y}) = k + l - 2m$. Conclude that the sum of two binary strings of even weight is of even weight, the sum of a binary string of even weight and a binary string of odd weight is of odd weight, and the sum of two binary strings of odd weight is of even weight.

† 3.4. Let C be a binary linear code. Prove that either all codewords have even weight or exactly half the codewords have even weight.

3.5. Let C be the ternary linear code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 2 & 2 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

List all the codewords in C and find the weight of each codeword. What is $d(C)$?

3.6. Construct a $(6, 3, 3)$ binary code.

† **3.7.** a) Construct a $(4, 2, 3)$ ternary code.

b) Is it possible to construct a $(4, 2, 3)$ binary code?

† **3.8.** Let C be the $(7, 4, 3)$ binary code with parity check matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

This code is called the $(7, 4, 3)$ binary Hamming code. (Hamming codes will be studied in §5.) Decode each of the following received strings using the decoding scheme in (3.12).

a) 1100110

b) 1000110

c) 0001011

† **3.9.** For the parity check matrix H in Exercise 3.8, explain why $H\hat{x}$ always must equal a scalar multiple of a column of H for every $\hat{x} \in GF(2)^7$.

† **3.10.** Let C be the $(5, 2, 3)$ ternary code with parity check matrix

$$H = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Use the decoding scheme in (3.12) to decode each of the following received strings or to ask for a retransmission.

a) 02211

b) 11101

c) 01112

d) 20210

e) 11111

3.11. (Singleton Bound) Suppose C is an (n, k, d) code. Show that $d - 1 \leq n - k$. (Hint: Use Theorem (3.10).)

3.12. (Gilbert-Varshamov Bound) Show that there is an (n, k) code with minimum distance at least d over $GF(q)$ if

$$\sum_{i=0}^{d-2} (q-1)^i \binom{n-1}{i} < q^{n-k}.$$

(Hint: One needs to show the existence of an $(n - k) \times n$ matrix H such that every $d - 1$ columns of H are linearly independent. Try to form such a matrix one column at a time. The left side of the above inequality gives the number of vectors that are a linear combination of $d - 2$ or fewer vectors from a given set of $n - 1$ vectors. The right side of the inequality is the number of vectors in $GF(q)^{n-k}$.)

4. Coset Decoding

In this section, we want to use the algebraic properties of a linear code to implement nearest neighbor decoding without having to compute the Hamming distance from a received string to every codeword. Suppose C is an (n, k) code over $GF(q)$. Suppose the codeword \hat{c} is transmitted and the string \hat{y} is received. Let $\hat{e} = \hat{y} - \hat{c}$ denote the error string. Then $\hat{y} - \hat{e} = \hat{c}$. Using nearest neighbor decoding, if we receive a string \hat{y} , then it is most likely that \hat{e} is a string of minimum weight such that $\hat{y} - \hat{e} \in C$. That is, we should consider the set

$$S_{\hat{y}} = \{\hat{e} : \hat{y} - \hat{e} \in C\}$$

and we should assume that the error string is a member of this set of minimum weight.

Since C is a subspace of $GF(q)^n$, it is, in particular, a subgroup of the additive group $(GF(q)^n, +)$. Thinking of C as a subgroup in this way, the set $S_{\hat{y}}$ above should look familiar. We have $\hat{e} \in S_{\hat{y}}$ if and only if $\hat{e} = \hat{y} + \hat{c}'$ for some $\hat{c}' \in C$. (Note that $\hat{c} \in C$ if and only if $-\hat{c} \in C$ also). Thus $S_{\hat{y}}$ is exactly the coset $\hat{y} + C$. Note that since $(GF(q)^n, +)$ is a commutative group, we have that C is a normal subgroup and so every left coset of C is also a right coset.

This suggests the following decoding scheme. We will list the cosets of C and take as a representative for each coset an element in that coset of minimum weight. Such an element is called a *coset leader*. Given a received string \hat{y} , we will find to which coset that string belongs, and then we will decode \hat{y} by subtracting the coset leader from \hat{y} . Such a listing is called a *standard array* or *decoding table* for C .

(4.1) ALGORITHM FOR CONSTRUCTING A STANDARD ARRAY. Let C be an (n, k) code over $GF(q)$. Suppose $k < n$.

Step 1. List the codewords of C , starting with $\hat{0}_n$, as the first row.

Step 2. Among the vectors in $GF(q)^n$ that are not codewords, choose one of minimum weight, call it \hat{x}_1 . List the coset $\hat{x}_1 + C$ as the second

row by entering $\hat{x}_1 + \hat{c}$ under the codeword \hat{c} . If $q = 2$ and $k = n - 1$, then all vectors in $GF(q)^n$ have now appeared, so stop; otherwise, set $r = 2$.

Step 3. Among the vectors that have not appeared in the first r rows, choose one of minimum weight, call it \hat{x}_r . List the coset $\hat{x}_r + C$ by entering $\hat{x}_r + \hat{c}$ under the codeword \hat{c} .

Step 4. Replace r by $r + 1$. If $r = q^{n-k}$, then all vectors have been used (see below), so stop; otherwise, go to Step 3.

The (n, k) code C has q^k codewords. Thus, from Lemma (III.4.5), each coset of C also has q^k elements. Since there are q^n elements in $GF(q)^n$, there will be q^{n-k} rows in our standard array. The vectors $\hat{0}_n, \hat{x}_1, \dots, \hat{x}_{q^{n-k}-1}$ are the coset leaders. Note that a standard array is not necessarily unique; indeed, at step 3 there may be more than one vector of minimum weight that has not appeared in the previous rows, so we may have to make an arbitrary choice of a coset leader.

► **(4.2) EXAMPLE.** Let C be the $(5, 2, 3)$ code of Example (1.4). A standard array for C is

	Coset Leaders			
	↓			
Codewords →	00000	10011	01101	11110
	00001	10010	01100	11111
	00010	10001	01111	11100
	00100	10111	01001	11010
	01000	11011	00101	10110
	10000	00011	11101	01110
	00110	10101	01011	11000
	10100	00111	11001	01010

Note that we had to make a choice of coset leader in the last two rows in this array. For example, in the last row we could just as well have chosen 01010 as our coset leader. (In that case, the last row would have been 01010, 11001, 00111, and 10100, which is a permutation of the last row in our table.) ◀

By the discussion preceding (4.1) and the way we construct a standard array, we decode a received string \hat{y} as follows:

Locate \hat{y} in the array and decode \hat{y} to be the codeword at the top of the column in which \hat{y} appears.

Note that the codeword at the top of the column in which \hat{y} appears is the difference of \hat{y} and the coset leader that begins the row in which \hat{y} appears. (Please remember that rows are horizontal and columns are vertical.)

- **(4.3) EXAMPLE.** We will use the code C of Example (1.4) and the standard array constructed in Example (4.2) to decode some messages.

Suppose the string 00101 is received. We locate this string in the fifth row and third column of the array in Example (4.2). The codeword that heads this column is 01101, so we decode 00101 as 01101. It is easy to see that 01101 is in fact the codeword that is nearest to 00101; indeed, these two strings differ in only one place and the next closest codeword, 00000, differs in two places from 00101.

Now suppose the string 00111 is received. We locate this string in the last row and second column of the array in (4.2). Therefore, we will decode this string as the codeword 10011. The distance between these two strings is 2. Note that if we had made a different choice in forming our array and had taken 01010 as the coset leader for the last row, then 00111 would have appeared in the third column, and we would have decoded it as 01101, which is another codeword that differs from 00111 in two places. Thus, while we have decoded the received string as a nearest neighbor, we can not be very confident that we have accurately decoded the message. The problem here is that our code C is just a 1-error-correcting code, not a 2-error-correcting code, and two errors (most likely) occurred to transform a codeword into 00111. Because of the ambiguity involved in decoding a string that appears in one of the last two rows of this array, one may want to ask for a retransmission whenever the received string is in one of the last two rows. (This scheme would be called "incomplete decoding.") In fact, if we compare this method with the decoding scheme in Example (3.12), then we see that the first possibility in (3.12) corresponds to the received string being in the first row of our standard array, the second possibility in (3.12) corresponds to the received string being in one of the second through sixth rows in our array, and the last possibility in (3.12) corresponds to the received string being in one of the last two rows in our array. ◀

In implementing coset decoding on a computer, one does not need to store a complete standard array and search through the array each time to locate a string. The key is once again to compute $H\hat{y}$, where H is a parity check matrix for our code.

(4.4) DEFINITION. Suppose C is an (n, k) code over $GF(q)$ and H is a parity check matrix. If $\hat{y} \in GF(q)^n$, then we call $H\hat{y}$ the *syndrome* of \hat{y} (with respect to H).

In medical terminology, a “syndrome” is a number of symptoms that characterize a disease. In coding theory, a syndrome characterizes which errors have occurred in transmission.

(4.5) LEMMA. Two strings have the same syndrome if and only if they belong to the same coset of C .

PROOF. Suppose $\hat{y}_1, \hat{y}_2 \in GF(q)^n$. Then

$$\begin{aligned} H\hat{y}_1 = H\hat{y}_2 &\Leftrightarrow H(\hat{y}_1 - \hat{y}_2) = \hat{0}_{n-k} \\ &\Leftrightarrow \hat{y}_1 - \hat{y}_2 \in C. \end{aligned}$$

And $\hat{y}_1 - \hat{y}_2 \in C$ exactly means that \hat{y}_1 and \hat{y}_2 are equivalent mod C (viewing C as a subgroup of $GF(q)^n$), which means that they lie in the same coset. ■

Instead of storing a complete standard array as above, we just need to store the coset leaders and their syndromes. We then decode according to the following scheme.

(4.6) SYNDROME DECODING SCHEME

Step 1. For a received string \hat{y} , compute the syndrome $H\hat{y}$.

Step 2. Determine the coset leader \hat{x} that has the same syndrome as \hat{y} .

Step 3. Decode \hat{y} as $\hat{y} - \hat{x}$.

Notice that there are q^n entries in a standard array, but there are only q^{n-k} coset leaders. So in a table of coset leaders and syndromes, we will only have $2q^{n-k}$ entries, significantly fewer entries to store than in a standard array. However, for large codes even $2q^{n-k}$ may be too big to make this method truly efficient and more specialized decoding schemes may be required.

Given an (n, k) code C over $GF(q)$ with parity check matrix H , we can determine a syndrome table without first computing an entire standard array by proceeding as follows.

(4.7) ALGORITHM FOR CONSTRUCTING A SYNDROME TABLE

Step 0. We know that the syndrome of $\hat{0}_n$ is $\hat{0}_{n-k}$.

Step 1. Compute the syndrome of each weight 1 vector in $GF(q)^n$. Each time a vector has a syndrome that has not previously occurred, add

that vector to the table as a coset leader. If q^{n-k} distinct syndromes have been found, then stop; otherwise, set $w = 2$.

Step 3. Compute the syndrome of each vector of weight w . Each time a vector has a syndrome that has not previously occurred, add that vector to the table as coset leader.

Step 4. If q^{n-k} distinct syndromes have been found, then stop; otherwise, replace w by $w + 1$ and go to Step 3.

► (4.8) EXAMPLES

1) Using the coset leaders in Example (4.2) and the parity check matrix H from Example (2.12), we compute syndromes and determine the following syndrome table for the $(5, 2, 3)$ binary code C of Example (1.4).

Coset Leader	Syndrome
00000	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
00001	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
00010	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$
00100	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$
01000	$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$
10000	$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$
00110	$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$
10100	$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Notice that if the coset leader consists of just 1's and 0's (as it must in a binary code), then the associated syndrome is simply the sum of the columns of H corresponding to the 1's in the coset leader. For example, if the coset leader has 1's in the third and fourth positions and 0's elsewhere, then the syndrome of this coset leader is the sum of the third and fourth columns of H . Let's use this syndrome table to decode a string. Suppose the received string is 11000. We compute

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

The coset leader corresponding to this syndrome is 00110. Therefore, we decode the message as the codeword $11000 - 00110 = 11110$.

2) Let C be the $(4, 2)$ ternary code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 2 \end{bmatrix}.$$

By Proposition (2.16), a parity check matrix for C is

$$H = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

Note that no column of H is a scalar multiple of another column, but the second column is the sum of the third and fourth. Therefore, the minimum weight of C is 3 by Theorem (3.10) and C is a 1-error-correcting code by Theorem (3.8). We will compute a syndrome table for C by using Algorithm (4.7). We find, upon computing the syndrome of each weight 1 vector in $GF(q)^4$, that each of these vectors has a different syndrome, and we obtain the following table:

Coset Leader	Syndrome
0000	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
0001	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
0002	$\begin{bmatrix} 0 \\ 2 \end{bmatrix}$
0010	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
0020	$\begin{bmatrix} 2 \\ 0 \end{bmatrix}$
0100	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
0200	$\begin{bmatrix} 2 \\ 2 \end{bmatrix}$
1000	$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$
2000	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$

The algorithm stops at this point, since we have found $3^{4-2} = 9$ distinct syndromes (and this is the number of distinct cosets of C in $GF(3)^4$). We note that we should not expect the algorithm always to stop so soon. We should expect that there may be duplication among the syndromes of weight 1 vectors and that usually we will have to compute syndromes of vectors of weight greater than 1. As we will see in the next section, the code C here is an example of a perfect code, and those are quite rare.

Let's use the syndrome table to decode a message. Suppose the codeword 1101 ($=1012+0122$) is transmitted, but due to noise the receiver re-

ceives the string 1121. The receiver then computes

$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

The receiver then looks up this syndrome in the table and finds that the corresponding coset leader is 0020. The receiver then decodes the message as $1121 - 0020 = 1101$, thus correcting the single error.

Suppose two errors had occurred in the transmission of 1101, and the receiver had received 1200. The receiver would then compute

$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

This syndrome corresponds to the coset leader 0010. Therefore, the receiver would decode the message as $1200 - 0010 = 1220$. This is a codeword since $1220 = 2 \cdot 1012 + 0122$, but of course it is not the codeword that was transmitted. We should not expect a 1-error-correcting code always to decode accurately a received string with two errors in it. ◀

EXERCISES

4.1. Let $C = \{000, 111\} \subset GF(2)^3$.

- Is this a 1-error-detecting code? Is this a 1-error-correcting code?
- Construct a standard array for C .
- Use your array to decode the received strings $\hat{y}_1 = 010$ and $\hat{y}_2 = 101$.

† 4.2. Suppose C is the binary $(5,2,3)$ code of Example (1.4). Use the standard array in Example (4.2) for the following decoding problems.

- Suppose the codeword 10011 is transmitted and two errors occur so that the received string is 10101. Will this string be decoded accurately?
- Suppose the codeword 10011 is transmitted and two errors again occur so that the received string is now 01011. Will this string be decoded accurately?

† 4.3. Let C be the (4,2) ternary code of Example (4.8.2). Use the syndrome table in that example for the following decoding problems.

- Suppose the codeword 0122 is transmitted and one error occurs so that the received string is 2122. Will this string be decoded accurately?
- Suppose the codeword 0122 is transmitted and two errors occur so that the received string is 0012. Will this string be decoded accurately?
- Suppose the codeword 0122 is transmitted and two errors occur so that the received string is 0221. Will this string be decoded accurately?

4.4. Construct a standard array for the (5,2) binary code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Use your array to decode the received strings $\hat{y}_1 = 11101$ and $\hat{y}_2 = 01001$.

4.5. Let C be the (4,2) binary code with parity check matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

- Is this a 1-error-correcting code? Is this a 1-error-detecting code?
- Construct a syndrome table for C .
- Use your table to decode the received strings $\hat{y}_1 = 1011$ and $\hat{y}_2 = 0011$.

4.6. Let C be the (4,2) binary code with parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

Construct a syndrome table for C and use your table to decode the received strings $\hat{y}_1 = 0100$ and $\hat{y}_2 = 1011$.

† 4.7. Let C be the (5,3) ternary code with parity check matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 2 & 2 & 1 & 0 & 1 \end{bmatrix}.$$

Construct a syndrome table for C and use your table to decode the received strings $\hat{y}_1 = 11202$, $\hat{y}_2 = 20112$, and $\hat{y}_3 = 20201$.

- 4.8. Suppose C is a linear t -error-correcting code. Prove that if a coset of C contains a vector \hat{x} of weight at most t , then every other vector in that coset has weight greater than t .

5. Hamming Codes

An interesting code was presented in Exercise 3.8. It was a $(7,4,3)$ binary code C called a Hamming code with parity check matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Notice that the columns of H are the binary representations of the integers from 1 to 7. Thus the columns of H are all the nonzero vectors in $GF(2)^3$. This means that if $\hat{y} \in GF(2)^7$, then the syndrome $H\hat{y}$ is either $\hat{0}_3$ or one of the columns of H . This implies that every string $\hat{y} \in GF(2)^7$ is either a codeword or of the form $\hat{c} + \hat{e}$ where \hat{e} is a string of weight 1. Thus every string that is not a codeword is exactly one unit in Hamming distance from a codeword.

Notice that this code is a 1-error-correcting code and that it also has the property that if more than one error occurs in the transmission of a codeword, then the received string will certainly be decoded inaccurately (using nearest neighbor decoding). This is because if more than one error occurs in the transmission of a codeword \hat{c} , then the received string will now be closer to a different codeword \hat{c}' (since every string is one unit away from some codeword) than it is to \hat{c} . Thus we will decode the received string inaccurately as \hat{c}' .

There is something "optimal" about the fact that the columns of the parity check matrix of this code comprise exactly all the nonzero vectors in $GF(2)^3$. Can we create a whole family of binary codes with this property? Given an integer $r \geq 2$, let's form the matrix H_r whose columns consist precisely of the binary representations of the integers from 1 to $2^r - 1$, with the i^{th} column of H_r being the binary representation of the integer i . The matrix H_r then has r rows and $2^r - 1$ columns, and obviously no column is a scalar multiple of any other column. Note that a column with exactly two 1's is the sum of two other columns of H_r , each having exactly one 1.

Hence there are sets of three columns in H_r that are linearly dependent sets. It follows from Theorem (3.10) that H_r is a parity check matrix for a 1-error-correcting code of minimum distance 3.

(5.1) DEFINITION. The code with parity check matrix H_r will be denoted $\text{Ham}(r,2)$ and is called a *binary Hamming code*.

Notice that since H_r is a parity check matrix, the code $\text{Ham}(r,2)$ is a $(2^r - 1, 2^r - 1 - r, 3)$ code. The $(7,4,3)$ code of Exercise 3.8 could now be referred to as $\text{Ham}(3,2)$. As long as a parity check matrix for a code has the binary representations of the nonzero integers from 1 to $2^r - 1$ as its columns, we will refer to it as a Hamming code, even if the i^{th} column of the parity check matrix is not the binary representation of the integer i ; i.e., the columns may be permuted from the way they appear in H_r . Two such codes are equivalent according to Definition (2.6). Later in this section, we will consider Hamming codes over $GF(q)$ for $q > 2$.

The decoding scheme for 1-error-correcting codes in (3.12) is particularly simple for $\text{Ham}(r,2)$. Given a received string \hat{y} , we compute the syndrome $H_r \hat{y}$. If this syndrome is $\hat{0}_r$, then \hat{y} is a codeword and we decode \hat{y} as itself. If this syndrome is not zero, then it must be one of the columns, say the i^{th} column, of H_r . Notice that if the syndrome is the binary representation of the integer i , then we know it is the i^{th} column of H_r without having to search through the columns of H_r . We then decode \hat{y} by subtracting (or adding since our operations are mod 2) 1 from the i^{th} place in \hat{y} . The third possibility in (3.12), namely that the syndrome is not one of the columns of our parity check matrix, cannot occur here, since the columns of H_r consist of all nonzero binary r -tuples.

► **(5.2) EXAMPLE.** $\text{Ham}(4,2)$ is a binary $(15,11,3)$ code with parity check matrix

$$H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

For an example of decoding, suppose the received string is

$$\hat{y} = 101100110101110.$$

We compute $H_4\hat{y}$ and find it to be

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

This is the binary representation of the integer 12. Thus we subtract 1 from the twelfth digit of the received string and decode \hat{y} as

$$101100110100110. \blacktriangleleft$$

Now let's construct Hamming codes over an arbitrary finite field $GF(q)$. What should we mean by the generalization of the binary Hamming codes to an arbitrary finite field? One way would be to consider for each r the matrix whose columns consist of all the nonzero strings in $GF(q)^r$. But this code would have minimum distance 1 if $q > 2$, for if $a \in GF(q)$ with $a \neq 0, 1$, then the string with a in the i^{th} place and 0's elsewhere would be a scalar multiple of the string with 1 in the i^{th} place and 0's elsewhere. This would not be a very useful code. What we would like is that our Hamming code should be a 1-error-correcting code such that every possible syndrome is a scalar multiple of one of the columns of the parity check matrix.

Define an equivalence relation on $GF(q)^r$ by calling two vectors \hat{x} and \hat{y} equivalent if there exists a nonzero scalar $a \in GF(q)$ such that $\hat{x} = a\hat{y}$. If $\hat{x} \neq \hat{0}_r$, then there will be $q - 1$ vectors equivalent to \hat{x} . Therefore, besides the equivalence class $\{\hat{0}_r\}$, there will be $(q^r - 1)/(q - 1)$ equivalence classes containing nonzero vectors.

What we want to do now is to form a parity check matrix by choosing one representative from each of these $(q^r - 1)/(q - 1)$ equivalence classes as our columns. The code defined by such a parity check matrix is called a *q-ary Hamming code*. Such a parity check matrix will be an $r \times (q^r - 1)/(q - 1)$ matrix such that no column is a scalar multiple of another column. It is easy to see that there will be sets of three columns that will be linearly dependent sets. Thus this parity check matrix will define a code with minimum distance 3.

In the case that our field is $GF(p)$, where p is a prime, we can even give a "canonical" way to choose such a matrix. Let $H_{r,p}$ have as its columns all the nonzero vectors in $GF(p)^r$ that have first nonzero entry equal to 1 and order these columns in increasing order according to the integer that each represents. This is much easier to understand with a couple of examples.

► (5.3) EXAMPLES

1) With $r = 2$ and $q = 3$, the nonzero vectors in $GF(3)^2$ are partitioned into the following equivalence classes:

$$\{01, 02\}, \{10, 20\}, \{11, 22\}, \text{ and } \{12, 21\}.$$

The matrix $H_{2,3}$ is then

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix},$$

2) The matrix $H_{3,3}$ is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{bmatrix}. \blacktriangleleft$$

(5.4) **DEFINITION.** The code with parity check matrix $H_{r,p}$ will be denoted $\text{Ham}(r, p)$.

In summary, a q -ary Hamming code is a 1-error-correcting code and has the following “optimal” property: every possible syndrome is a scalar multiple of a column of the parity check matrix, so that the third possibility in the decoding scheme in (3.12) never occurs. Thus, using that decoding scheme, we would never ask for a retransmission. This does not mean that we necessarily accurately decode every message. If more than one error occurs in transmission, then we will definitely decode the received string incorrectly; but, again, it is more likely that one error occurs than that more than one error occurs.

We will now look at the “optimal” property of a Hamming code from another point of view.

(5.5) **DEFINITION.** For $\hat{x} \in GF(q)^n$ and $r \in \mathbf{P}$, put

$$S_r(\hat{x}) = \{\hat{y} \in GF(q)^n : d(\hat{x}, \hat{y}) \leq r\}.$$

We call $S_r(\hat{x})$ the *sphere of radius r centered at \hat{x}* .

Thus, $S_r(\hat{x})$ contains all strings that differ from \hat{x} in at most r places. Of course, this “sphere” has only a finite number of points. Let’s count exactly how many points are in such a sphere.

(5.6) LEMMA. A sphere of radius r in $GF(q)^n$ contains exactly

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r$$

vectors. (Recall that $\binom{n}{s}$ is the binomial coefficient $n!/s!(n-s)!.$)

PROOF. Consider the vectors that differ from \hat{x} in exactly s places, where $s \leq n$. There are $\binom{n}{s}$ ways to choose s places from n places and in each of these s places, there are $q-1$ ways to choose an element in $GF(q)$ different from the corresponding entry in \hat{x} . Therefore, the number of vectors that are at a Hamming distance s from \hat{x} is $\binom{n}{s}(q-1)^s$. The lemma follows since $S_r(\hat{x})$ consists of vectors that differ from \hat{x} in at most r places. ■

Now suppose C is an $(n, k, 2t+1)$ code over $GF(q)$ (so C is a t -error-correcting code). The spheres of radius t around each codeword are then disjoint; indeed, if there were two codewords \hat{c}_1 and \hat{c}_2 and a vector \hat{y} such that $d(\hat{c}_1, \hat{y}) \leq t$ and $d(\hat{c}_2, \hat{y}) \leq t$, then we would have $d(\hat{c}_1, \hat{c}_2) \leq t+t = 2t$ by Exercise 3.1. But this would contradict the fact that the minimum distance of C is $2t+1$. Now, since these spheres of radius t around each of the q^k codewords are disjoint and since there are a total of q^n vectors in $GF(q)^n$, by applying Lemma (5.6) we obtain the following result.

(5.7) THEOREM. (The Sphere-packing or Hamming Bound) If C is an $(n, k, 2t+1)$ code over $GF(q)$, then

$$(5.8) \quad q^k \left[\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t \right] \leq q^n.$$

(5.9) DEFINITION. An $(n, k, 2t+1)$ code over $GF(q)$ is called *perfect* if equality holds in (5.8).

Thus, a t -error-correcting code of dimension k is perfect if the spheres of radius t about the codewords completely fill $GF(q)^n$. Since these spheres are disjoint and fill $GF(q)^n$, the number of these spheres (namely q^k) is the maximum number of disjoint spheres of radius t that may be "packed" into $GF(q)^n$.

The perfect code consisting of the whole space $GF(q)^n$ (i.e., the $(n, n, 1)$ code) or the perfect binary $(n, 1, n)$ codes generated by the vector with all entries equal to 1 when n is odd (since $n = 2t+1$) are called *trivial* perfect codes. Note that there are only certain values of q, n, k , and t for which we could possibly have nontrivial perfect $(n, k, 2t+1)$ codes. For example,

suppose $q = 2$ and $n = 6$. Note that t then could be either 1 or 2, since we must have $2t + 1 \leq 6$. If $t = 1$, then we have

$$\binom{6}{0} + \binom{6}{1} = 7,$$

which is not a power of 2. If $t = 2$, then we have

$$\binom{6}{0} + \binom{6}{1} + \binom{6}{2} = 1 + 6 + 15 = 22,$$

which is also not a power of 2. Hence there cannot be a nontrivial perfect binary code of length 6 since we could not have equality in (5.8) for $t = 1$ or 2 and any value of k .

Our feeling that Hamming codes are in some sense "optimal" is reflected in the following result.

(5.10) PROPOSITION. Hamming codes are perfect.

PROOF. A q -ary Hamming code has parameters $((q^r - 1)/(q - 1), n - r, 3)$. Substituting into the left side of (5.8) (note that $t = 1$), we obtain

$$q^{n-r}(1 + (q - 1)[(q^r - 1)/(q - 1)]) = q^{n-r}(1 + (q^r - 1)) = q^{n-r}q^r = q^n.$$

Therefore, we have equality in (5.8) and Hamming codes are perfect. ■

The amazing thing is that there are only two other nontrivial perfect (linear) codes. They are a binary (23,12,7) code and a ternary (11,6,5) code, both discovered by M. Golay in 1949. The fact that these are the only other nontrivial perfect codes was established by A. Tietäväinen in 1973 following much work on this problem by J. H. van Lint. Note that there are other values of q, n , and t that give equality in (5.8), but there do not exist codes with these parameters. For example, Golay showed that there is no binary (90,78,5) code even though (5.8) becomes an equality if we substitute $q = 2, n = 90, k = 78$, and $t = 2$.

EXERCISES

- 5.1. Assume that the code Ham(3,2) is being used. Decode each of the following received strings.
- 0111100
 - 1101011

- 5.2. Find a generator matrix in standard form for a binary Hamming (7,4,3) code.
- † 5.3. Assume the code $\text{Ham}(2,3)$ given by the parity check matrix $H_{2,3}$ in Example (5.3.1) is being used. Decode each of the following received strings.
- 2021
 - 1211
 - 2010
- 5.4. Find a parity check matrix for another Hamming code that is equivalent to the code in Example (5.3.1).
- † 5.5. Find the matrix $H_{2,5}$.
- † 5.6. Assume that the code $\text{Ham}(4,2)$ of Example (5.2) is being used. Decode the following received strings.
- 11100000000000
 - 000111000111000
- 5.7. Show that $\text{Ham}(2,2)$ is a trivial perfect code.
- † 5.8. a) Does there exist a nontrivial binary perfect code of length 8?
b) Does there exist a nontrivial perfect code of length 8 over $GF(7)$?
- 5.9. a) Determine the nine codewords in the code $\text{Ham}(2,3)$.
b) Note that since the minimum distance of this code is 3, no two codewords have the same first two digits. For $i, j \in \{0, 1, 2\}$, let a_{ij} (respectively, b_{ij}) denote the third (respectively, fourth) digit in the codeword with first digit i and second digit j . Form the two 3×3 matrices $A = [a_{ij}]$ and $B = [b_{ij}]$. Show that A and B are orthogonal Latin squares (cf. IV.9). For further connections between codes and Latin squares, see Hill [17].
- 5.10. Let C be the binary code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

- a) Show that C is equivalent to $\text{Ham}(3,2)$.

- b) Find all the codewords in C that have weight 3. (There are seven of them.)
- c) Let S denote the set of codewords of weight 3 that you found in (b). For this problem, we will index a vector in $GF(2)^7$ beginning with the subscript 0; i.e., $x_0x_1x_2x_3x_4x_5x_6$ will denote a vector in $GF(2)^7$. Let T denote the set of three-element subsets of the set $\{0, 1, 2, 3, 4, 5, 6\}$. Define a mapping Φ from S to T by

$$i \in \Phi(x_0x_1x_2x_3x_4x_5x_6) \Leftrightarrow x_i = 1.$$

For example, $\Phi(1101000) = \{0, 1, 3\}$. Show that the image of Φ is the Steiner triple system in Example (IV.9.15). (For more connections between perfect codes and combinatorial designs, see Pless [26].)

6. Polynomial and Cyclic Codes

We have seen another vector space of dimension n over $GF(q)$ besides $GF(q)^n$. Put $F = GF(q)$. Suppose $f(X) \in F[X]$ is a polynomial of degree n and put $R = F[X]/(f(X))$. In Proposition (IV.7.6) we showed that R is a vector space of dimension n over F with basis $1, \alpha, \alpha^2, \dots, \alpha^{n-1}$, where α denotes the image of X in R .

Recall from linear algebra that if V and W are two vector spaces of dimension n over F , then V is isomorphic to W , which means that there is a linear transformation $T: V \rightarrow W$ that is 1-1 and onto. Indeed, there are many such isomorphisms. We get one by fixing a basis v_1, v_2, \dots, v_n of V and a basis w_1, w_2, \dots, w_n of W and defining T by $T(a_1v_1 + \dots + a_nv_n) = a_1w_1 + \dots + a_nw_n$. It will help us to set up an isomorphism between $GF(q)^n$ and R if we denote strings in $GF(q)^n$ beginning with the subscript 0; i.e., in this section, an arbitrary element of $GF(q)^n$ will be denoted $a_0a_1 \dots a_{n-1}$. We will then identify elements in R with strings in $GF(q)^n$ via the vector space isomorphism

$$(6.1) \quad \begin{aligned} GF(q)^n &\longrightarrow R \\ a_0a_1 \dots a_{n-1} &\longmapsto a_0 + a_1\alpha + \dots + a_{n-1}\alpha^{n-1}. \end{aligned}$$

Under this correspondence, an (n, k) code over $GF(q)$ can be identified with a k -dimensional vector subspace of R . A vector subspace of R is a subgroup of the additive group of R that is also closed under multiplication by elements of F . Recall that an ideal of R is a subgroup of the additive

group of R that is closed under multiplication by elements of R . Thus an ideal of R is a special kind of vector subspace of R (since it is closed not only under multiplication by elements of F , but under multiplication by elements of R as well). We might expect that the codes that correspond to ideals of R will have some special property.

In fact, this will be true if we take $f(X) = X^n - 1$. Put

$$R_n = F[X]/(X^n - 1).$$

We will show below that ideals of R_n correspond to cyclic codes, which we now define.

(6.2) DEFINITION. A linear code C of length n is called *cyclic* if whenever $a_0a_1 \cdots a_{n-1}$ is a codeword, then so is $a_{n-1}a_0a_1 \cdots a_{n-2}$.

The string $a_{n-1}a_0a_1 \cdots a_{n-2}$ is called a *cyclic shift* of $a_0a_1 \cdots a_{n-1}$.

There is another equivalent way to give this definition. We can define an action of S_n , the n^{th} symmetric group, on $GF(q)^n$ as follows. View S_n as being the group of permutations of the set $\{0, 1, \dots, n-1\}$. Then if $\mu \in S_n$, define

$$\mu(a_0a_1 \cdots a_{n-1}) = a_{\mu(0)}a_{\mu(1)} \cdots a_{\mu(n-1)}.$$

Let $\sigma \in S_n$ denote the cycle $(n-1, n-2, \dots, 1, 0)$. Then C is a cyclic code if $\sigma(\hat{c}) \in C$ for all $\hat{c} \in C$. Note that if C is cyclic, then we may perform *repeated* cyclic shifts on a codeword and always obtain another codeword; i.e., $\sigma^i(\hat{c}) \in C$ for $i = 1, 2, \dots, n-1$.

(6.3) LEMMA. Suppose $\hat{x}, \hat{y} \in GF(q)^n$, $a, b \in F$, and $\mu \in S_n$. Then $\mu(a\hat{x} + b\hat{y}) = a\mu(\hat{x}) + b\mu(\hat{y})$.

PROOF. Exercise 6.1.

(6.4) LEMMA. Suppose C is the (n, k) code generated by $\hat{c}_1, \hat{c}_2, \dots, \hat{c}_k$. If $\sigma(\hat{c}_i) \in C$ for each $i = 1, 2, \dots, k$, then C is cyclic.

PROOF. If \hat{c} is any codeword in C , then

$$\hat{c} = b_1\hat{c}_1 + b_2\hat{c}_2 + \cdots + b_k\hat{c}_k$$

for some $b_1, b_2, \dots, b_k \in F$. But then by applying Lemma (6.3) one sees that

$$\sigma(\hat{c}) = b_1\sigma(\hat{c}_1) + b_2\sigma(\hat{c}_2) + \cdots + b_k\sigma(\hat{c}_k), \quad 262$$

which is also in C by our hypothesis. ■

► (6.5) EXAMPLES

1) The $(n, 1, n)$ code generated by the string consisting of n 1's is a cyclic code.

2) The $(3, 2)$ binary code $C = \{000, 110, 101, 011\}$ is cyclic.

3) The $(7, 4, 3)$ binary code C of Exercise 5.10 is a cyclic code. Indeed, the generators of that code were $\hat{c}_1 = 1010001$, $\hat{c}_2 = 1101000$, $\hat{c}_3 = 0011010$, and $\hat{c}_4 = 0110100$. Hence, if σ denotes the cycle $(6, 5, 4, 3, 2, 1, 0)$, then $\sigma(\hat{c}_1) = \hat{c}_2$, $\sigma(\hat{c}_2) = \hat{c}_4$, $\sigma(\hat{c}_3) = \hat{c}_1 + \hat{c}_2 + \hat{c}_4$, and $\sigma(\hat{c}_4) = \hat{c}_3$. Thus C is cyclic by Lemma (6.4). Since C was equivalent to the $(7, 4, 3)$ binary Hamming code, it follows that this Hamming code is equivalent to a cyclic code. We will show below that every binary Hamming code is equivalent to a cyclic code. In fact, many of the most important codes are cyclic or equivalent to cyclic codes. ◀

We now show that cyclic codes correspond to ideals in the ring R_n (under the isomorphism in (6.1)).

(6.6) PROPOSITION. A linear code of length n is cyclic if and only if the corresponding subspace of R_n is an ideal.

PROOF. Let C be a linear code of length n and let I denote the vector subspace of R_n corresponding to C under the isomorphism (6.1). The key point is that if $\hat{c} = c_0 \cdots c_{n-1} \in C$ corresponds to $c(\alpha) = c_0 + c_1\alpha + \cdots + c_{n-1}\alpha^{n-1} \in I$, then $\sigma^i(\hat{c})$ corresponds to $\alpha^i c(\alpha)$, since $\alpha^n = 1$. For example, if $n = 4$ and if $1011 \in C$, then $\sigma^2(1011) = 1110$, while

$$\alpha^2(1 + \alpha^2 + \alpha^3) = \alpha^2 + \alpha^4 + \alpha^5 = 1 + \alpha + \alpha^2,$$

since $\alpha^4 = 1$.

Therefore, C is cyclic if and only if $\alpha^i c(\alpha) \in I$ for all $c(\alpha) \in I$ and for $i = 1, 2, \dots, n-1$. But this is equivalent to the subspace I being closed under multiplication by elements in R , which says exactly that I is an ideal. ■

Thus we can identify cyclic codes of length n with ideals of R_n . There is one more identification we need to make. Let P_{n-1} denote the vector space of polynomials in $F[X]$ of degree at most $n-1$. Then that there is a vector space isomorphism

$$(6.7) \quad \begin{array}{ccc} R_n & \longrightarrow & P_{n-1} \\ a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} & \longmapsto & a_0 + a_1X + \cdots + a_{n-1}X^{n-1} \end{array}$$

so we can identify elements of R_n with polynomials in $F[X]$ of degree at most $n-1$. Therefore, we can think of the codewords belonging to a cyclic code of length n as polynomials in $F[X]$ of degree at most $n-1$.

(6.8) PROPOSITION. There is a 1-1 correspondence between cyclic codes of length n and ideals $(g(X))$ of $F[X]$ such that $g(X)$ divides $X^n - 1$.

PROOF. We know that there is a 1-1 correspondence between cyclic codes of length n and ideals of R_n . From Exercise IV.3.11 we know that the ideals of R_n are in 1-1 correspondence with the ideals in $F[X]$ that contain $(X^n - 1)$. From the proof of Corollary (IV.6.15) we know that every ideal in $F[X]$ is a principal ideal that, if nonzero, is generated by the unique monic polynomial of least degree in the ideal. If an ideal $(g(X))$ of $F[X]$ contains $(X^n - 1)$, then $X^n - 1$ must be a multiple of $g(X)$; in other words, $g(X)$ must divide $X^n - 1$. ■

To summarize, given any nonzero cyclic code C of length n , if we think of the codewords as polynomials in $F[X]$ of degree at most $n - 1$, then there is a unique monic polynomial $g(X)$ of least degree in C and $g(X)$ divides $X^n - 1$. If we think of C as an ideal in R_n , then that ideal is the principal ideal generated by $g(\alpha)$. (Other elements in this ideal may also generate it.) Conversely, given a monic polynomial $g(X) \in F[X]$ that divides $X^n - 1$, then we get a cyclic code of length n by looking at all polynomials of degree at most $n - 1$ that are in the principal ideal generated by $g(X)$ in $F[X]$ (equivalently, by letting C be the code that corresponds to the principal ideal $(g(\alpha))$ in R_n).

(6.9) DEFINITION. If C is a nonzero cyclic code of length n , then the unique monic polynomial of least degree in C is called the *generator polynomial* of C .

► (6.10) EXAMPLES

1) If C is the $(n, 1, n)$ code generated by the string with n 1's, then the generator polynomial of C is $1 + X + X^2 + \cdots + X^{n-1}$. Notice that

$$X^n - 1 = (X - 1)(X^{n-1} + X^{n-2} + \cdots + X + 1).$$

2) If $C = \{000, 110, 101, 011\}$, then the generator polynomial of C is $1 + X$. Note that $X^3 - 1 = (X + 1)(X^2 + X + 1)$, where we use the fact that $1 = -1$ here.

3) These two examples almost give us all binary cyclic codes of length 3. To find all cyclic codes of length n we must factor $X^n - 1$ over F into a product of irreducible monic polynomials. If $X^n - 1 = p_1(X)^{r_1} \cdots p_s(X)^{r_s}$, where the $p_i(X)$ are monic irreducible polynomials, then all monic divisors of $X^n - 1$ are of the form $p_1(X)^{t_1} \cdots p_s(X)^{t_s}$, where $0 \leq t_i \leq r_i$ for $i = 1, 2, \dots, s$. When $n = 3$ and $F = GF(2)$, we have $X^3 - 1 = (X - 1)(X^2 + X + 1)$ for the factorization of $X^3 - 1$ into a product of irreducible monic

polynomials. Thus we have four monic divisors of $X^3 - 1$; namely 1, $X + 1$, $X^2 + X + 1$, and $X^3 - 1$. The cyclic code corresponding to 1 is the whole space $GF(2)^3$, the cyclic code corresponding to $X + 1$ is the code in Example (6.10.2) above, and the code corresponding to $X^2 + X + 1$ is the code $\{000, 111\}$, which is Example (6.10.1) with $n = 3$. The cyclic code corresponding to $X^3 - 1$ is the zero code $\{000\}$. So we have shown that there are four cyclic binary codes of length 3. The general problem of factoring $X^n - 1$ over $GF(q)$ is considered in [23]. ◀

Once we know the generator polynomial of a cyclic code, we can write down a generator matrix by using the next result.

(6.11) PROPOSITION. Suppose C is a cyclic code of length n with generator polynomial

$$g(X) = a_0 + a_1X + \cdots + X^r$$

of degree $r < n$. Then the dimension of C is $n - r$ and a generator matrix for C is

$$G = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & a_0 & a_1 & a_2 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 0 & a_0 & a_1 & a_2 & \cdots & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & a_0 & a_1 & a_2 & \cdots & 1 \end{bmatrix},$$

PROOF. The codewords in C are exactly the polynomials of degree at most $n - 1$ in $F[X]$ that are of the form $f(X)g(X)$ for some $f(X) \in F[X]$. Then we must have $f(X) = b_0 + b_1X + \cdots + b_{n-r-1}X^{n-r-1}$ for some $b_0, b_1, \dots, b_{n-r-1} \in F$, since $g(X)$ has degree r . The $n - r$ rows of the matrix G correspond to the codewords

$$g(X), Xg(X), X^2g(X), \dots, X^{n-r-1}g(X).$$

These codewords generate C because

$$f(X)g(X) = b_0g(X) + b_1Xg(X) + \cdots + b_{n-r-1}X^{n-r-1}g(X).$$

It is easy to see that G has rank $n - r$ because the i^{th} row of G has a 1 in the $(r + i)^{\text{th}}$ column as its last nonzero entry. ■

- **(6.12) EXAMPLE.** Suppose $F = GF(3)$. Then the factorization of $X^4 - 1$ into monic irreducible factors is

$$X^4 - 1 = (X - 1)(X + 1)(X^2 + 1).$$

Let C be the cyclic code of length 4 over $GF(3)$ with generator polynomial $X - 1 = 2 + X$. Then C has dimension 3 and a generator matrix for C is

$$G = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{bmatrix}.$$

Next we will see how to determine a parity check matrix for a cyclic code. Suppose C is an (n, k) cyclic code with generator polynomial $g(X)$ of degree $n - k$. We know that $g(X) | (X^n - 1)$, so there exists $h(X)$ of degree k such that $X^n - 1 = g(X)h(X)$.

(6.13) DEFINITION. The polynomial $h(X)$ is called the *check polynomial* of C .

Note that in the quotient ring R_n , we have $g(\alpha)h(\alpha) = 0$. In fact, we have the following characterization of codewords.

(6.14) PROPOSITION. With notation as above, if we identify codewords in C with elements in R_n , then $y(\alpha) \in R_n$ is a codeword in C if and only if $y(\alpha)h(\alpha) = 0$.

PROOF. We know that the code C corresponds to the principal ideal $(g(\alpha))$ in R_n . Hence $y(\alpha) \in C$ if and only if $y(\alpha) = f(\alpha)g(\alpha)$ for some $f(\alpha) \in R_n$. So, if $y(\alpha)$ is a codeword, then $y(\alpha)h(\alpha) = f(\alpha)g(\alpha)h(\alpha) = f(\alpha) \cdot 0 = 0$.

Conversely, suppose $y(\alpha)h(\alpha) = 0$. This means that in $F[X]$, we have $y(X)h(X) \in (X^n - 1)$, the principal ideal generated by $X^n - 1$. In $F[X]$, apply the Division Algorithm to write $y(X) = f(X)g(X) + r(X)$, where $\deg(r(X)) < n - k$. Then

$$y(X)h(X) = f(X)g(X)h(X) + r(X)h(X) \in (X^n - 1),$$

and since $g(X)h(X) \in (X^n - 1)$, we conclude that $r(X)h(X) \in (X^n - 1)$. But the degree of $r(X)$ is less than $n - k$ and the degree of $h(X)$ is k . Hence we have $\deg(r(X)h(X)) < n$. The only way that $r(X)h(X)$ can then be a multiple of $X^n - 1$ is to have $r(X)h(X) = 0$. Thus we conclude that

$r(X) = 0$ and so $y(X) \in \langle (g(X)) \rangle$, which shows that $y(\alpha)$ is a codeword in C . ■

Suppose $h(X) = h_0 + h_1X + \cdots + X^k$ and suppose $y(X) = b_0 + b_1X + \cdots + b_{n-1}X^{n-1}$. (Recall that we have identified strings in $GF(q)^n$ with polynomials in $F[X]$ of degree at most $n-1$). Then Proposition (6.14) says that $y(X)$ is a codeword if and only if $y(X)h(X)$ is in $(X^n - 1)$. In particular, this means that the coefficients of $x^k, x^{k+1}, \dots, x^{n-1}$ in the product $y(X)h(X)$ must all be 0. This gives us the $n - k$ equations

$$\begin{array}{ccccccc} b_0 + b_1h_{k-1} + \cdots + b_kh_0 & & & & & & = 0 \\ b_1 + b_2h_{k-1} + \cdots + b_{k+1}h_0 & & & & & & = 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ b_{n-k-1} + b_{n-k}h_{k-1} + \cdots + b_{n-1}h_0 & & & & & & = 0 \end{array}$$

In matrix form, we may write this system of equations as

$$\begin{bmatrix} 1 & h_{k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 & h_{k-1} & \cdots & h_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{bmatrix} = \hat{0}_{n-k}$$

Since the $(n - k) \times n$ matrix H on the left of this equation clearly has rank $n - k$ (it's in row echelon form) and since a string \hat{y} is a codeword in C only if $H\hat{y} = \hat{0}_{n-k}$, we see that H must be a parity check matrix for C . We record this as

(6.15) PROPOSITION. Suppose C is an (n, k) cyclic code with check polynomial $h(X) = h_0 + h_1X + \cdots + h_{k-1}X^{k-1} + X^k$. Then a parity check matrix for C is

$$H = \begin{bmatrix} 1 & h_{k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 & h_{k-1} & h_{k-2} & \cdots & h_0 \end{bmatrix}.$$

► **(6.16) EXAMPLE.** Recall from Example (6.12) that over $GF(3)$ the polynomial $X^4 - 1$ factors into the product $(X - 1)(X + 1)(X^2 + 1)$ of irreducible polynomials. Let C be the ternary cyclic code of length 4 with

generator polynomial $g(X) = X^2 + 1$. Then the check polynomial of C is $h(X) = (X - 1)(X + 1) = X^2 - 1 = X^2 + 2$ and a parity check matrix for C is

$$H = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}.$$

There are two good ways for us to determine whether a string of length 4 is a codeword here. Given $\hat{y} = y_0y_1y_2y_3$, we can compute the matrix product $H\hat{y}$ and see if we get $\hat{0}_2$. Or we can divide the polynomial $y_0 + y_1X + y_2X^2 + y_3X^3$ by $X^2 - 1$ and see if we get a remainder of 0. For a concrete example, suppose $\hat{y} = 1212$. Then $H\hat{y} =$

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

so \hat{y} is a codeword. Or we also have

$$\begin{array}{r} X^2 + 1 \overline{) 2X^3 + X^2 + 2X + 1} \\ \underline{2X^3 + + 2X} \\ X^2 + + 1 \\ \underline{X^2 + + 1} \\ \end{array} \quad \blacktriangleleft$$

So, we have seen that if we know the generator polynomial of a cyclic code, then we may find the dimension of the code, a generator matrix and a parity check matrix. In general, there is no easy way to determine the minimum distance of a cyclic code from the generator polynomial.

We promised earlier in this section that we would show that binary Hamming codes are (equivalent to) cyclic codes.

(6.17) PROPOSITION. The binary Hamming code $\text{Ham}(r, 2)$ is equivalent to a cyclic code.

PROOF. Recall from Chapter IV that the finite field $GF(2^r)$ is isomorphic to

$$GF(2)[X]/(p(X)),$$

where $p(X)$ is any irreducible polynomial of degree r . By Theorem (IV.8.12), there is a primitive element γ in $GF(2^r)$, which means that γ generates the

cyclic multiplicative group of order $2^r - 1$ of nonzero elements in $GF(2^r)$. Consider the matrix $H = [1 \ \gamma \ \gamma^2 \ \cdots \ \gamma^{2^r-2}]$. This matrix has entries in $GF(2^r)$, but by the identification between $GF(2^r) = GF(2)[X]/(p(X))$ and $GF(2)^r$ at the beginning of this section (see (6.1)), we may identify each entry in H with a column of r binary digits (i.e., a vector in $GF(2)^r$). Since the entries in H run through all the nonzero elements in $GF(2^r)$, when we view H as a binary matrix, its columns contain all the nonzero elements in $GF(2)^r$ (not necessarily in order of the integers that these binary columns represent). Let C be the binary code with parity check matrix H . Then C is equivalent to the binary Hamming code $\text{Ham}(r, 2)$ (because the columns of H are all the nonzero elements in $GF(2)^r$). We leave it to the reader to see that, with the identifications we have made, a string $\hat{y} = b_0b_1 \cdots b_{2^r-2}$ is a codeword if and only if

$$[1 \ \gamma \ \cdots \ \gamma^{2^r-2}] \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{2^r-2} \end{bmatrix} = [b_0 + b_1\gamma + \cdots + b_{2^r-2}\gamma^{2^r-2}] = [0].$$

Thus the codewords in C correspond exactly to the polynomials $f(X) \in GF(2)[X]$ of degree at most $2^r - 2$ such that $f(\gamma) = 0$. These polynomials are contained in the ideal in $GF(2)[X]$ generated by the minimal polynomial $g(X)$ of γ (see Exercise IV.7.9 for the definition of minimal polynomial). Note that since γ is a primitive element in $GF(2^r)$, we have $\gamma^{2^r-1} = 1$. Therefore, γ is a root of the polynomial $X^n - 1$, where $n = 2^r - 1$. It follows from Exercise IV.7.9 that $g(X)$ divides $X^n - 1$. Hence, by (6.7) and Proposition (6.8), C is a cyclic code of length $n = 2^r - 1$. ■

We note that not all Hamming codes are equivalent to cyclic codes, since it can be shown (see Exercise 6.6) that the ternary Hamming code $\text{Ham}(2,3)$ is not equivalent to a cyclic code.

To close this section, we describe a clever idea for constructing t -error-correcting codes, for any t , over $GF(p)$, where p is a prime. The codes we are about to describe are the simplest type of BCH codes. These codes were discovered independently by Bose and Ray-Chaudhuri (1960) and Hocquenghem (1959).

Suppose we are given t and p and we wish to construct a t -error-correcting code over $GF(p)$. Put $d = 2t + 1$. Choose m such that $p^m \geq d$ and put $n = p^m - 1$. We wish to construct a matrix such that every set of $d - 1$ columns in this matrix is a linearly independent set.

Let γ denote a primitive element in $GF(p^m)$ and consider the $(d-1) \times n$ matrix

$$H' = \begin{bmatrix} 1 & \gamma & \gamma^2 & \cdots & \gamma^{n-1} \\ 1 & \gamma^2 & \gamma^4 & \cdots & \gamma^{2(n-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \gamma^{d-1} & \gamma^{2(d-1)} & \cdots & \gamma^{(n-1)(d-1)} \end{bmatrix}.$$

A matrix formed from any collection of $d-1$ columns of H' looks like

$$M = \begin{bmatrix} \gamma^{j_1} & \gamma^{j_2} & \cdots & \gamma^{j_{d-1}} \\ \gamma^{2j_1} & \gamma^{2j_2} & \cdots & \gamma^{2j_{d-1}} \\ \vdots & \vdots & \cdots & \vdots \\ \gamma^{j_1(d-1)} & \gamma^{j_2(d-1)} & \cdots & \gamma^{j_{d-1}(d-1)} \end{bmatrix}.$$

The determinant of this matrix is

$$\gamma^{j_1+j_2+\cdots+j_{d-1}} \cdot \det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \gamma^{j_1} & \gamma^{j_2} & \cdots & \gamma^{j_{d-1}} \\ \vdots & \vdots & \cdots & \vdots \\ (\gamma^{j_1})^{(d-2)} & (\gamma^{j_2})^{(d-2)} & \cdots & (\gamma^{j_{d-1}})^{(d-2)} \end{bmatrix}.$$

This last determinant has a very special form and is called a Vandermonde determinant. We leave it as an exercise to see that the value of this determinant is

$$\prod_{i>k} (\gamma^{j_i} - \gamma^{j_k}),$$

which of course is nonzero here.

The upshot of all this is that every set of $d-1$ columns of H' is linearly independent over $GF(p^m)$ and hence over $GF(p)$. If we view H' as being a matrix with entries in $GF(p)$ by identifying each element in $GF(p^m)$ with a (column) vector in $GF(p)^m$ as in (6.1), then every set of $d-1$ columns of H' will still be linearly independent over $GF(p)$. Let C be the code of length n over $GF(p)$ that is the null space of H' , viewed as a matrix with entries in $GF(p)$. We note that H' is probably not a parity check matrix for C since in all likelihood the rows of H' are not linearly independent. Indeed, it is not so easy to determine the rank of H' . However, a parity check matrix for C will consist of certain rows of H' and it can be shown that every set of $d-1$ columns in this parity check matrix for C will still

be linearly independent over $GF(p)$. Thus we have $d(C) \geq d = 2t + 1$ by Theorem (3.10), and so C will be t -error-correcting, if nonzero, by Theorem (3.8). Actually, it may turn out that C is trivial, since H' may have rank n . This cannot happen though if we take m to be sufficiently large, say such that $n = p^m - 1 > m(d - 1) + 1$, for then the matrix H' must have rank less than $n - 1$. A table of binary BCH codes of length at most 255 appears in [23].

One may see as in the proof of Proposition (6.17) that the code C corresponds to all polynomials of degree at most $n - 1$ in $GF(p)[X]$ that vanish (equal zero) at γ^i for $i = 1, 2, \dots, d - 1$. These polynomials are contained in the ideal of $GF(p)[X]$ that is generated by the least common multiple of the minimal polynomials of the γ^i . Note that since each γ^i is an element of a cyclic group of order n , we have $(\gamma^i)^n = 1$ for all i . It follows that for every i , the minimal polynomial of γ^i divides $X^n - 1$ and hence that the least common multiple of these minimal polynomials divides $X^n - 1$. In other words, this code C is a cyclic code of length n . These codes have been studied extensively, and there are decoding algorithms for such codes that are much better than coset decoding. We refer the interested reader to MacWilliams and Sloane [23], Mackiw [22], and Blahut [3].

- **(6.18) EXAMPLE.** Suppose we wish to construct a 2-error-correcting BCH binary code. Then $d = 5$. At first, one might want to take $m = 3$ and consider $GF(8)$, the field of Example (IV.7.8). A primitive element in this field is α and one could form the 4×7 matrix

$$H' = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^8 & \alpha^{10} & \alpha^{12} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} \\ 1 & \alpha^4 & \alpha^8 & \alpha^{12} & \alpha^{16} & \alpha^{20} & \alpha^{24} \end{bmatrix}.$$

Every four columns of this matrix are linearly independent. We can identify this matrix with a 12×7 binary matrix by using the multiplication table in the field $GF(8)$ and the correspondence

$$1 \mapsto \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \alpha \mapsto \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \alpha^2 \mapsto \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

However, the resulting 12×7 binary matrix has rank 7 and so defines the zero code. (Actually, we should realize from the beginning that there cannot

be a 2-error-correcting binary code of length 7 with more than one nonzero codeword, since the sum of two vectors of weight at least 5 in $GF(2)^7$ can have weight at most 4.)

Therefore, if we want a nontrivial 2-error-correcting BCH binary code, we must try $m = 4$. Consider the field $GF(16)$ of Example (IV.8.16). A primitive element is α and we can form the 4×15 matrix

$$H' = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{14} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{28} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{42} \\ 1 & \alpha^4 & \alpha^8 & \cdots & \alpha^{56} \end{bmatrix}.$$

This matrix corresponds to a 16×15 binary matrix, and it can be shown that this binary matrix has rank 8. Thus the null space of this binary matrix is a $(15, 7)$ binary BCH code and it turns out that this code has minimum distance precisely 5 (see [23] for details). It can be shown that α, α^2 , and α^4 all have the same minimal polynomial, namely $p(X) = X^4 + X + 1$, over $GF(2)$ and that the minimal polynomial of α^3 over $GF(2)$ is $q(X) = X^4 + X^3 + X^2 + X + 1$. It follows that the generator polynomial of the cyclic code we have constructed is $p(X)q(X)$. ◀

EXERCISES

- 6.1. Prove Lemma (6.3).
- † 6.2. Let E_n denote the binary code consisting of all even weight vectors of length n .
 - a) Show that E_n is cyclic.
 - b) Find a parity check matrix for E_n .
 - c) Find the generator polynomial and check polynomial for E_n .
- † 6.3. If n is even, give a generator matrix for an $(n, 2)$ cyclic code.
- 6.4. Show that the ring R_n is not an integral domain.
- 6.5. Give generator and check polynomials for all binary cyclic codes of length 5. (Hint: Check to see if $X^5 - 1$ has an irreducible factor of degree 2 or degree 3.)
- † 6.6. a) Find the generator polynomial, the check polynomial, a generator matrix, and a parity check matrix for each ternary cyclic code of length 4.

- b) Show that the ternary Hamming code $\text{Ham}(2,3)$ is not equivalent to a cyclic code. (Hint: Note the minimum distance of each code in part (a).)

- † 6.7. Give generator and check polynomials for all binary cyclic codes of length 7. (Hint: $X^7 - 1$ has two irreducible factors of degree 3.)
- 6.8. Suppose $g(X) = a_0 + a_1X + \cdots + X^r$ is the generator polynomial of a cyclic code of length n . Prove that $a_0 \neq 0$. (Hint: If $a_0 = 0$, then $g(X) = Xf(X)$, where $\deg(f) = r - 1$. Show that $f(X)$ is a codeword.)
- † 6.9. Suppose $X^n - 1$ is the product of m distinct irreducible polynomials over $GF(q)$. How many cyclic codes of length n are there over $GF(q)$?
- † 6.10. Show that there is a binary cyclic code of length 23 with generator polynomial $X^{11} + X^{10} + X^6 + X^5 + X^4 + X^2 + 1$. (This is the perfect (23,12,7) binary Golay code.)
- 6.11. Show that there is a ternary cyclic code of length 11 with generator polynomial $X^5 + X^4 - X^3 + X^2 - 1$. (This is the perfect (11,6,5) ternary Golay code.)
- 6.12. (Vandermonde determinant) Suppose that a_1, a_2, \dots, a_n are n distinct elements in a field F . Show that

$$\det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ a_1^2 & a_2^2 & \cdots & a_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ a_1^{n-1} & a_2^{n-1} & \cdots & a_n^{n-1} \end{bmatrix} = \prod_{i>k} (a_i - a_k).$$

(Hint: Subtract a_1 times the i^{th} row from the $(i+1)^{\text{st}}$ row, factor out a common term from each column, expand the determinant by the first column and use induction.)

- † 6.13. Give a parity check matrix for a 2-error-correcting code of length 6 and dimension 2 over $GF(7)$. (Hint: Use Exercise 6.12.)
- 6.14. Suppose $\alpha \in GF(2^r)$. Show that the minimal polynomial (cf. Exercise IV.7.9) of α is the same as the minimal polynomial of α^2 . (Hint: Use Lemma (IV.8.7).)

- 6.15.** Consider the field $GF(16)$ of Examples (IV.8.9.2) and (IV.8.16).
- a) Show that $X^4 + X + 1$ is the minimal polynomial of $\alpha, \alpha^2, \alpha^4$, and α^8 over $GF(2)$.
 - b) Show that $X^4 + X^3 + X^2 + X + 1$ is the minimal polynomial of $\alpha^3, \alpha^6, \alpha^9$, and α^{12} over $GF(2)$.
 - c) Show that $X^2 + X + 1$ is the minimal polynomial of α^5 and α^{10} over $GF(2)$.
 - d) Show that $X^4 + X^3 + 1$ is the minimal polynomial of $\alpha^7, \alpha^{11}, \alpha^{13}$, and α^{14} over $GF(2)$.
- 6.16.** Describe how you would construct a (15,5,7) binary BCH code. What is the generator polynomial of this code?

CHAPTER VI

Boolean Algebras and Switching Functions

The study of Boolean algebras dates back to 1854 when the British mathematician George Boole published his book "An Investigation of the Laws of Thought." The point of Boole's work was to put logic within a mathematical framework. Boolean algebras were studied as a branch of abstract mathematics until the 1930's when Claude Shannon (the same person who was instrumental in developing coding theory) realized that they could be applied to the study of electrical circuits. In this chapter, we will introduce Boolean algebras and consider some of their elementary properties. The main goal of the chapter is to consider the application of Boolean algebras to switching functions and circuit design.

1. Axioms and Examples

We begin with the main model for a Boolean algebra, the power set of a set.

- **(1.1) EXAMPLE.** Let S be a set and let $\mathcal{P}(S)$ denote the set of all subsets, or power set, of S . We consider the following three operations on $\mathcal{P}(S)$: union (\cup), intersection (\cap), and complement ($\bar{}$). (If A is a subset of S , then the complement of A in S , denoted \bar{A} , is given by $\bar{A} = \{x \in S : x \notin A\}$.) Obviously, $\mathcal{P}(S)$ is closed under these three operations. Among the properties that hold for any $A, B, C \in \mathcal{P}(S)$ are the following:

1) $A \cup B = B \cup A$ and $A \cap B = B \cap A$.

- 2) $(A \cup B) \cup C = A \cup (B \cup C)$ and $(A \cap B) \cap C = A \cap (B \cap C)$.
- 3) $A \cap (A \cup B) = A$ and $A \cup (A \cap B) = A$.
- 4) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ and $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.
- 5) $A \cup \emptyset = A$ and $A \cap S = A$.
- 6) $A \cup S = S$ and $A \cap \emptyset = \emptyset$.
- 7) $A \cup A = A$ and $A \cap A = A$.
- 8) $A \cup \bar{A} = S$ and $A \cap \bar{A} = \emptyset$.
- 9) $\overline{(\bar{A})} = A$.
- 10) $\overline{A \cup B} = \bar{A} \cap \bar{B}$ and $\overline{A \cap B} = \bar{A} \cup \bar{B}$.

Note that in each number above (except (9)), the second listed property follows from the first if we replace \cup, \cap, \emptyset , and S with \cap, \cup, S , and \emptyset , respectively. ◀

As usual, in defining an algebraic system we want to assume the fewest possible number of axioms that will allow us to derive other desired properties as theorems. The following system of axioms for a Boolean algebra was given by Huntington in 1904. In the exercises, we will ask you to show that the properties analogous to properties (2), (3), (7), (9), and (10) above can be derived from the axioms.

(1.2) DEFINITION. A *Boolean algebra* $(A, \vee, \wedge, \bar{}, 0, 1)$ is a set A containing two distinct special elements 0 and 1 and with two binary operations \vee and \wedge on A and a unary operation $\bar{}$ on A such that:

- 1) $a \vee b = b \vee a$ and $a \wedge b = b \wedge a$ for all $a, b \in A$ (Commutativity).
- 2) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ and $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ for all $a, b, c \in A$ (Distributivity).
- 3) $a \vee 0 = a$ and $a \wedge 1 = a$ for all $a \in A$ (Identity).
- 4) $a \vee \bar{a} = 1$ and $a \wedge \bar{a} = 0$ for all $a \in A$ (Complement).

We note that (A, \vee, \wedge) is *not* a ring, since neither \vee nor \wedge satisfies the Inverse Axiom necessary for addition in a ring; however, one can use \vee, \wedge , and $\bar{}$ to define an addition on A as we will see in Exercise 1.15. We will refer to the operation \vee as “join” or “disjunction” or “sum” (but keep in mind that the only element with an inverse with respect to \vee is 0). We will refer to the operation \wedge as “meet” or “conjunction” or “product.”

As we noted above in the power set example, there is an obvious “duality” here in the sense that if one starts with any of the above properties and replaces $\vee, \wedge, 0, 1$ with $\wedge, \vee, 1, 0$, respectively, then one obtains another one of these properties. It follows that we have the

(1.3) PRINCIPLE OF DUALITY. Any theorem about a Boolean algebra $(A, \vee, \wedge, \bar{}, 0, 1)$ yields another “dual” theorem if we replace $\vee, \wedge, 0, 1$

with $\wedge, \vee, 1, 0$, respectively.

As an illustration, we will prove a type of cancellation property and then state the dual result.

(1.4) LEMMA. Suppose A is a Boolean algebra and $a, b, c \in A$. If $a \vee b = a \vee c$ and $\bar{a} \vee b = \bar{a} \vee c$, then $b = c$.

PROOF. We have

$$\begin{aligned} b &= 0 \vee b = (a \wedge \bar{a}) \vee b \\ &= (a \vee b) \wedge (\bar{a} \vee b) = (a \vee c) \wedge (\bar{a} \vee c) \\ &= (a \wedge \bar{a}) \vee c = 0 \vee c = c \end{aligned}$$

and the proof is complete. ■

The dual result corresponding to Lemma (1.4) is

(1.4') LEMMA. Suppose A is a Boolean algebra and $a, b, c \in A$. If $a \wedge b = a \wedge c$ and $\bar{a} \wedge b = \bar{a} \wedge c$, then $b = c$.

(1.5) COROLLARY. $\bar{0} = 1$.

PROOF. By the Complement Axiom, we have $0 = 0 \wedge \bar{0}$ and by the Identity Axiom, we have $0 = 0 \wedge 1$. By Exercise 1.1, we have $\bar{0} \wedge \bar{0} = \bar{0}$ and from the Identity Axiom we have $\bar{0} \wedge 1 = \bar{0}$. Applying (1.4'), with $a = 0$, we conclude that $\bar{0} = 1$.

(1.5') COROLLARY. $\bar{1} = 0$.

As we stated above, the main example of a Boolean algebra is the power set of a set. Here are some other examples.

► (1.6) EXAMPLES

1) Put $\mathbf{B} = \{0, 1\}$ and define operations by

$$\begin{aligned} 0 \vee 0 &= 0, 0 \vee 1 = 1, 1 \vee 0 = 1, 1 \vee 1 = 1 \\ 0 \wedge 0 &= 0, 0 \wedge 1 = 0, 1 \wedge 0 = 0, 1 \wedge 1 = 1 \\ \bar{0} &= 1, \bar{1} = 0. \end{aligned}$$

Then \mathbf{B} is a Boolean algebra. Note that \wedge is the same as multiplication on \mathbf{Z}_2 , but that \vee is not the same as addition on \mathbf{Z}_2 (since $1 \vee 1 = 1$).

2) The product of Boolean algebras is a Boolean algebra with operations defined in a coordinatewise manner. For example,

$$\mathbf{B}^n = \underbrace{\mathbf{B} \times \cdots \times \mathbf{B}}_{n \text{ times}}$$

is a Boolean algebra if we define

$$\begin{aligned}(a_1, a_2, \dots, a_n) \vee (b_1, b_2, \dots, b_n) &= (a_1 \vee b_1, a_2 \vee b_2, \dots, a_n \vee b_n) \\ (a_1, a_2, \dots, a_n) \wedge (b_1, b_2, \dots, b_n) &= (a_1 \wedge b_1, a_2 \wedge b_2, \dots, a_n \wedge b_n) \\ \overline{(a_1, a_2, \dots, a_n)} &= (\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n).\end{aligned}$$

Examples (1) and (2) will play a crucial role in the rest of this chapter.

3) Let A denote the subset of \mathbf{P} consisting of all divisors of 30. So we have

$$A = \{1, 2, 3, 5, 6, 10, 15, 30\}.$$

We put a Boolean algebra structure on A by defining

$$\begin{aligned}a \vee b &= \text{the least common multiple of } a \text{ and } b \\ a \wedge b &= \text{the greatest common divisor of } a \text{ and } b \\ \bar{a} &= 30/a.\end{aligned}$$

The "0" in this Boolean algebra is the integer 1, and the "1" in this Boolean algebra is the integer 30. If $a \in A$, then the least common multiple of a and 1 is a and the greatest common divisor of a and 30 is a ; thus, the Identity Axiom is satisfied. Also, the least common multiple of a and $30/a$ is 30, and the greatest common divisor of a and $30/a$ is 1; thus, the Complement Axiom is satisfied as well.

To see that the Commutativity and Distributivity Axioms are satisfied, we first note that if $a \in A$, then the prime factorization of a is

$$a = 2^{a_1} 3^{a_2} 5^{a_3}$$

where a_1, a_2, a_3 are each either 0 or 1; i.e., $a_i \in \mathbf{B}$ for $i = 1, 2, 3$. Now, if $a = 2^{a_1} 3^{a_2} 5^{a_3}$ and $b = 2^{b_1} 3^{b_2} 5^{b_3}$ are two elements of A , then we have

$$\begin{aligned}a \vee b &= 2^{a_1 \vee b_1} 3^{a_2 \vee b_2} 5^{a_3 \vee b_3} = 2^{b_1 \vee a_1} 3^{b_2 \vee a_2} 5^{b_3 \vee a_3} = b \vee a \\ a \wedge b &= 2^{a_1 \wedge b_1} 3^{a_2 \wedge b_2} 5^{a_3 \wedge b_3} = 2^{b_1 \wedge a_1} 3^{b_2 \wedge a_2} 5^{b_3 \wedge a_3} = b \wedge a.\end{aligned}$$

Thus, the Commutativity Axiom is satisfied in A . Similarly, ~~one~~^{one} may use the distributivity in \mathbf{B} to see that the Distributivity Axiom is satisfied in A . It was critical in this example that 30 was the product of *distinct* primes. ◀

The original motivation for the study of Boolean algebras was the application of these algebras to logic. Suppose that p and q are logical statements. Examples of logical statements are

(1.7) Every integer is even

and

(1.8) Some integers are prime.

One defines $p \vee q$ to be the disjunction of p and q . The disjunction of (1.7) and (1.8) is

Every integer is even or some integers are prime.

The disjunction $p \vee q$ is true if and only if p is true or q is true (or both are true). One defines $p \wedge q$ to be the conjunction of p and q . The conjunction of (1.7) and (1.8) is

Every integer is even and some integers are prime.

The conjunction of p and q is true if and only if both p and q are true. One defines \bar{p} to be the negation of p . The negation of (1.7) is

Some integers are not even.

The negation of p is true if and only if p is false.

(1.9) DEFINITION. A *well-formed formula*, or *logical expression*, involving the statements p_1, p_2, \dots, p_n is defined as follows:

- 1) p_i is a well-formed formula, $i = 1, 2, \dots, n$.
- 2) The negation of a well-formed formula is a well-formed formula.
- 3) The disjunction or conjunction of two well-formed formulas is a well-formed formula.

The collection of all well-formed formulas is clearly closed under the operations $\vee, \wedge, \bar{}$, but it is *not* a Boolean algebra, for the Identity and Complement Axioms are not satisfied. However, we can obtain a Boolean algebra in the following manner. Suppose p_1, \dots, p_n are logical statements. Let A denote the set of all well-formed formulas involving p_1, p_2, \dots, p_n .

(1.10) DEFINITION. Two expressions in A are said to be *equivalent* if they have the same truth table.

A truth table gives the logical value of the expression for all possible logical values of the statements involved in the expression.

► (1.11) **EXAMPLE.** The truth table for $(p_1 \vee \bar{p}_2) \wedge p_3$ is

p_1	p_2	p_3	$(p_1 \vee \bar{p}_2) \wedge p_3$
\bar{F}	\bar{F}	\bar{F}	\bar{F}
\bar{F}	\bar{F}	T	T
\bar{F}	T	\bar{F}	\bar{F}
\bar{F}	T	T	\bar{F}
T	\bar{F}	\bar{F}	\bar{F}
T	\bar{F}	T	T
T	T	\bar{F}	\bar{F}
T	T	T	T

It is easy to see that the relation of “having the same truth table” is an equivalence relation on A , and it is not hard to see that the operations \vee, \wedge, \neg yield well-defined operations on the set of equivalence classes. This set of equivalence classes is then a Boolean algebra with the “0” of this algebra being the class of “contradictions” (statements that are always false, like $p_1 \wedge \bar{p}_1$) and with the “1” of this algebra being the class of “tautologies” (statements that are always true, like $p_1 \vee \bar{p}_1$).

Our main interest in Boolean algebras is their application to electrical circuits. By an electrical circuit, we mean a collection of switches connected in series or parallel. Each switch may be open or closed. We will use 0 to denote a switch being open and 1 to denote a switch being closed.

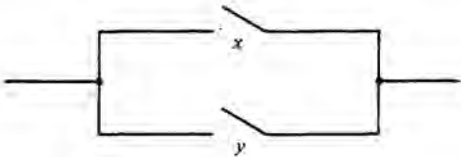


Figure 1.1



Figure 1.2

Suppose x and y are two switches or, more generally, two circuits. We define $x \vee y$ to be the circuit obtained by connecting x and y in parallel (see Figure 1.1). Then $x \vee y$ is closed if and only if either x is closed or y is closed. We define $x \wedge y$ to be the circuit obtained by connecting x and y in series (see Figure 1.2). The circuit $x \wedge y$ is closed if and only if both x and y are closed. We define \bar{x} to be the switch (or circuit if x is a circuit) that is closed if and only if x is open. Comparing this situation with the case of mathematical logic, we see that the switches play the role of logical statements and the circuits play the role of well-formed formulas. As with well-formed formulas, the circuits do not form a Boolean algebra, but we may try to define an equivalence relation, as we did with well-formed formulas, to obtain a Boolean algebra.

Given a circuit, we may form a “closure” table, similar to a truth table, that tells when the circuit is open or closed depending on the closure of the switches in the circuit.

- (1.12) **EXAMPLE.** Consider the circuit E shown in Figure 1.3. Using the operations defined above, we may represent this circuit algebraically by writing

$$E = (x \wedge y) \vee (x \wedge z) \vee (x \wedge w).$$

The closure table for E is shown in Table (1.13). ◀

(1.13) **TABLE**

x	y	z	w	E
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

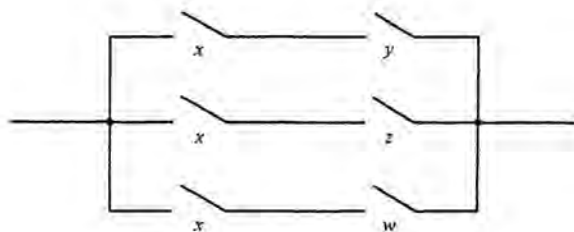


Figure 1.3

Now consider all (parallel and series) circuits constructed from a fixed set of switches $\{x_1, x_2, \dots, x_n\}$ and their complements $\{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n\}$. Define an equivalence relation on this set by declaring two circuits to be equivalent if and only if they have the same closure tables. Then it may be seen that the set of equivalence classes forms a Boolean algebra under the operations induced by \vee , \wedge , and $\bar{}$.

- **(1.14) EXAMPLE.** By constructing a closure table or by applying the Distributive Axiom, it is easy to see that the circuit of Example (1.12) is equivalent to the circuit

$$x \wedge (y \vee z \vee w).$$

Note that this circuit contains fewer switches than the circuit in (1.12). ◀

In §3 we will return to electrical circuits, but we will consider gating networks instead of series and parallel circuits. Our main objectives in the later sections of this chapter will be to recognize when two networks (or circuits) are equivalent and to identify a minimal (so hopefully least expensive and fastest) network equivalent to a given one.

EXERCISES

In the following exercises, we assume that $(A, \vee, \wedge, \bar{}, 0, 1)$ is a Boolean algebra and $a, b, c \in A$.

- 1.1. Establish the Idempotent Law: $a \vee a = a$ and $a \wedge a = a$. (Hint: $a = a \vee 0$ and $0 = a \wedge \bar{a}$. Note that, by duality, you need only show that $a \vee a = a$.)
- 1.2. Show that $a \wedge 0 = 0$ and $a \vee 1 = 1$. (Hint: $a \wedge 0 = (a \wedge 0) \vee (a \wedge \bar{a})$.)
- 1.3. Establish the Absorption Law: $a \wedge (a \vee b) = a$ and $a \vee (a \wedge b) = a$. (Hint: $a \wedge (a \vee b) = (a \vee 0) \wedge (a \vee b)$.)

- † 1.4. Show that \wedge and \vee are associative. (Hint: Use Exercise 1.3 and Lemma (1.4') to show that $a \vee (b \vee c) = (a \vee b) \vee c$.)
- 1.5. Show that if $a \vee b = 0$, then $a = 0$. (Hint: Use Exercise 1.3.) Conclude by duality that if $a \wedge b = 1$, then $a = 1$.
- † 1.6. Show that if $a \vee b = 1$ and $a \wedge b = 0$, then $b = \bar{a}$. Conclude that each element has a unique complement. (Hint: Use that $\bar{a} = \bar{a} \vee (a \wedge b)$ and $\bar{a} \vee b = (\bar{a} \vee b) \wedge (a \vee b)$.)
- 1.7. Show that $\bar{\bar{a}} = a$.
- † 1.8. Establish DeMorgan's Laws: $\overline{a \wedge b} = \bar{a} \vee \bar{b}$ and $\overline{a \vee b} = \bar{a} \wedge \bar{b}$. (Hint: Use Exercise 1.6.)
- 1.9. Show that $a \wedge \bar{b} = 0$ if and only if $a \wedge b = a$. State the dual result. (Hint: $a \wedge \bar{b} = 0$ if and only if $\bar{a} \vee b = 1$ by DeMorgan's Laws.)
- † 1.10. Show that each of the operations \vee and \wedge in a Boolean algebra has a *unique* identity element.
- 1.11. Define a Boolean algebra structure on the subset of \mathbf{P} consisting of the divisors of 1155.
- 1.12. Determine the truth table for the logical expression $p \vee \bar{q}$.
- 1.13. Draw the electrical circuit that is represented algebraically by $E = (x \vee y) \wedge (x \vee z)$ and determine the closure table for this circuit.
- † 1.14. Show that the circuit $E_1 = (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3)$ is equivalent to the circuit $E_2 = x_1 \wedge x_3$.
- 1.15. Define an operation \oplus on A by
- $$a \oplus b = (a \wedge \bar{b}) \vee (\bar{a} \wedge b).$$
- a) Show that (A, \oplus) is a commutative group.
- b) Show that (A, \oplus, \wedge) is a commutative ring with identity such that $a \wedge a = a$ for all $a \in A$. Such a ring is called a Boolean ring (cf. Exercise IV.1.12).
- 1.16. Suppose (R, \oplus, \wedge) is a Boolean ring (see the previous exercise). Define operations \vee and $\bar{}$ on R by

$$r \vee s = r \oplus s \oplus (r \wedge s) \text{ and } \bar{r} = r \oplus 1.$$

Show that $(R, \vee, \wedge, \bar{}, 0, 1)$ is a Boolean algebra.

2. Finite Boolean Algebras

Let S be a set. It is easy to see how to define a partial ordering on the power set of S ; namely, if X and Y are subsets of S , then we say $X \leq Y$ if $X \subseteq Y$. Note that $X \subseteq Y$ if and only if $X \cap Y = X$ (this was Proposition (I.1.2)). This observation allows one to extend this idea to get a partial ordering on an arbitrary Boolean algebra.

(2.1) DEFINITION. Suppose A is a Boolean algebra. Define a relation \leq on A by

$$a \leq b \text{ if and only if } a \wedge b = a.$$

(2.2) PROPOSITION. The relation \leq is a partial ordering on A .

PROOF. By Exercise 1.1 we have $a \wedge a = a$ for every $a \in A$. Hence $a \leq a$, which shows that \leq is reflexive. Next suppose $a \leq b$ and $b \leq c$. Then we have

$$a = a \wedge b = b \wedge a = b,$$

so \leq is antisymmetric. Finally, if $a \leq b$ and $b \leq c$, then

$$a \wedge c = (a \wedge b) \wedge c = a \wedge (b \wedge c) = a \wedge b = a,$$

where we used the fact that \wedge is associative (Exercise 1.4). Thus we have $a \leq c$ and \leq is transitive. ■

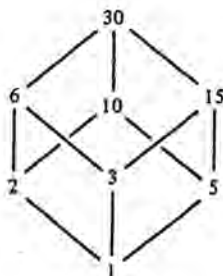


Figure 2.1

- **(2.3) EXAMPLE.** Consider the Boolean algebra of divisors of 30 (Example 1.6.3). We may draw the Hasse diagram (see II.2), which gives a pictorial representation of the relation \leq on this algebra. The diagram is shown in Figure 2.1. ◀

An important role is played by the elements that lie just above the 0 of a Boolean algebra.

(2.4) DEFINITION. Let A be a Boolean algebra, partially ordered by the relation \leq above. A nonzero element $a \in A$ is called an *atom* of A if whenever $b \in A$ and $0 \leq b \leq a$, then either $b = 0$ or $b = a$.

Thus there are no other elements of A that lie strictly between 0 and an atom.

► **(2.5) EXAMPLES**

1) If $\mathcal{P}(S)$ is the power set of a set S , then the atoms in $\mathcal{P}(S)$ are the sets containing exactly one element of S ; i.e., the singleton subsets.

2) In the Boolean algebra pictured in Figure 2.1, the atoms are the primes 2, 3, and 5.

3) In \mathbf{B}^n , the atoms are the n -tuples with precisely one coordinate equal to 1. ◀

(2.6) LEMMA. If a is an atom of A and if b is any element of A , then either $a \wedge b = 0$ or $a \wedge b = a$.

PROOF. Note that $(a \wedge b) \wedge a = a \wedge (a \wedge b) = (a \wedge a) \wedge b = a \wedge b$. Hence we have $a \wedge b \leq a$. Since a is an atom, we must have either $a \wedge b = 0$ or $a \wedge b = a$. ■

(2.7) COROLLARY. If a_1 and a_2 are distinct atoms of A , then we have $a_1 \wedge a_2 = 0$.

PROOF. Since a_1 is an atom, we have that $a_1 \wedge a_2$ is either 0 or a_1 . Since a_2 is an atom, we have that $a_1 \wedge a_2$ is either 0 or a_2 . Since $a_1 \neq a_2$, we conclude that $a_1 \wedge a_2 = 0$. ■

(2.8) LEMMA. Suppose A is a finite Boolean algebra and let a_1, a_2, \dots, a_n be all the atoms in A . If $b \in A$ and if $b \wedge a_i = 0$ for $i = 1, 2, \dots, n$, then $b = 0$.

PROOF. By way of contradiction, suppose $b \neq 0$ and put

$$X = \{a \in A : a \leq b \text{ and } a \neq 0\}.$$

Note that $b \in X$. Since X is finite, we may choose a minimal element $a' \in X$ (in the sense that if $a \in X$ and $a \leq a'$, then $a = a'$). But then, by the definition of X , no element of A lies strictly between 0 and a' , so a' is an atom. By our hypothesis, $b \wedge a' = 0$, but since $a' \leq b$, we also have $b \wedge a' = a'$. Hence $a' = 0$, which is a contradiction. ■

The next result shows that the set of atoms in a finite Boolean algebra is similar to a basis in a finite-dimensional vector space.

(2.9) THEOREM. Suppose A is a finite Boolean algebra. Let a_1, a_2, \dots, a_n be all the atoms in A . Then every element $a \in A$ may be written uniquely (up to order) in the form

$$a = c_1 a_1 \vee c_2 a_2 \vee \cdots \vee c_n a_n$$

where $c_i \in \{0, 1\}$ for $i = 1, 2, \dots, n$. (Here we mean that if $c_i = 1$, then a_i appears in this sum, while if $c_i = 0$, then a_i does not appear.)

PROOF. We first claim that

$$b = a_1 \vee a_2 \vee \cdots \vee a_n = 1.$$

Indeed, we have

$$0 = \bar{b} \wedge b = (\bar{b} \wedge a_1) \vee (\bar{b} \wedge a_2) \vee \cdots \vee (\bar{b} \wedge a_n).$$

By Exercise 1.5, this shows that $\bar{b} \wedge a_i = 0$ for $i = 1, 2, \dots, n$. Hence, by Lemma (2.8), we conclude that $\bar{b} = 0$ and $b = \bar{\bar{b}} = \bar{0} = 1$.

Now if $a \in A$, we have

$$\begin{aligned} a &= a \wedge 1 = a \wedge (a_1 \vee \cdots \vee a_n) \\ &= (a \wedge a_1) \vee (a \wedge a_2) \vee \cdots \vee (a \wedge a_n). \end{aligned}$$

By Lemma (2.6), for each i , either $a \wedge a_i = 0$ or $a \wedge a_i = a_i$. Thus we have represented a as a join (or sum) of atoms. Note that the atoms that occur in this sum are exactly those atoms a_i such that $a_i \leq a$.

To establish the uniqueness, suppose that a may also be written as $a = a'_1 \vee \cdots \vee a'_k$, where the a'_j are atoms. Then, using Corollary (2.7), we see that

$$\begin{aligned} a'_j \wedge a &= (a'_j \wedge a'_1) \vee \cdots \vee (a'_j \wedge a'_k) \\ &= (a'_j \wedge a'_j) = a'_j. \end{aligned}$$

Therefore, $a'_j \leq a$. Conversely, if a' is an atom such that $a' \leq a$, then

$$a' = a' \wedge a = (a' \wedge a'_1) \vee \cdots \vee (a' \wedge a'_k)$$

and applying Corollary (2.7) we see that $a' = a'_j$ for some j . Thus, the *only* way to represent a as a sum of atoms is as the sum of all atoms less than or equal to a . ■

(2.10) COROLLARY. If A is a finite Boolean algebra with n atoms then $|A| = 2^n$.

PROOF. There are two possibilities for each of the coefficients c_1, c_2, \dots, c_n in the (unique) representation of an element of A as a sum of atoms. ■

► **(2.11) EXAMPLE.** In the Boolean algebra \mathbf{B}^n , Theorem (2.9) simply becomes

$$(c_1, c_2, \dots, c_n) = c_1(1, 0, 0, \dots, 0) \vee c_2(0, 1, 0, 0, \dots, 0) \\ \vee \cdots \vee c_n(0, 0, \dots, 0, 1). \blacktriangleleft$$

Next, we wish to establish the fact that any finite Boolean algebra with n atoms actually looks just like \mathbf{B}^n . This is similar to the fact that an n -dimensional vector space over \mathbf{R} looks like \mathbf{R}^n .

We first need a notion of homomorphism of Boolean algebras.

(2.12) DEFINITIONS. A map φ from a Boolean algebra $(A, \vee_A, \wedge_A, \overline{A}, 0_A, 1_A)$ to a Boolean algebra $(B, \vee_B, \wedge_B, \overline{B}, 0_B, 1_B)$ is called a *homomorphism of Boolean algebras* if for all $a_1, a_2 \in A$, the following three properties hold:

- 1) $\varphi(a_1 \vee_A a_2) = \varphi(a_1) \vee_B \varphi(a_2)$
- 2) $\varphi(a_1 \wedge_A a_2) = \varphi(a_1) \wedge_B \varphi(a_2)$
- 3) $\varphi(\overline{a_1^A}) = \overline{\varphi(a_1)^B}$.

An *isomorphism of Boolean algebras* is a 1-1, onto homomorphism of Boolean algebras. Two Boolean algebras are called *isomorphic* if there exists an isomorphism from one to the other.

(2.13) THEOREM. Suppose A is a finite Boolean algebra with n atoms. Then A is isomorphic to \mathbf{B}^n .

PROOF. Let the atoms of A be a_1, a_2, \dots, a_n . Define a map $\varphi: \mathbf{B}^n \rightarrow A$ by

$$\varphi(c_1, c_2, \dots, c_n) = c_1 a_1 \vee c_2 a_2 \vee \cdots \vee c_n a_n.$$

We leave it to the reader to check that φ is a homomorphism of Boolean algebras. That φ is 1-1 and onto follows from Theorem (2.9). ■

By a similar argument, we can show that any finite Boolean algebra is isomorphic to a power set $\mathcal{P}(S)$.

(2.14) THEOREM. Suppose A is a finite Boolean algebra with atoms a_1, a_2, \dots, a_n . Put $S = \{a_1, a_2, \dots, a_n\}$. Then A and $\mathcal{P}(S)$ are isomorphic Boolean algebras.

PROOF. Define a map $\varphi: \mathcal{P}(S) \rightarrow A$ by

$$\varphi(X) = c_1 a_1 \vee c_2 a_2 \vee \cdots \vee c_n a_n,$$

where for each $i = 1, 2, \dots, n$, we have $c_i = 1$ if $a_i \in X$ and $c_i = 0$ if $a_i \notin X$. Again, Theorem (2.9) shows that φ is 1-1 and onto, and we leave it to the reader to show that φ is a homomorphism of Boolean algebras. ■

We note that Theorem (2.14) is not true if we drop the finiteness assumption. However, it is true that every infinite Boolean algebra admits a 1-1 homomorphism into a power set. For details, we refer the reader to Mendelson [24].

EXERCISES

- † 2.1. Show that $a \wedge b = a$ if and only if $a \vee b = b$. Conclude that the relation \leq could also be defined by $a \leq b$ if $a \vee b = b$.
- † 2.2. Suppose A is a Boolean algebra and \leq is defined, as in Definition (2.1), by $a \leq b$ if and only if $a \wedge b = a$. Show that $a \leq b$ if and only if $\bar{b} \leq \bar{a}$.
- 2.3. Let (S, \leq) be a partially ordered set and suppose $X \subseteq S$. An element $u \in S$ is called an *upper bound* for X if $x \leq u$ for all $x \in X$. An upper bound s for X is called a *least upper bound* for X if $s \leq u$ for all upper bounds u for X .
- Show that if a least upper bound exists, then it is unique.
 - If A is a Boolean algebra partially ordered by \leq as in Definition (2.1) and if $a, b \in A$, show that $a \vee b$ is the least upper bound for $\{a, b\}$.
 - Give definitions of a lower bound and a greatest lower bound for a subset of a partially ordered set.

- d) With hypotheses as in part (b), show that $a \wedge b$ is the greatest lower bound for $\{a, b\}$.
- e) A partially ordered set such that each pair of elements has a least upper bound and a greatest lower bound is called a *lattice*. Conclude that every Boolean algebra is a lattice. For more information on lattices, the reader may see Dornhoff and Hohn [10] or Lidl and Pilz [21].
- 2.4. (This exercise uses material from Chapter III.) Let A be the set of all subgroups of the group S_3 (see Example (III.3.13)). Partially order A by inclusion.
- a) Draw the Hasse diagram for this partially ordered set.
- b) Is A a Boolean algebra? Is A a lattice (see the previous exercise)?
- 2.5. Explain why the partially ordered set shown in Figure 2.2 is not a lattice (see Exercise 2.3).



Figure 2.2

- 2.6. Draw Hasse diagrams for the Boolean algebras \mathbf{B}^2 and \mathbf{B}^3 partially ordered by \leq .
- † 2.7. Show that the set of positive divisors of 36 cannot be given the structure of a Boolean algebra.
- 2.8. If $n = p_1 p_2 \cdots p_m$ is the product of m distinct primes, then the set of positive divisors of n can be given the structure of a Boolean algebra (as in Example (1.6.3)). Show that this Boolean algebra is isomorphic to \mathbf{B}^m . What are the atoms in this Boolean algebra?
- † 2.9. True or false (and why): Figure 2.3 can be the Hasse diagram of a Boolean algebra partially ordered by \leq .

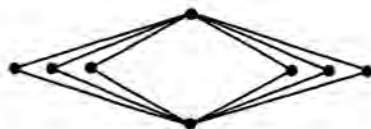


Figure 2.3

- 2.10. Suppose $\varphi : A \rightarrow B$ is a homomorphism of Boolean algebras.
- Show that $\varphi(0_A) = 0_B$ and $\varphi(1_A) = 1_B$.
 - Show that if $a_1 \leq_A a_2$ in A , then $\varphi(a_1) \leq_B \varphi(a_2)$, where $b_1 \leq_B b_2$ if and only if $b_1 \wedge_B b_2 = b_1$.
 - Show that if a is an atom of A , then $\varphi(a)$ is an atom of B .
- 2.11. a) Prove that the map φ in the proof of Theorem (2.13) is a homomorphism of Boolean algebras.
- b) Prove that the map φ in the proof of Theorem (2.14) is a homomorphism of Boolean algebras.

3. Gating Networks and Switching Functions

We now return to the application of Boolean algebras to electrical engineering that we discussed briefly in §1. Instead of the circuits involving electrical switches that we looked at in §1, we will consider here logic, or gating, networks.

(3.1) DEFINITIONS. By a *logic gate* we mean an electrical device that receives one or more inputs, each of which equals 0 or 1, and produces an output of 0 or 1. By a *logic or gating network* we mean a circuit formed by connecting logic gates in series or parallel.

Physically, to say that an input (or output) is 0 or 1 will usually mean that it is one of two possible voltages. The three most common logic gates are:

- 1) The AND gate, which is symbolized by



The AND gate produces an output of 1 if and only if each input is 1. If an AND gate receives n inputs, then it can be considered analogous to n switches being connected in series. If an AND gate receives inputs x_1, x_2, \dots, x_n , then we will represent this gate algebraically by

$$x_1 \wedge x_2 \wedge \cdots \wedge x_n.$$

- 2) The OR gate, which is symbolized by



The OR gate produces an output of 1 if and only if at least one of the inputs equals 1. If an OR gate receives n inputs, then it can be considered

analogous to n switches connected in parallel. If an OR gate receives inputs x_1, x_2, \dots, x_n , then we will represent this gate algebraically by

$$x_1 \vee x_2 \vee \dots \vee x_n.$$

3) The NOT gate (or inverter), which is symbolized by



The NOT gate produces an output of 1 if and only if the (single) input it receives is 0.

► (3.2) EXAMPLES

1) Figures 3.1 and 3.2 both represent gating networks that compute the sum of two single-digit binary numbers x and y . The output s is the last digit of the sum (which is also $x +_2 y$, the sum of x and y viewed as elements of \mathbb{Z}_2). If we view x and y as being in the Boolean algebra \mathbf{B} (see Example (1.6)), then note that $(x \vee y) \wedge \overline{(x \wedge y)}$ equals 0 if $x = y = 0$ or if $x = y = 1$ and equals 1 if one of x and y is 0 and the other is 1. This is precisely what we want for the last digit of the sum of x and y . The same remarks hold for the expression $(\bar{x} \wedge y) \vee (x \wedge \bar{y})$ in Figure 3.2.

The output c , the carry, is the digit in the “two’s place” of the sum. Note that $c = x \wedge y$ is 1 precisely when both x and y are 1. Either of the gating networks in Figures 3.1 and 3.2 is called a *half-adder*.

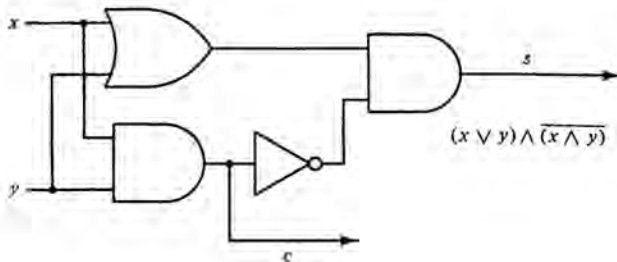


Figure 3.1

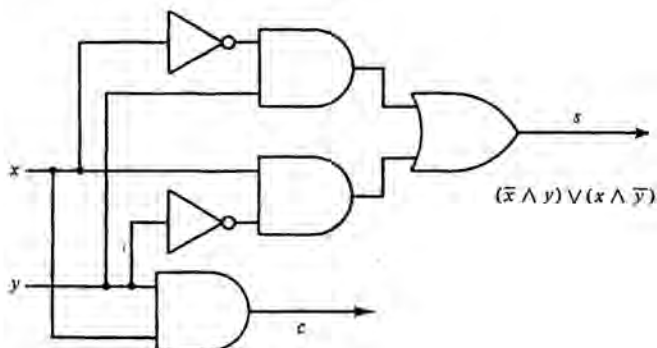


Figure 3.2

2) Figure 3.3 represents a gating network that has as inputs two binary single-digit numbers x and y and the carry c_{in} from a previous addition and that produces as output the sum (with carry c_{out}) of c_{in} , x , and y . Here H.A. stands for a half-adder as in (1) above, s_1 is $x +_2 y$ and s_2 is $s_1 +_2 c_{in}$. Such a network is called a *full-adder*. ◀

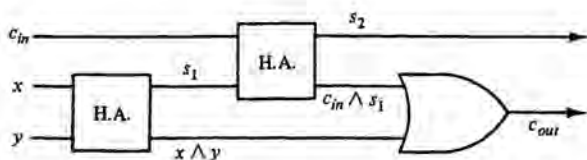


Figure 3.3

To simplify matters somewhat we will usually omit NOT gates from diagrams and will treat \bar{x}_i as an input just like x_i . The important connection with §1 is that there is a 1-1 correspondence between

- 1) gating networks involving independent inputs x_1, x_2, \dots, x_n and their complements and having a single output
and
- 2) logical expressions (or well-formed formulas) in x_1, x_2, \dots, x_n .

(3.3) DEFINITION. We will let \mathcal{E}_n denote the set of all logical expressions involving the symbols x_1, x_2, \dots, x_n . These symbols and their complements will be referred to as *literals*.

Equivalently, \mathcal{E}_n is the set of all gating networks having inputs x_1, x_2, \dots, x_n and their complements and having a single output. Because of the above 1-1 correspondence, we will use the terms “gating network” and “logical expression” interchangeably.

► (3.4) EXAMPLES

1) The logical expression $(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge x_3)$ corresponds to the gating network shown in Figure 3.4.

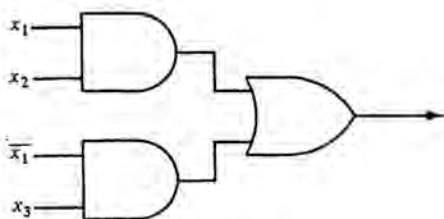


Figure 3.4

2) The gating network shown in Figure 3.5 corresponds to the logical expression $(\bar{x}_1 \vee x_2) \wedge (x_1 \vee (x_2 \wedge x_3))$. ◀

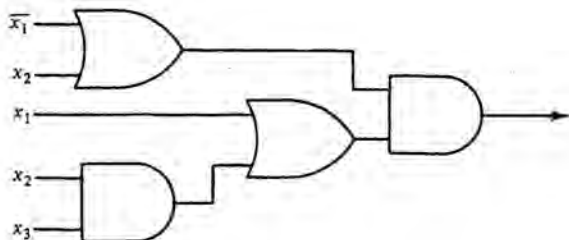


Figure 3.5

As we noted in §1, the set of all logical expressions in x_1, \dots, x_n and their complements is not a Boolean algebra. However, we can put an equivalence relation \equiv on \mathcal{E}_n such that the set of equivalence classes will form a Boolean algebra. For each gating network in \mathcal{E}_n we could consider a table similar to the “closure” tables in §1. Instead, we observe that such a table is equivalent to giving a function from \mathbf{B}^n to \mathbf{B} , where the last column of the table gives the value of the function on the n -tuple given by the first n columns in that row. We will say that two gating networks in \mathcal{E}_n are equivalent if and only if they determine the same function from \mathbf{B}^n to \mathbf{B} . (This is the same as saying that the two circuits have the same closure table.)

(3.5) DEFINITION. A function $f : \mathbf{B}^n \rightarrow \mathbf{B}$ is called a *switching function*. Given a gating network $E \in \mathcal{E}_n$, we will denote by f_E the switching function determined by E . We will denote the set of all switching functions from \mathbf{B}^n to \mathbf{B} by \mathcal{F}_n .

With this notation, given two gating networks E_1 and E_2 , we have $E_1 \equiv E_2$ if and only if $f_{E_1} = f_{E_2}$. We note that a switching function is *any* function from \mathbf{B}^n to \mathbf{B} — it need not be a homomorphism between these two Boolean algebras.

► **(3.6) EXAMPLE.** The switching function $f : \mathbf{B}^2 \rightarrow \mathbf{B}$ given by

$$f(x_1, x_2) = (\bar{x}_1 \wedge \bar{x}_2) \vee x_1$$

is not a homomorphism of Boolean algebras (since, for instance, $f(0, 0) = 1$). This function is the switching function f_{E_1} determined by the gating network E_1 shown in Figure 3.6.

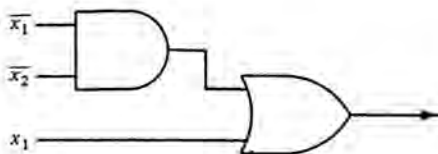


Figure 3.6

The (closure or truth) table that completely describes f_{E_1} is shown in Table (3.7).

(3.7) TABLE

x_1	x_2	$(\bar{x}_1 \wedge \bar{x}_2) \vee x_1$
0	0	1
0	1	0
1	0	1
1	1	1

The function f_{E_1} is equal to the function f_{E_2} determined by the gating network E_2 shown in Figure 3.7. These two functions are equal (and thus these two gating networks are equivalent) because $(\bar{x}_1 \wedge \bar{x}_2) \vee x_1$ is logically equivalent to $(\bar{x}_1 \vee x_1) \wedge (\bar{x}_2 \vee x_1)$, which in turn is logically equivalent to $\bar{x}_2 \vee x_1$. Indeed, the truth table for the expression $\bar{x}_2 \vee x_1$ looks just like Table (3.7). ◀

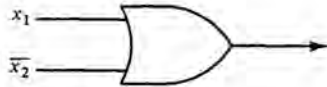


Figure 3.7

We define operations \vee , \wedge and $\bar{}$ on \mathcal{F}_n as follows:

$$(f \vee g)(b_1, \dots, b_n) = f(b_1, \dots, b_n) \vee g(b_1, \dots, b_n)$$

$$(f \wedge g)(b_1, \dots, b_n) = f(b_1, \dots, b_n) \wedge g(b_1, \dots, b_n)$$

$$\bar{f}(b_1, \dots, b_n) = \overline{f(b_1, \dots, b_n)}.$$

We will let $\mathbf{0}$ (respectively, $\mathbf{1}$) denote the switching function that takes the value 0 (respectively, 1) on every n -tuple of \mathbf{B}^n . We leave it to the reader to supply the necessary details to prove that

(3.8) PROPOSITION. $(\mathcal{F}_n, \vee, \wedge, \bar{}, \mathbf{0}, \mathbf{1})$ is a Boolean algebra.

Note that \mathcal{F}_n contains 2^{2^n} elements. This number becomes very large very quickly. For example, for $n = 4$ there are 65,536 switching functions!

As in the previous section, we can put a partial order \leq on \mathcal{F}_n by defining $f \leq g$ if and only if $f \wedge g = f$. We leave a proof of the following lemma as an exercise.

(3.9) LEMMA. $f \leq g$ if and only if for every $(b_1, \dots, b_n) \in \mathbf{B}^n$, if $f(b_1, \dots, b_n) = 1$, then $g(b_1, \dots, b_n) = 1$.

What are the atoms in \mathcal{F}_n ? From the previous lemma, it is not hard to see that the atoms are the switching functions that take the value 1 at precisely one n -tuple in \mathbf{B}^n .

(3.10) DEFINITION. For $i = 0, 1, \dots, 2^n - 1$, if $i = c_1 2^{n-1} + c_2 2^{n-2} + \dots + c_{n-1} 2 + c_n$, where $c_j \in \mathbf{B}$, then let m_i denote the switching function that is 1 on (c_1, c_2, \dots, c_n) and is zero on every other n -tuple in \mathbf{B}^n .

For example, the switching function $m_7 \in \mathcal{F}_4$ takes the value 1 on the 4-tuple $(0, 1, 1, 1)$ and takes the value 0 on every other 4-tuple. Note that m_7 is thus the switching function determined by the network $\bar{x}_1 \wedge x_2 \wedge x_3 \wedge x_4$. From our remarks above, it follows that the atoms in \mathcal{F}_n are precisely $m_0, m_1, \dots, m_{2^n-1}$. The atoms of \mathcal{F}_n are often called *minterms*. Note that if $g \in \mathcal{F}_n$, then by Lemma (3.9) the atom m_i is less than or equal to g if and only if $g(c_1, c_2, \dots, c_n) = 1$, where $c_1 c_2 \dots c_n$ is the binary representation of i .

(3.11) NOTATION. Suppose $g \in \mathcal{F}_n$. We will let $g(i)$ denote the value of the switching function g on the n -tuple (c_1, c_2, \dots, c_n) such that $c_1 c_2 \dots c_n$ is the binary representation of i (i.e., $i = c_1 2^{n-1} + c_2 2^{n-2} + \dots + c_{n-1} 2 + c_n$).

Applying Theorem (2.9), we obtain the following result.

(3.12) PROPOSITION. For any $g \in \mathcal{F}_n$,

$$g = g(0)m_0 \vee g(1)m_1 \vee \dots \vee g(2^n - 1)m_{2^n - 1},$$

and this is the unique representation of g as a sum of atoms of \mathcal{F}_n . (Here we mean that if $g(i) = 1$, then m_i appears in this sum, while if $g(i) = 0$, then m_i does not appear.)

(3.13) NOTATION. If $g = m_{i_1} \vee m_{i_2} \vee \dots \vee m_{i_k}$, then we will sometimes write $g = \sum(i_1, i_2, \dots, i_k)$.

► **(3.14) EXAMPLE.** Suppose $E = (x_1 \vee \bar{x}_2) \wedge x_3$. Then the closure (or truth) table for E is

x_1	x_2	x_3	$(x_1 \vee \bar{x}_2) \wedge x_3$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

We see from the table that $f_E(1) = 1$, $f_E(5) = 1$, and $f_E(7) = 1$. Thus, the representation of f_E as a sum of minterms is $f_E = m_1 \vee m_5 \vee m_7$. We can also write $f_E = \sum(1, 5, 7)$. ◀

In the next section, we will use Proposition (3.12) to obtain the so-called “disjunctive normal form” for a gating network. This normal form is a representative for the equivalence class of the gating network that has a particularly simple form. We will be able to determine whether two gating networks are equivalent by comparing their disjunctive normal forms. This normal form will also be of help in determining a “minimal” network equivalent to a given network.

EXERCISES

3.1. Draw the gating network associated to each of the following logical expressions:

- a) $(x_1 \vee x_2) \wedge x_3$
- b) $(x_1 \vee \bar{x}_2) \vee (\bar{x}_2 \wedge x_3)$
- c) $x_1 \vee (x_2 \wedge x_3) \vee (\bar{x}_2 \vee \bar{x}_3)$
- d) $[(x_1 \vee x_2) \wedge (x_2 \wedge \bar{x}_3)] \vee (\bar{x}_1 \vee x_3)$

† **3.2. a)** Determine the logical expression associated to the gating network E shown in Figure 3.8.

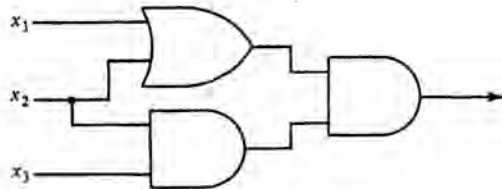


Figure 3.8

- b) Determine the table, as in (3.14), that describes the switching function f_E .
 - c) Find a simpler network to which E is equivalent.
- 3.3. a)** Determine the logical expression associated to the gating network E shown in Figure 3.9.

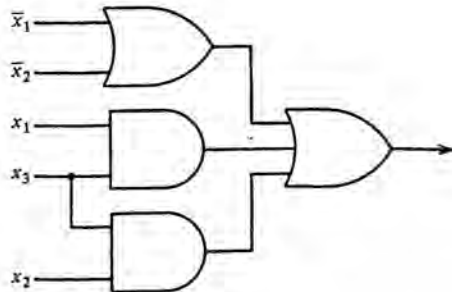


Figure 3.9

- b) Determine the table, as in (3.14), that describes the switching function f_E .

- c) Show that E is equivalent to the network $\bar{x}_1 \vee \bar{x}_2 \vee x_3$.
- † 3.4. For each $i = 0, 1, \dots, 2^n - 1$, determine a gating network (i.e., logical expression) $E_{(i)}$ such that the switching function determined by $E_{(i)}$ is the atom m_i .
- 3.5. a) Show that the Boolean algebra \mathcal{F}_2 is isomorphic to the Boolean algebra of all positive divisors of 210.
b) Draw a Hasse diagram for the Boolean algebra \mathcal{F}_2 .
- † 3.6. For each of the gating networks (logical expressions) in Exercise 3.1 above, write the associated switching function f_E as a sum (join) of atoms of \mathcal{F}_3 .
- † 3.7. Prove Lemma (3.9).
- 3.8. For $i = 0, 1, \dots, 2^n - 1$, let M_i denote the switching function such that $M_i(c_1, \dots, c_n) = 0$, where $i = c_1 2^{n-1} + \dots + c_{n-1} 2 + c_n$, and $M_i(b_1, \dots, b_n) = 1$ for all other $(b_1, \dots, b_n) \in \mathbf{B}^n$. The M_i are called the *maxterms* of \mathcal{F}_n .
a) Show that $\overline{m_i} = M_i$.
b) For any $g \in \mathcal{F}_n$, show that g may be written in a unique way (up to order) as a *conjunction* of maxterms.
- 3.9. This exercise uses material from Chapter III. Define a mapping

$$\begin{aligned}\tilde{\cdot}: S_n &\longrightarrow \text{Perm}(\mathbf{B}^n) \\ \sigma &\longmapsto \tilde{\sigma}\end{aligned}$$

by

$$\tilde{\sigma}(b_1, b_2, \dots, b_n) = (b_{\sigma^{-1}(1)}, b_{\sigma^{-1}(2)}, \dots, b_{\sigma^{-1}(n)}).$$

- a) Show that this mapping is a homomorphism of groups. (Hint: You must see that $(\tau \circ \sigma) = \tilde{\tau} \circ \tilde{\sigma}$. Put $(c_1, \dots, c_n) = \tilde{\sigma}(b_1, \dots, b_n)$ and suppose $\tau^{-1}(i) = j$. Then the i^{th} coordinate of $\tilde{\tau}(c_1, \dots, c_n)$ is $c_{\tau^{-1}(i)}$, which equals $b_{\sigma^{-1}(j)}$. Hence the i^{th} coordinate of $(\tilde{\tau} \circ \tilde{\sigma})(b_1, \dots, b_n)$ is $b_{\sigma^{-1}(\tau^{-1}(i))}$.)
- b) Show that this homomorphism is 1-1. (Hint: Compute the kernel.)
- Thus S_n is isomorphic to a subgroup of $\text{Perm}(\mathbf{B}^n)$. Call this subgroup G . We are now in the situation of III.9 and III.10, and we can define a right action of $G(\cong S_n)$ on $\mathcal{F}_n (= \mathbf{B}^{\mathbf{B}^n})$. Explicitly, if $g \in S_n$ and $f \in \mathcal{F}_n$, then

$$f\sigma(b_1, \dots, b_n) = f(b_{\sigma^{-1}(1)}, \dots, b_{\sigma^{-1}(n)}).$$

The orbits of this action of S_n on \mathcal{F}_n are equivalence classes of gating networks, where two gating networks are equivalent here if they define the same switching function after any possible permutation of the inputs (but one is not allowed here to replace x_i with its complement \bar{x}_i or vice versa).

- c) Find the number of distinct orbits of the above action of S_3 on \mathcal{F}_3 . (Hint: First you must represent S_3 as a subgroup of S_8 by seeing how each $\bar{\sigma}$ permutes the elements of \mathbf{B}^3 . Then write each $\bar{\sigma}$ as a product of disjoint cycles in S_8 . Finally, use Corollary (III.10.4).)

For a more detailed treatment of the material in Exercise 3.9, and of the case when one allows replacement of an input by its complement as well, the reader may see Harrison [14] or Dornhoff and Hohn [10].

4. Disjunctive Normal Form and Prime Implicants

We now will use the results from the previous section to pick out a distinguished, or “canonical,” representative from each equivalence class of gating networks. (Recall that two gating networks are called equivalent if they determine the same switching function.)

(4.1) NOTATION. Given $c_i \in \mathbf{B}$, we will let $x_i^{c_i}$ denote x_i if $c_i = 1$ and \bar{x}_i if $c_i = 0$. Also, we will henceforth denote the conjunction of two expressions simply by juxtaposition; so, for example, we will write x_1x_2 instead of $x_1 \wedge x_2$.

(4.2) DEFINITION. For $i = 0, 1, \dots, 2^n - 1$, put

$$E_{(i)} = x_1^{c_1} x_2^{c_2} \cdots x_n^{c_n}$$

where $c_1c_2 \cdots c_n$ is the binary representation of i .

It is clear from our definitions that we have

(4.3) LEMMA. For each $i = 0, 1, \dots, 2^n - 1$, we have $f_{E_{(i)}} = m_i$; that is, the switching function determined by the expression (or network) $E_{(i)}$ is the minterm m_i of \mathcal{F}_n .

Indeed, $E_{(i)}$ is the “most obvious” gating network representing the switching function m_i .

(4.4) **THEOREM.** Suppose $E \in \mathcal{E}_n$ and $f_E \neq 0$. Then E is equivalent to

$$(4.5) \quad f_E(0)E_{(0)} \vee f_E(1)E_{(1)} \vee \cdots \vee f_E(2^n - 1)E_{(2^n - 1)}.$$

(Here, if $f_E(i) = 1$, then $E_{(i)}$ appears in the sum and if $f_E(i) = 0$, then $E_{(i)}$ does not appear.)

PROOF. This follows immediately from the previous lemma and Proposition (3.12). ■

(4.6) **DEFINITION.** The expression (4.5) is called the *disjunctive normal form (d.n.f.) representing f_E* or the *disjunctive normal form for E* . If $f_E = 0$, then we will call $x_1\bar{x}_1$ the disjunctive normal form for E .

The disjunctive normal form is a “sum of products” such that for each $i = 1, 2, \dots, n$, either x_i or \bar{x}_i occurs in each product term. We have shown that a given expression is equivalent to a unique disjunctive normal form. Thus, for example, *two expressions are equivalent if and only if they have the same disjunctive normal forms.*

► (4.7) EXAMPLES

1) Note that there are as many possible d.n.f.’s for expressions in \mathcal{E}_n as there are switching functions in \mathcal{F}_n ; so there are 2^{2^n} possible d.n.f.’s for expressions in \mathcal{E}_n . The 16 possible d.n.f.’s for expressions in \mathcal{E}_2 are:

$$\begin{aligned} x_1\bar{x}_1, E_{(0)} = \bar{x}_1\bar{x}_2, E_{(1)} = \bar{x}_1x_2, E_{(2)} = x_1\bar{x}_2, E_{(3)} = x_2x_2, \\ E_{(0)} \vee E_{(1)}, E_{(0)} \vee E_{(2)}, E_{(0)} \vee E_{(3)}, E_{(1)} \vee E_{(2)}, E_{(1)} \vee E_{(3)}, \\ E_{(2)} \vee E_{(3)}, E_{(0)} \vee E_{(1)} \vee E_{(2)}, E_{(0)} \vee E_{(1)} \vee E_{(3)}, E_{(0)} \vee E_{(2)} \vee E_{(3)}, \\ E_{(1)} \vee E_{(2)} \vee E_{(3)}, E_{(0)} \vee E_{(1)} \vee E_{(2)} \vee E_{(3)}. \end{aligned}$$

2) Consider the expression

$$E = (x_1 \vee \bar{x}_2)(\bar{x}_1 \vee x_2).$$

Then it is easy to compute that

$$\begin{aligned} f_E(0) &= f_E(0, 0) = (0 \vee 1) \wedge (1 \vee 0) = 1 \wedge 1 = 1 \\ f_E(1) &= f_E(0, 1) = (0 \vee 0) \wedge (1 \vee 0) = 0 \wedge 1 = 0 \\ f_E(2) &= f_E(1, 0) = (1 \vee 1) \wedge (0 \vee 0) = 1 \wedge 0 = 0 \\ f_E(3) &= f_E(1, 1) = (1 \vee 0) \wedge (0 \vee 1) = 1 \wedge 1 = 1. \end{aligned} \quad 300$$

Therefore, applying Theorem (4.4), we see that the d.n.f. for E is

$$(\bar{x}_1\bar{x}_2) \vee (x_1x_2).$$

We could have obtained the d.n.f. algebraically by using properties of a Boolean algebra to see that

$$\begin{aligned} E &\equiv ((x_1 \vee \bar{x}_2)\bar{x}_1) \vee ((x_1 \vee \bar{x}_2)x_2) \\ &\equiv ((x_1\bar{x}_1) \vee (\bar{x}_2\bar{x}_1)) \vee ((x_1x_2) \vee (\bar{x}_2x_2)) \\ &\equiv (\bar{x}_2\bar{x}_1) \vee (x_1x_2) \\ &\equiv (\bar{x}_1\bar{x}_2) \vee (x_1x_2). \blacktriangleleft \end{aligned}$$

As the reader may see from the previous example, there are two strategies we may follow to find the d.n.f. of an expression $E \in \mathcal{E}_n$. Namely, we could compute $f_E(i)$ for $i = 0, 1, \dots, 2^n - 1$, which would be quite laborious if n is at all large; or we can use the algebraic properties of a Boolean algebra to obtain an expression equivalent to E that has the correct form to be a d.n.f. To describe this latter process in some more detail, if E is not a "contradiction" (i.e., if $f_E \neq 0$), then by applying, as necessary, the Distributive Laws, the Idempotent Law (Exercise 1.1), DeMorgan's Laws (Exercise 1.8) and the Absorption Law (Exercise 1.3), we may obtain an expression that is equivalent to E and that is a "sum of products." This will not be the required d.n.f. unless each product term has either x_i or \bar{x}_i , for each $i = 1, 2, \dots, n$, as a factor. In order to obtain the d.n.f. we can insert, for each $i = 1, 2, \dots, n$, the term $(x_i \vee \bar{x}_i)$ into every product term that does not contain either x_i or \bar{x}_i and then expand again using the Distributive, Idempotent, and Absorption Laws, as needed, to obtain a sum of products that now must be the d.n.f. We illustrate this with another example.

► (4.8) **EXAMPLE.** Let

$$E = ((\overline{x_1 \vee \bar{x}_2}) \vee (x_2x_1))(x_2 \vee \bar{x}_3) \in \mathcal{E}_3.$$

An application of DeMorgan's Laws gives

$$E \equiv ((\bar{x}_1x_2) \vee (x_2x_1))(x_2 \vee \bar{x}_3).$$

Now applying the Distributive Laws we obtain:

$$\begin{aligned} E &\equiv ((\bar{x}_1x_2)(x_2 \vee \bar{x}_3)) \vee ((x_2x_1)(x_2 \vee \bar{x}_3)) \\ &\equiv ((\bar{x}_1x_2)x_2) \vee ((\bar{x}_1x_2)\bar{x}_3) \vee ((x_2x_1)x_2) \vee ((x_2x_1)\bar{x}_3). \end{aligned}$$

Applying the Idempotent and Commutative Laws then yields the sum of products expression

$$E \equiv (\bar{x}_1 x_2) \vee (\bar{x}_1 x_2 \bar{x}_3) \vee (x_1 x_2) \vee (x_1 x_2 \bar{x}_3).$$

Note that this is *not* the d.n.f. for E because the first and third product terms lack a factor of either x_3 or \bar{x}_3 . We can insert $x_3 \vee \bar{x}_3$ into these two terms and expand by Distributivity to obtain finally the d.n.f. of E :

$$E \equiv (\bar{x}_1 x_2 x_3) \vee (\bar{x}_1 x_2 \bar{x}_3) \vee (x_1 x_2 x_3) \vee (x_1 x_2 \bar{x}_3).$$

(Note that we also used the Idempotent Law in this last step.) ◀

Our main goal in this section and the next is to obtain a “minimal” expression equivalent to a given expression E . It should be clear that in most cases the d.n.f. will not be such a minimal expression. For example, the expression E in Example (4.8) is actually equivalent to x_2 ! However, the d.n.f. gives us a “standardized” form on which we can apply an algorithm (due to Quine and McCluskey) to obtain a minimal equivalent expression. The first step in this algorithm is to identify the so-called prime implicants of an expression E .

(4.9) DEFINITIONS. Suppose $Q \in \mathcal{E}_n$ is a product of m literals and suppose E is any expression in \mathcal{E}_n . Then we call Q an *implicant of degree m* of E if $f_Q \leq f_E$. If $P \in \mathcal{E}_n$ is an implicant of E and if no proper subproduct of P (i.e., a product of a proper subset of the literals occurring in P) is an implicant of E , then P is called a *prime implicant* of E .

This terminology comes from logic. If $f_Q \leq f_E$, then the logical expression E is true whenever the expression Q is true; i.e., Q implies E .

- **(4.10) EXAMPLE.** Consider the expression $E = (x_1 \vee \bar{x}_2)(\bar{x}_1 \vee x_2)$ of Example (4.7.2). We claim that $P_1 = \bar{x}_1 \bar{x}_2$ and $P_2 = x_1 x_2$ are prime implicants of E . To see that P_1 is an implicant, note that $f_{P_1}(i) = 1$ only when $i = 0$, and we computed in Example (4.7.2) that $f_E(0) = 1$. Therefore, $f_{P_1} \leq f_E$. We have that P_1 is a *prime* implicant since neither \bar{x}_1 nor \bar{x}_2 (the proper subproducts of P_1) is an implicant of E . (Since $f_{\bar{x}_1}(1) = 1$ while $f_E(1) = 0$ and $f_{\bar{x}_2}(2) = 1$ while $f_E(2) = 0$.) Similar arguments show that P_2 is a prime implicant of E . ◀

From the definitions of d.n.f. (i.e., (4.5)) and implicant, we have the following:

(4.11) LEMMA. The product terms in the d.n.f. of $E \in \mathcal{E}_n$ are exactly the implicants of degree n of E .

(4.12) LEMMA. An expression E is equivalent to the sum of its prime implicants.

PROOF. Suppose P_1, P_2, \dots, P_k are the prime implicants of E and put $E' = P_1 \vee P_2 \vee \dots \vee P_k$. If $f_{E'}(i) = 1$ for some i , $0 \leq i \leq 2^n - 1$, then $f_{P_j}(i) = 1$ for some j , $1 \leq j \leq k$. Since P_j is an implicant of E , we must then have $f_E(i) = 1$. Thus $f_{E'} \leq f_E$.

Now suppose $f_E(i) = 1$. Then there is a product term $E_{(j)}$ in the d.n.f. of E such that $f_{E_{(j)}}(i) = 1$. The switching function determined by any subproduct of $E_{(j)}$ will then also take the value 1 at i and since $E_{(j)}$ is an implicant of E , some subproduct of $E_{(j)}$ must be a prime implicant of E (namely, a subproduct of $E_{(j)}$ of least degree that is still an implicant of E). Thus there is an l , $1 \leq l \leq k$, such that $f_{P_l}(i) = 1$. Hence we have that $f_{E'}(i) = 1$, and $f_E \leq f_{E'}$, completing the proof. ■

The idea now is to identify the prime implicants of E and then to consider a minimal sum of these prime implicants that is still equivalent to E .

The key observation in devising an algorithm to find the prime implicants is: If Q is an implicant of degree $n-1$ of $E \in \mathcal{E}_n$ and if x_i is the literal that is missing from Q , then $Q \wedge x_i$ and $Q \wedge \bar{x}_i$ are implicants of degree n of E , and so both of these must occur as product terms in the d.n.f. of E . Then we repeat this argument for implicants of degree $n-2, n-3, \dots, 1$. We omit a formal proof that the following algorithm indeed finds the prime implicants of an expression.

(4.13) ALGORITHM FOR FINDING PRIME IMPLICANTS. Suppose $E \in \mathcal{E}_n$ and suppose the d.n.f. of E is known. We will construct a table as follows.

- 1) In the first column of the table enter the binary representation of each integer i such that $E_{(i)}$ occurs in the d.n.f. of E (equivalently, such that $f_E(i) = 1$). List these integers in increasing order.
- 2) Begin with the first integer in this column and compare it with every other integer in the column. If the first integer and another integer in this column differ in exactly one place, then put an entry in the second column having a dash in this one place and having the same digits as these two integers for its other places. Place a check mark next to each of the integers in a pair which differ in exactly one digit. For example, if the first integer is 000, then look down the column for 001, 010, and

100. If, say, 010 occurs in this column, then enter 0-0 in the second column and place check marks next to 000 and 010.
- 3) Now proceed to the second integer in the first column and compare it with each integer below it. For each such pair that differ in exactly one place, put an entry in the second column as in step (2) and place check marks next to the integers in your pair. Checked entries may be used more than once to produce a new entry in the next column. Also, check marks should be placed next to such a pair even if they produce an entry that already appears in the second column. Repeat this process for each subsequent integer in the first column.
 - 4) Continue this process with the second column (thus possibly forming a third column) and with each subsequent column until no new columns are generated.
 - 5) The *unchecked* entries correspond to the prime implicants of E .

► (4.14) EXAMPLES

- 1) Consider the network

$$E = ((x_1 \vee \bar{x}_2) \vee (x_2 x_1))(x_2 \vee \bar{x}_3)$$

of Example (4.8). We showed that the d.n.f. of E is

$$(\bar{x}_1 x_2 x_3) \vee (\bar{x}_1 x_2 \bar{x}_3) \vee (x_1 x_2 x_3) \vee (x_1 x_2 \bar{x}_3).$$

Performing the above algorithm, we obtain the following table:

010 ✓	-10 ✓ (1st and 2nd)	-1-
110 ✓	01- ✓ (1st and 3rd)	
011 ✓	11- ✓ (2nd and 4th)	
111 ✓	-11 ✓ (3rd and 4th)	

Note that 01- and 11- are checked even though they produce the entry -1- in the third column, which was already produced by -10 and -11. Since the only unchecked entry is -1-, the only prime implicant of E is x_2 . From Lemma (4.12), it follows that E is equivalent to x_2 and obviously this is the minimal network equivalent to E .

- 2) Suppose $E \in \mathcal{E}_3$ and the d.n.f. of E is

$$(\bar{x}_1 \bar{x}_2 \bar{x}_3) \vee (\bar{x}_1 \bar{x}_2 x_3) \vee (\bar{x}_1 x_2 x_3) \vee (x_1 \bar{x}_2 \bar{x}_3) \vee (x_1 x_2 \bar{x}_3) \vee (x_1 x_2 x_3).$$

Applying the algorithm for finding prime implicants, we obtain the following table:

000 ✓	00-
001 ✓	-00
011 ✓	0-1
100 ✓	-11
110 ✓	1-0
111 ✓	11-

It follows that the prime implicants of E are

$$\bar{x}_1\bar{x}_2, \bar{x}_2\bar{x}_3, \bar{x}_1x_3, x_2x_3, x_1\bar{x}_3, x_1x_2.$$

By Lemma (4.12), E is equivalent to the sum of all six of these prime implicants. But note that E is also equivalent to each of the following sums of prime implicants:

$$\begin{aligned} &(\bar{x}_1x_3) \vee (\bar{x}_2\bar{x}_3) \vee (x_1\bar{x}_3) \vee (x_2x_3), \\ &\text{or } (\bar{x}_1x_3) \vee (x_1x_2) \vee (\bar{x}_2\bar{x}_3), \\ &\text{or } (\bar{x}_1\bar{x}_2) \vee (x_1\bar{x}_3) \vee (x_2x_3). \end{aligned}$$

(One way to see that E is equivalent to each of these three sums is to see that each of these expressions has the same d.n.f. as E .) ◀

This example makes it clear that we have more work to do after finding prime implicants if we want to find a “minimal” network equivalent to E . We will describe this additional work in the next section. We note that another method for finding prime implicants has been developed by B. Reusch (see Dornhoff and Hohn [10]). Also, some readers may have already seen the method of Karnaugh maps for finding a minimal equivalent network. (For a description of Karnaugh maps, see [24] or [21].) Karnaugh maps are probably easier to use than the Quine-McCluskey algorithm when dealing with a small number of literals. However, the Quine-McCluskey algorithm has the advantages that it is applicable for any number of literals and that it is adaptable to a computer.

EXERCISES

- † 4.1. For each of the following expressions in \mathcal{E}_2 , find the d.n.f. and the prime implicants.
- x_2
 - $x_1 \vee (\bar{x}_1x_2)$
 - $(x_1 \vee x_2)(\bar{x}_1 \vee \bar{x}_2)$

$$d) \overline{(x_1 \bar{x}_2)} \vee x_2$$

- † 4.2. For each of the following pairs of expressions in \mathcal{E}_2 , determine if the two given expressions are equivalent. (Recall that two expressions are equivalent if and only if they have the same d.n.f.; or you may use truth tables.)

$$a) \bar{x}_1 \bar{x}_2, (\bar{x}_1 \vee x_2) \overline{(x_1 \vee x_2)}$$

$$b) x_1, \overline{(\bar{x}_1 \vee x_2)} \bar{x}_2$$

$$c) x_1 \bar{x}_2, x_1 (\bar{x}_2 \vee \bar{x}_1) \vee \bar{x}_2 (x_1 \vee x_2)$$

- 4.3. For each of the following expressions in \mathcal{E}_3 , find the d.n.f. and the prime implicants.

$$a) (x_1 \bar{x}_2) \vee (x_1 x_3) \vee (\bar{x}_2 x_3)$$

$$b) (x_1 x_2 x_3) \vee (x_1 \bar{x}_2) \vee (x_1 x_3)$$

$$c) E_{(6)} \vee E_{(7)} \vee (x_1 x_3)$$

$$d) E_{(0)} \vee (x_2 \bar{x}_3) \vee (x_1 x_2)$$

$$e) (\bar{x}_1 \bar{x}_3) \vee (x_2 x_3) \vee (x_1 \bar{x}_2)$$

- † 4.4. For each of the following expressions in \mathcal{E}_4 , find the prime implicants.

$$a) E_{(12)} \vee E_{(13)} \vee E_{(15)}$$

$$b) E_{(1)} \vee E_{(3)} \vee E_{(9)} \vee E_{(11)} \vee E_{(12)} \vee E_{(13)}$$

$$c) E_{(3)} \vee E_{(7)} \vee E_{(8)} \vee E_{(11)} \vee E_{(14)} \vee E_{(15)}$$

$$d) E_{(2)} \vee E_{(4)} \vee E_{(7)} \vee E_{(10)} \vee E_{(12)} \vee E_{(14)} \vee E_{(15)}$$

$$e) E_{(0)} \vee E_{(1)} \vee E_{(8)} \vee E_{(9)} \vee E_{(11)} \vee E_{(13)} \vee E_{(15)}$$

- 4.5. Suppose $E \in \mathcal{E}_n$ and $f_E \neq 1$. Show that E is equivalent to a “product of sums” where each sum contains either x_i or \bar{x}_i , for $i = 1, 2, \dots, n$. This representative for f_E is called the *conjunctive normal form* representing f_E or the c.n.f. for E . (Hint: Consider the d.n.f. of \bar{E} .)

- † 4.6. Find the c.n.f. (see Exercise 4.5 above) of each of the expressions in Exercise 4.3 above.

5. Minimization

In this section, we present the remainder of the Quine-McCluskey algorithm for finding a “minimal” network equivalent to a given one. But first, we need to discuss what “minimal” will mean here. Given a gating network E , one would ideally like to find an equivalent network with the fewest occurrences of literals (gates). Unfortunately, there is no efficient algorithm for doing this. The Quine-McCluskey algorithm finds a “sum of products”

expression E' that is equivalent to E and that is a minimal sum of products expression equivalent to E , in the sense that if E'' is another sum of products expression equivalent to E , then either E'' has more product terms than E' or E'' has the same number of product terms as E' and at least as many occurrences of literals as E' . Note that there may be more than one minimal sum of products expression equivalent to E (see Example (5.7.1) below). Sometimes, as in Example (4.14.1), it will be clear that the algorithm has indeed produced the minimal expression (not just the minimal sum of products expression) equivalent to E . But note that the minimal sum of products expression equivalent to $x_1(x_2 \vee x_3)$ is $(x_1x_2) \vee (x_1x_3)$, which has one more gate than the original network.

(5.1) DEFINITION. We will say that a product of literals P covers a product of literals Q if P is a subproduct of Q .

Suppose $P \in \mathcal{E}_n$ is a product of literals and $E_{(i)}$ is a minterm as in Definition (4.2). Then note that P covers $E_{(i)}$ if and only if $f_P(i) = 1$. For example, the products of literals that cover the minterm $E_5 = x_1\bar{x}_2x_3 \in \mathcal{E}_3$ are $x_1, \bar{x}_2, x_3, x_1\bar{x}_2, x_1x_3, \bar{x}_2x_3$, and $x_1\bar{x}_2x_3$.

(5.2) LEMMA. Suppose P_1, P_2, \dots, P_k are prime implicants of an expression E . Then

$$E \equiv P_1 \vee P_2 \vee \dots \vee P_k$$

if and only if each of the products $E_{(i)}$ in the d.n.f. of E is covered by at least one of the $P_j, j = 1, \dots, k$.

PROOF. Suppose $E \equiv P_1 \vee \dots \vee P_k$. If $E_{(i)}$ is a minterm in the d.n.f. of E , then $f_E(i) = 1$. But this implies that $f_{P_j}(i) = 1$ for some $j, 1 \leq j \leq k$ and this means that P_j covers $E_{(i)}$.

Conversely, assume that each minterm in the d.n.f. of E is covered by a P_j and put

$$E' = P_1 \vee P_2 \vee \dots \vee P_k.$$

If $f_E(i) = 1$, then $f_{P_j}(i) = 1$, where P_j covers $E_{(i)}$, so we have $f_{E'}(i) = 1$. If $f_E(i) = 0$, then $f_{P_j}(i) = 0$ for $j = 1, 2, \dots, k$ since each P_j is a prime implicant of E . Hence in this case we have $f_{E'}(i) = 0$, thus showing that $f_E = f_{E'}$. ■

(5.3) DEFINITIONS. A prime implicant P of E is called *essential* if there exists a product term $E_{(i)}$ in the d.n.f. of E such that P is the only prime implicant of E that covers $E_{(i)}$. The sum of all the essential prime implicants of E is called the *core* of E .

It follows from Lemma (5.2) that any minimal representation of E as a sum of products must contain the essential prime implicants of E as summands. Therefore, we must see how to determine the core of E .

We assume that $E \in \mathcal{E}_n$ and that the d.n.f. of E and the prime implicants of E , call them P_1, P_2, \dots, P_r , have been found (say by the algorithm in the previous section). We now form a table showing which prime implicants cover which terms in the d.n.f. of E .

(5.4) PROCEDURE FOR FINDING THE CORE

- 1) Construct a table with one row for each prime implicant of E and one column for each of the product terms in the d.n.f. of E .
- 2) Put an "x" in the row corresponding to P_i and the column corresponding to $E_{(j)}$ if and only if P_i covers $E_{(j)}$.
- 3) Circle each "x" that is the only "x" in its column. A prime implicant P_i is essential if and only if there is a circled "x" in the row corresponding to P_i .

► (5.5) EXAMPLES

1) Suppose $E \in \mathcal{E}_4$ and $f_E = \sum(0, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15)$. We leave it to the reader to verify that the prime implicants of E are

$$\bar{x}_1\bar{x}_3\bar{x}_4, \bar{x}_2\bar{x}_3\bar{x}_4, \bar{x}_1x_2, x_1\bar{x}_2, x_1x_3, \text{ and } x_2x_3.$$

We obtain the following covering table, where we have written j , in binary representation, instead of $E_{(j)}$, and where we have represented each prime implicant using 0's, 1's, and dashes as in the previous section.

	0000	0100	0101	0110	0111	1000	1001	1010	1011	1110	1111
0-00	x	x									
-000	x					x					
01--		x	⊗	x	x						
10--						x	⊗	x	x		
1-1-								x	x	x	x
-11-				x	x					x	x

Thus the core of E is $(\bar{x}_1x_2) \vee (x_1\bar{x}_2)$.

2) As in Example (4.14.2), we suppose $E \in \mathcal{E}_3$ and the d.n.f. of E is

$$E_{(0)} \vee E_{(1)} \vee E_{(3)} \vee E_{(4)} \vee E_{(6)} \vee E_{(7)}.$$

We saw in the last section that the prime implicants of E are 308

$$\bar{x}_1\bar{x}_2, \bar{x}_2\bar{x}_3, \bar{x}_1x_3, x_2x_3, x_1\bar{x}_3, \text{ and } x_1x_2.$$

We obtain the following covering table:

	000	001	011	100	110	111
00-	x	x				
-00	x			x		
0-1		x	x			
-11			x			x
1-0				x	x	
11-					x	x

Since there is no column with a single "x" in it, this network has no core.

3) Suppose $E \in \mathcal{E}_4$ and $f_E = \sum(0, 6, 7, 8, 9, 10, 11, 14, 15)$. Then the reader may check that the prime implicants of E are

$$\bar{x}_2\bar{x}_3\bar{x}_4, x_2x_3, x_1\bar{x}_2, \text{ and } x_1x_3.$$

We obtain the following covering table:

	0000	0110	0111	1000	1001	1010	1011	1110	1111
-000	⊗			x					
-11-		⊗	⊗					x	x
10--				x	⊗	x	x		
1-1-						x	x	x	x

Thus the core of E is

$$(\bar{x}_2\bar{x}_3\bar{x}_4) \vee (x_2x_3) \vee (x_1\bar{x}_2).$$

Since every minterm in the d.n.f. of E is covered by an essential prime implicant, we see that, in this example, E is equivalent to its core and thus (in this example) the core of E is the minimal sum of products equivalent to E . ◀

If every minterm in the d.n.f. of an expression E is covered by an essential prime implicant (as in Example (5.5.3)), then the algorithm stops — the core is then the minimal sum of products equivalent to E . In the general case, if some minterm in the d.n.f. of E is not covered by any essential prime implicant, then E is not equivalent to its core. We must then see which nonessential prime implicants can be joined to the core to ensure that every term in the d.n.f. of E is covered, and we wish to do this in a "minimal" way.

► (5.6) **EXAMPLE.** Suppose we have found the core of an expression E . Further suppose that the minterms $E_{(j)}$ and $E_{(k)}$ in the d.n.f. of E are not

covered by any essential prime implicant. We know from Lemma (4.12) that some prime implicants of E must cover $E_{(j)}$ and $E_{(k)}$. Suppose that P_1 and P_2 are prime implicants of E that cover $E_{(j)}$ and that Q_1 and Q_2 are prime implicants of E that cover $E_{(k)}$. Let C denote the core of E . Then, by Lemma (5.2), E is equivalent to the following sums of products:

$$C \vee P_1 \vee Q_1, C \vee P_1 \vee Q_2, C \vee P_2 \vee Q_1, \text{ and } C \vee P_2 \vee Q_2.$$

Notice that if we put

$$(5.7) \quad D = (P_1 \vee P_2)(Q_1 \vee Q_2) \equiv P_1Q_1 \vee P_1Q_2 \vee P_2Q_1 \vee P_2Q_2,$$

then the "sum of products" expressions that are equivalent to E are equal to the sum of the core of E with each of the prime implicants in any product term in the representation of D as a sum of products of prime implicants (i.e. the far right side of (5.7)). The expression D thus helps us to see which prime implicants must be joined to the core in order to cover every minterm in the d.n.f. of E . ◀

To determine which prime implicants of E must be joined to the core, we consider a subtable of the covering relationships table.

(5.8) QUINE-McCLUSKEY ALGORITHM (CONCLUSION). We assume that an expression E has been given and that the d.n.f. and prime implicants of E have been determined. We also assume that the covering table has been constructed as in (5.4), that the core has been determined and that not every minterm in the d.n.f. of E is covered by an essential prime implicant.

- 1) For each row in the covering table that has a circled "x," put a circle about every "x" in that row.
- 2) Form a new table by deleting from the covering table every row and every column that contains a circled "x."
- 3) It may be possible to shorten this new table even more in one of the following two ways:
 - a) if the column corresponding to E_j has an "x" in every row in which the column corresponding to E_k has an "x," then the column corresponding to E_j can be eliminated, since any prime implicant that covers E_k will then also cover E_j .
 - b) if the row corresponding to a prime implicant P_i has an "x" in every column in which the row corresponding to P_j has an "x" and if P_i contains strictly fewer literals than P_j , then the row corresponding to P_j may be eliminated.

- 4) Now, for each column form the sum of the prime implicants that cover that term in the d.n.f. (i.e., which have an "x" in that column), and form the product of these sums (as in (5.7)). Expand this expression, applying the Distributive, Absorptive, and Idempotent Laws on the prime implicants, as necessary, to get an equivalent expression, call it D , that is a sum of products of prime implicants. (Note: Do not combine the literals in two different prime implicants in following this procedure. For example, if two of the prime implicants involved here are x_1x_2 and \bar{x}_2x_3 , then the product of these two should be written as $(x_1x_2)(\bar{x}_2x_3)$ and one should not use the fact that $x_2\bar{x}_2 \equiv 0$.)
- 5) The candidates for the minimal sums of products equivalent to E are now obtained by taking the sum of the core of E with each of the prime implicants in any product term of D . So, if D is the sum of k product terms (each term being a product of implicants), then there will be k candidates for minimal sums of products equivalent to E . The minimal sum of products expressions equivalent to E are those candidates that have the fewest number of literals.

We will omit a formal proof of Algorithm (5.8).

► (5.9) EXAMPLES

1) We consider the expression E of Example (5.5.1). After performing steps 1 and 2 of Algorithm (5.8), we obtain the following table:

	0000	1110	1111
0-00	x		
-000	x		
1-1-		x	x
-11-		x	x

By step 3a of Algorithm (5.8), we may delete the last (or the middle) column of this table, which leaves us with

	0000	1110
0-00	x	
-000	x	
1-1-		x
-11-		x

We now form the product of $(\bar{x}_1\bar{x}_3\bar{x}_4) \vee (\bar{x}_2\bar{x}_3\bar{x}_4)$ and $(x_1x_3) \vee (x_2x_3)$. Expanding this product, we obtain the equivalent sum of products

$$D = (\bar{x}_1\bar{x}_3\bar{x}_4)(x_1x_3) \vee (\bar{x}_1\bar{x}_3\bar{x}_4)(x_2x_3) \vee (\bar{x}_2\bar{x}_3\bar{x}_4)(x_1x_3) \vee (\bar{x}_2\bar{x}_3\bar{x}_4)(x_2x_3).$$

There are then four candidates for minimal sums of products equivalent to E . Recall that the core of E was found to be $(\bar{x}_1x_2) \vee (x_1\bar{x}_2)$. The candidates for minimal sums of products expressions equivalent to E are then

$$\begin{aligned} &(\bar{x}_1x_2) \vee (x_1\bar{x}_2) \vee (\bar{x}_1\bar{x}_3\bar{x}_4) \vee (x_1x_3) \\ &(\bar{x}_1x_2) \vee (x_1\bar{x}_2) \vee (\bar{x}_1\bar{x}_3\bar{x}_4) \vee (x_2x_3) \\ &(\bar{x}_1x_2) \vee (x_1\bar{x}_2) \vee (\bar{x}_2\bar{x}_3\bar{x}_4) \vee (x_1x_3) \\ &(\bar{x}_1x_2) \vee (x_1\bar{x}_2) \vee (\bar{x}_2\bar{x}_3\bar{x}_4) \vee (x_2x_3), \end{aligned}$$

and since each of these candidates contains 9 literals, each is a minimal sum of products equivalent to E .

2) We consider the expression E of Example (5.5.2). The covering table has no circled "x" and steps 3a and 3b of Algorithm (5.8) do not apply, so we may not eliminate any columns or rows from the covering table. Following the steps in Algorithm (5.8), we would then form the product of the sums $\bar{x}_1\bar{x}_2 \vee \bar{x}_2\bar{x}_3$, $\bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3$, $\bar{x}_1x_3 \vee x_2x_3$, $\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_3$, $x_1\bar{x}_3 \vee x_1x_2$, and $x_2x_3 \vee x_1x_2$. Expanding this product into an equivalent sum of products would yield a sum of many products of prime implicants. However, if we keep in mind that our objective is to cover all terms of the d.n.f. as efficiently as possible, then it is apparent from the covering table that E is equivalent to either of the following two minimal sums of products:

$$\bar{x}_1\bar{x}_2 \vee x_2x_3 \vee x_1\bar{x}_3 \text{ and } \bar{x}_2\bar{x}_3 \vee \bar{x}_1x_3 \vee x_1x_2.$$

One could also see this by computing the products above and applying the Absorption Law as much as possible, but sometimes it is much better to keep in mind your objective and think than to follow blindly the steps of an algorithm. ◀

Sometimes one wishes to minimize a network, and one has the additional knowledge that certain conditions either cannot occur or have no effect on the operation of the network in question. Such conditions are usually called "*don't care conditions*." To use the Quine-McCluskey algorithm to minimize a network E subject to the don't care conditions D_1, D_2, \dots, D_k , one finds the prime implicants of $E \vee D_1 \vee \dots \vee D_k$ and forms the covering table with these prime implicants as rows and with the terms of the d.n.f. of E (not the terms of the d.n.f. of $E \vee D_1 \vee \dots \vee D_k$) as columns. The rest of the algorithm proceeds as before.

- (5.10) **EXAMPLE.** Suppose E is the network $x_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$. Suppose one is given a positive integer m and that the input x_1 is 1 if and only if $2|m$, the input x_2 is 1 if and only if $6|m$, and the input x_3 is 1 if and only

if $3|m$. Then it is easy to see that E outputs a 1 if and only if $3|m$, so E is equivalent to x_3 . However, according to our algorithm for minimization, E is already a minimal sum of products. By the definition of the inputs, it is easy to see that the conditions $D_1 = \bar{x}_1x_2x_3$ and $D_2 = x_1\bar{x}_2x_3$ cannot possibly occur. To minimize E subject to the don't care condition D_1 and D_2 , we find the prime implicants of $E \vee D_1 \vee D_2$, and it is easy to determine that x_3 is the only prime implicant.

We note that one must sometimes choose don't care conditions wisely. For instance, in this example, the condition $D_3 = x_1x_2\bar{x}_3$ also cannot occur, but if we add this condition to the previous ones, we would find that the prime implicants of $E \vee D_1 \vee D_2 \vee D_3$ are x_3 and x_1x_2 . ◀

EXERCISES

- 5.1. Find the minimal sum of products expressions equivalent to each of the expressions in Exercise 4.1.
- † 5.2. Find the minimal sum of products expressions equivalent to each of the expressions in Exercise 4.3. For each expression, sketch the original network and one of the minimal sums of products equivalent to it.
- † 5.3. Find the minimal sum of products expressions equivalent to each of the expressions in Exercise 4.4.
- † 5.4. a) Describe how to find the minimal *product of sums* expressions equivalent to a given network.
b) Find the minimal product of sums expressions equivalent to each of the expressions in Exercise 4.1.
- † 5.5. Find the minimal sum of products expressions equivalent to each of the expressions in Exercise 4.3 subject to the don't care conditions $x_1\bar{x}_3$ and $x_2\bar{x}_3$.

CHAPTER VII

Graph Theory

Many mathematical or real-life problems may be modeled by a graph. Roughly speaking, a graph consists of points and line segments joining some of the points. Examples include a map showing cities as points and a line segment from one city to another if there is a flight between those cities, a diagram of an electrical network, and a representation of a molecule with the atoms represented by points and the bonds by line segments. In this chapter, we present some of the basic definitions and algorithms of graph theory. In the first section, we give a formal definition of a graph, and we consider the classical bridges of Königsberg problem. In the second section, we give a definition of a directed graph, or digraph. These appear in Chapters I and II to represent finite-state machines and relations. We also present Dijkstra's algorithm for finding a shortest path. In the third section, we consider trees. Examples of trees include a family tree or a tree that shows the various directories on a large computer system. An algorithm for finding a minimum spanning tree is presented. In the final section, we consider networks, prove the Max-Flow Min-Cut Theorem, and give an algorithm for finding a maximum flow.

Good references for further reading are Bondy and Murty [5], Chartrand and Lesniak [7], Bollobás [4], and Roberts [29].

1. Graphs

The origin of graph theory is frequently traced to Euler's solution in 1736 of the following problem.

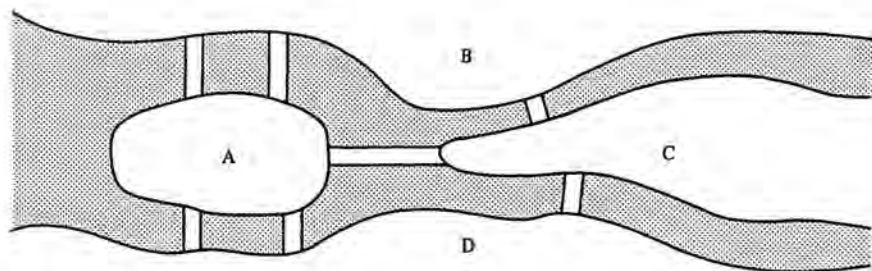


Figure 1.1

(1.1) THE BRIDGES OF KÖNIGSBERG PROBLEM. The city of Königsberg in Prussia (this city is now Kaliningrad in the Soviet Union) was divided into four land masses by the Pregel River, which ran through the city. There were seven bridges connecting these land areas as shown in Figure 1.1. The townspeople wondered whether one could walk through the town and cross each bridge exactly once. Euler proved, as we will see later in this section, that this was impossible.

Euler's basic idea was to label the land areas A, B, C, and D and to consider sequences of these letters corresponding to a walk through the town. We will consider the diagram in Figure 1.2, where we have replaced each land mass by a point, or vertex, and each bridge by an edge joining the corresponding vertices. This is an example of a graph.

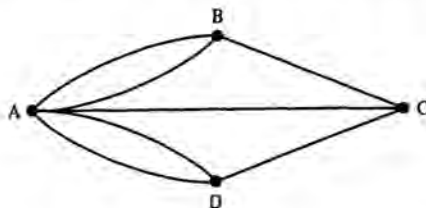


Figure 1.2

Before solving this problem, we give some of the basic definitions of graph theory.

(1.2) DEFINITIONS. A (finite) *graph* G is an ordered triple

$$(V(G), E(G), \psi_G)$$

consisting of

- 1) a finite nonempty set $V(G)$ whose elements are called the *vertices* of G ;
- 2) a (possibly empty) set $E(G)$ whose members are called the *edges* of G ;
- 3) a mapping ψ_G , called the *incidence mapping*, from $E(G)$ to the set consisting of all one-element and two-element subsets of $V(G)$.

If $e \in E(G)$ and $\psi_G(e) = \{u, v\}$, where u and v are distinct vertices, then e is called a *link* and we say e *joins* u and v . In this case, we also call u and v the *ends* of e , and we say that u (respectively, v) and e are *incident*. Two vertices are called *adjacent* if they are joined by an edge.

If $e \in E(G)$ and $\psi_G(e) = \{v\}$, then e is called a *loop*. A graph is called *simple* if it has no loops and if no two of its links have the same ends. If G is a simple graph, $e \in E(G)$, and $\psi_G(e) = \{u, v\}$, then we will also refer to e as "the edge uv ."

We must note here, unfortunately, that there is not complete agreement among graph theorists about the definition of a graph. In many books on graph theory, a "graph" will mean what we have called a simple graph. Our notion of graph above is referred to in some books as a "pseudograph," and a graph without loops but with multiple edges between the same vertices is often called a "multigraph."

(1.3) DEFINITION. A graph $H = (V(H), E(H), \psi_H)$ is a *subgraph* of G if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, and ψ_H is the restriction of ψ_G to $E(H)$. If V' is a nonempty subset of $V(G)$, then the subgraph of G whose vertex set is V' and whose edge set is the set of those edges of G that have both ends in V' is called the *subgraph of G induced by V'* . Similarly, if E' is a nonempty subset of $E(G)$, then the subgraph of G whose vertex set is the set of ends of edges in E' and whose edge set is E' is called the *subgraph of G induced by E'* .

In particular, if G has at least two vertices and $v \in V(G)$, then we will denote by $G - v$ the subgraph of G induced by $\{u \in V(G) : u \neq v\}$; that is, $G - v$ is obtained from G by deleting v and all edges incident with v . Similarly, if $e \in E(G)$, then $G - e$ will denote the subgraph of G obtained

by deleting e and any vertex that is incident with e and is not incident with any other edge of G .

As with most mathematical objects, we have a definition for when two graphs are “isomorphic,” or essentially the same.

(1.4) DEFINITION. Two graphs G and H are said to be *isomorphic*, denoted $G \cong H$, if there exist one-to-one onto mappings $\theta : V(G) \rightarrow V(H)$ and $\phi : E(G) \rightarrow E(H)$ such that: (1) for all loops $e \in E(G)$, $\psi_G(e) = \{u\}$ if and only if $\psi_H(\phi(e)) = \{\theta(u)\}$ and (2) for all links $e' \in E(G)$, $\psi_G(e') = \{u, v\}$ if and only if $\psi_H(\phi(e')) = \{\theta(u), \theta(v)\}$.

► **(1.5) EXAMPLE.** The two graphs in Figure 1.3 are isomorphic via the following correspondences: $\theta(u_1) = v_4$, $\theta(u_2) = v_1$, $\theta(u_3) = v_3$, $\theta(u_4) = v_2$, and $\phi(e_1) = f_2$, $\phi(e_2) = f_1$, $\phi(e_3) = f_3$. ◀

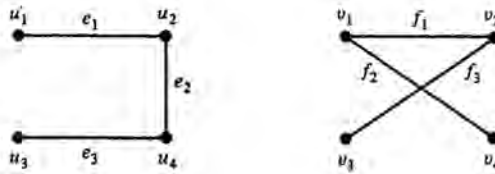


Figure 1.3

Of course, one cannot represent a graph in a computer’s memory by drawing a picture. One common way to represent a graph in a computer is by an array called the adjacency matrix.

(1.6) DEFINITION. Let G be a graph with n vertices v_1, v_2, \dots, v_n . The *adjacency matrix* of G is the $n \times n$ (symmetric) matrix $A(G) = [a_{ij}]$ where a_{ij} is the number of edges joining v_i and v_j .

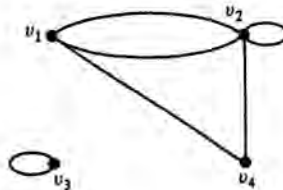


Figure 1.4

► **(1.7) EXAMPLE.** The adjacency matrix of the graph in Figure 1.4 is

$$A = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}, \blacktriangleleft$$

(1.8) DEFINITION. Let $v \in V(G)$. The *degree* of v , denoted $d(v)$, is defined to be the number of edges of G that are incident with v with each loop at v counting as two edges.

For example, in the graph of Figure 1.4, we have $d(v_1) = 3$ and $d(v_2) = 5$. In terms of the entries in the adjacency matrix $A(G) = [a_{ij}]$, we have

$$d(v_i) = \sum_{j \neq i} a_{ij} + 2a_{ii}.$$

Our first proposition is the following easy result.

(1.9) PROPOSITION. Suppose G has p vertices and q edges. Then

$$\sum_{i=1}^p d(v_i) = 2q.$$

PROOF. Adding the degrees of all the vertices amounts to taking the sum of all the entries in the adjacency matrix with the entries on the diagonal counted twice. But this sum is precisely twice the number of edges since each link is counted twice (if e links v_i and v_j , then e is counted in a_{ij} and in a_{ji}) and each loop is counted twice (since a loop at v_i is counted only in a_{ii} and the diagonal entry a_{ii} is counted twice in the above sum). ■

(1.10) COROLLARY. In any graph, there is an even number of vertices of odd degree.

PROOF. Split the vertices up into two sets with V_1 consisting of all the vertices of odd degree and V_2 consisting of all the vertices of even degree. Then

$$\sum_{v \in V_1} d(v) + \sum_{v \in V_2} d(v) = \sum_{v \in V} d(v) \quad (*)$$

is even by Proposition (1.9). Since the sum of even integers is even, the second sum on the left side of $(*)$ is even. Then, since the difference of even integers is even, we may conclude that $\sum_{v \in V_1} d(v)$ is even. Since each term in this sum is odd, there must be an even number of terms; i.e., $|V_1|$ must be even. ■

We now give several definitions related to traveling around a graph.

(1.11) DEFINITIONS

- 1) Let u and v be (not necessarily distinct) vertices of a graph G . A u - v walk of length n is a sequence

$$u = u_0, e_1, u_1, e_2, \dots, u_{n-1}, e_n, u_n = v$$

whose terms are alternately vertices and edges such that the ends of e_i are u_{i-1} and u_i for $i = 1, 2, \dots, n$. A u - v walk is called a *closed walk* if $u = v$. The u - u walk of length 0 consisting just of the vertex u is called a *trivial walk*.

- 2) A u - v *trail* is a u - v walk in which the edges are distinct. A closed trail of positive length is also called a *circuit* of G .
- 3) A u - v *path* is a u - v trail in which the vertices are distinct.
- 4) Two vertices u and v are said to be *connected* if there exists a u - v path. The graph G is called *connected* if every two vertices of G are connected.
- 5) A *cycle* of G is a circuit $u_0, e_1, u_1, \dots, u_{n-1}, e_n, u_n = u_0$ such that $u_i \neq u_0$ if $i \neq 0, n$.

► (1.12) **EXAMPLE.** Referring to the graph G in Figure 1.5, we have

$v_2, e_3, v_4, e_4, v_2, e_3, v_4$ is a walk, but not a trail.

v_1, e_1, v_1, e_2, v_2 is a trail, but not a path

$v_1, e_2, v_2, e_3, v_4, e_5, v_3$ is a path

$v_2, e_3, v_4, e_5, v_3, e_6, v_2$ is a cycle.

Note that G is also a connected graph. ◀

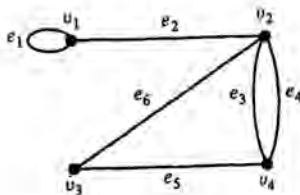


Figure 1.5

We note that in a simple graph, one may specify a walk just by giving the sequence of vertices in the walk.

We now return to the bridges of Königsberg problem. What is called for in that problem is a trail in which every edge appears.

(1.13) DEFINITION. An *Eulerian trail* of a graph G is a trail containing all the edges of G .

(1.14) THEOREM. Let G be a connected graph. Then G has a closed Eulerian trail if and only if every vertex of G has even degree.

PROOF. First, suppose that G has a closed Eulerian trail $v = u_0, e_1, u_1, \dots, u_{n-1}, e_n, u_n = v$. Notice that each time a vertex $u_i \neq v$ appears in this trail, it is incident with either two distinct edges or a loop. Since every edge of G occurs in this trail, we may conclude that every vertex, except possibly v , has even degree. But since the trail begins and ends with v , we see that v also must have even degree. (Note that the degree of v may be greater than two, since v may also occur as an "internal" vertex of the trail.)

We prove the converse by way of contradiction. Suppose that there exist nonempty connected graphs such that every vertex has even degree, but such that there is no closed Eulerian trail. Among all such graphs, choose one with the least number of edges and call it G . (This is done by invoking the Well-Ordering Principle (I.2.4).) We will wind up showing that G does in fact have a closed Eulerian trail.

It is obvious that a connected graph with exactly one vertex has a closed Eulerian trail. Also, any connected graph with precisely two vertices of even degree must have a closed Eulerian trail, since if the two vertices were not joined by a (positive) even number of distinct edges, then they would not have even degree. Therefore, we may assume that G has at least three distinct vertices, u, v , and w such that v is adjacent to u via an edge e_1 and v is adjacent to w via an edge e_2 . Let G' be the graph obtained from G by deleting the edges e_1 and e_2 and by adding an edge e' joining u and w . (Note: we do not delete v in this process.) Then in G' , the degrees of u and of w are the same as they were in G , and the degree of v is two less than it was in G . Thus, the degree of every vertex of G' is even.

There are now two cases to be considered. If G' is still connected, then G' must contain a closed Eulerian trail since G' has one less edge than G . By replacing the edge e' in this trail by the deleted edges e_1 and e_2 , we obtain a closed Eulerian trail of G .

Finally, suppose that G' is not connected. Let G'_1 denote the subgraph of G' induced by the set of vertices that are connected to u (and hence to w also) and let G'_2 denote the subgraph of G' induced by the set of all vertices that are connected to v . Then it is not hard to see that G'_1 and G'_2 are both connected (since G was) and clearly every vertex of each of these two graphs has even degree (since this was true in G'). Thus by our choice of G , there must exist closed Eulerian trails of G'_1 and G'_2 . A closed Eulerian trail of G may then be produced by replacing the edge e' in the closed Eulerian trail of G'_1 by the following: the edge e_1 , followed by the closed Eulerian trail of G'_2 , followed by the edge e_2 . ■

Since the graph in Figure 1.2 has three vertices of degree 3 and one vertex of degree 5, Theorem (1.14) shows that there cannot be a journey through the city of Königsberg that crosses each bridge exactly once and that begins and ends on the same land mass. Suppose we allow someone to begin and end the journey on different land masses. Then we are asking whether there exists an Eulerian trail of the graph in Figure 1.2 that is not closed. Such a nonclosed trail does not exist by virtue of the following result.

(1.15) COROLLARY. Let G be a connected graph. Then G contains a nonclosed Eulerian trail if and only if G has exactly two vertices of odd degree.

PROOF. If G contains an Eulerian u - v trail, then, as in the first part of the proof of Theorem (1.14), every vertex different from u and v must have even degree. Also u must have odd degree since the first edge in the trail contributes 1 to the degree of u , and any other (internal) occurrence of u in the trail contributes 2 to the degree of u . Similarly, v must also have odd degree.

Conversely, suppose that G is a connected graph with exactly two odd-degree vertices u and v . Add a new edge e to G which joins u and v and call this new graph H . Then every vertex of H has even degree, so by Theorem (1.14), H contains a closed Eulerian trail. Since this trail is closed and contains e , we may write it as $v, e, u, e_1, \dots, u_{n-1}, e_n, v$. Then $u, e_1, \dots, u_{n-1}, e_n, v$ is an Eulerian u - v trail in the original graph G . ■

EXERCISES

- † 1.1. A simple graph in which every pair of distinct vertices is joined by an edge is called a *complete graph*. Up to isomorphism, there is just one complete graph on n vertices and it is denoted by K_n .
- What is the degree of each vertex of K_n ?
 - Show that the number of edges of K_n is $\binom{n}{2}$.
 - If G is a simple graph with p vertices and q edges, show that $q \leq \binom{p}{2}$.
 - Draw K_3, K_4 , and K_5 . If possible, draw each graph in the plane so that its edges intersect only at their ends.
- 1.2. Let G be the graph in Figure 1.5.
- Find the adjacency matrix of G .
 - Find the degree of each vertex of G and verify Proposition (1.1) for this graph.
 - Does G contain an Eulerian trail? If so, find one.

- † 1.3. Show that the number of people at a party who shake hands an odd number of times is even.
- 1.4. Can you build one more bridge in the city of Königsberg so that there now exists a (nonclosed) Eulerian trail?
- † 1.5. Suppose G is a graph such that $d(v) \geq 2$ for all $v \in V(G)$. Show that G contains a cycle.
- † 1.6. Suppose that there are two distinct paths between two vertices u and v in a graph G . Show that G must have a cycle.
- † 1.7. A walk W is said to *contain* a walk W' if W' is a subsequence of W . Show that every u - v walk contains a u - v path.
- 1.8. There are 11 nonisomorphic simple graphs with exactly 4 vertices. Draw these graphs.
- † 1.9. Suppose that G is a simple graph with p vertices and q edges. If $q > \binom{p-1}{2}$, then show that G must be connected.
- 1.10. Show that the relation "is connected to" is an equivalence relation (see II.3) on the set of vertices of a graph. The equivalence classes of this equivalence relation are called the *connected components* of G .
- 1.11. A vertex v of G is called a *cut vertex* if the subgraph $G - v$ has more connected components than G does. In particular, if G is connected, then v is a cut vertex if and only if $G - v$ is not connected. Cut vertices would be important when one is constructing a communications network, for example. Find the cut vertices in the graphs in Figure 1.3 and in the graph in Figure 1.5.
- † 1.12. Prove that if G is a connected graph with $p > 1$ vertices, then G has at least $p - 1$ edges.
- 1.13. Let G be a simple graph with at least two vertices. Prove that (at least) two vertices of G must have the same degree. (Hint: Apply the Pigeonhole Principle.)
- 1.14. A graph G is called a *bipartite* graph if $V(G)$ can be partitioned into two nonempty subsets V_1 and V_2 such that each edge of G has one end in V_1 and one end in V_2 .
 - a) Draw a bipartite simple graph with five vertices. 322
 - b) Show that a bipartite simple graph with five vertices can have at most six edges.

- 1.15. A cycle of a graph G which contains every vertex of G is called a *Hamiltonian cycle*. Finding a Hamiltonian cycle would be useful to a traveling salesman who wishes to visit every city on a business trip exactly once. Unfortunately, there is no truly efficient algorithm for finding Hamiltonian cycles.
- Show that every complete graph (cf. Exercise 1.1 above) has a Hamiltonian cycle.
 - Does the graph in Figure 1.5 have a Hamiltonian cycle?
- 1.16. A very famous problem in graph theory was the four-color problem. This was solved in the mid-1970's by K. Appel and W. Haken. Let G be a graph without loops. A (proper) k -coloring of G is an assignment of k colors to the vertices of G such that no two distinct adjacent vertices have the same color. For example, in Figure 1.6 we give a 3-coloring of a graph.

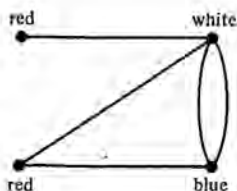


Figure 1.6

A graph is called k -colorable if it has a k -coloring. A graph is called *planar* if it can be drawn in the plane so that its edges intersect only at their ends. The Four-Color Theorem states that every planar graph is 4-colorable. This was thought to be true by mapmakers — a k -coloring of a planar graph corresponds to coloring countries on a map with k colors so that no two adjacent countries are colored the same.

- Find a 2-coloring of each of the graphs in Figure 1.3.
- Show that a graph is bipartite (cf. Exercise 1.14 above) if and only if it is 2-colorable.
- Is the complete graph K_5 (cf. Exercise 1.1 above) 4-colorable? Is this graph planar? (cf. Exercise 1.1(d) above)

2. Digraphs

If one wishes to model a system of two-way streets, then a graph will do fine, with the vertices representing intersections and the edges representing the streets. But suppose one wishes to model a system of streets that include one-way streets. Then a graph as in §1 will not serve because one needs to give an edge a direction or orientation when that edge represents a one-way street. What is needed in this case is a directed graph. Directed graphs appear in Chapters I and II, where they are used to represent finite-state machines and relations on a finite set. A digraph with three vertices is shown in Figure 2.1.

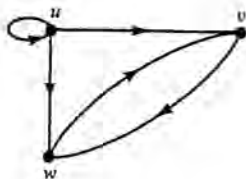


Figure 2.1

(2.1) DEFINITIONS. A (finite) *directed graph*, or *digraph*, D is an ordered triple

$$(V(D), A(D), \psi_D)$$

consisting of

- 1) a finite nonempty set $V(D)$ whose elements are called the *vertices*, or *nodes*, of D ;
- 2) a (possibly empty) set $A(D)$ whose members are called the *arcs*, or *directed edges*, of D ;
- 3) a mapping $\psi_D : V(D) \rightarrow V(D) \times V(D)$, called the *incidence mapping*; so, if $a \in A(D)$, then $\psi_D(a)$ is an ordered pair of (not necessarily distinct) vertices.

If $a \in A(D)$ and $\psi_D(a) = (u, v)$, then we picture a as an arrow with tail at u and head at v . In fact, we call u the *tail* of a and v the *head* of a , and we say that a *joins* u to v . In this case, we also say a is *incident from* u and *incident to* v , and we say u is *adjacent to* v and v is *adjacent from* u .

If $a \in A(D)$ and $\psi_D(a) = (u, u)$, then we call a a *loop* at u .³² A digraph is called *simple* if it is without loops and if no two of its arcs have the same tail and the same head. Again, many books use “digraph” to mean what

we have called a simple digraph. If D is a simple digraph, $a \in A(D)$ and $\psi_D(a) = (u, v)$, then we will also refer to a as “the arc (u, v) .”

The notions of subdigraph and isomorphic digraphs are defined in a similar manner to the analogous notions for (undirected) graphs. If $V(D) = \{v_1, v_2, \dots, v_n\}$, then the *adjacency matrix* of D is the $n \times n$ matrix $[a_{ij}]$ where a_{ij} is the number of arcs with tail v_i and head v_j .

The notion of a walk for a graph now becomes the notion of a *directed walk* with “arc” replacing “edge” in Definition (1.11). Similarly, one can define *directed path*, *directed trail*, and *directed cycle* for digraphs by replacing “edge” by “arc” in the definitions we gave for (undirected) graphs.

(2.2) DEFINITION. Suppose $u, v \in V(D)$. Then v is said to be *reachable* from u if there exists a u - v directed path in D .

For example, in the digraph in Figure 2.1, v is reachable from u , but u is unreachable from v .

For digraphs, we have the notions of indegree and outdegree of a vertex. These correspond to the notion of degree in an undirected graph.

(2.3) DEFINITIONS. Suppose $v \in V(D)$. The *indegree* of v , denoted $d^-(v)$, is the number of arcs with head v . The *outdegree* of v , denoted $d^+(v)$, is the number of arcs with tail v . The *degree* of v , denoted $d(v)$, is defined by $d(v) = d^-(v) + d^+(v)$.

In the digraph in Figure 2.1, we have $d^-(u) = 1$, $d^+(u) = 3$, and $d(u) = 4$.

There are some relationships between graphs and digraphs.

(2.4) DEFINITIONS. Suppose D is a digraph. The *underlying graph* of D is the graph obtained by “forgetting” the orientation on the arcs; i.e., the underlying graph of D is the graph G that has the same vertex set as D and has an edge corresponding to each arc of D . A digraph D is called *connected* if its underlying graph is connected.

The underlying graph of the digraph in Figure 2.1 is shown in Figure 2.2. Note that the digraph of Figure 2.1 is thus connected, even though u is unreachable from v .

Conversely, given any graph G , we can obtain a digraph by specifying an orientation for each edge of G . For example, in Figure 2.3, we have chosen an orientation for the complete graph on four vertices K_4 . Such a digraph has a special name.

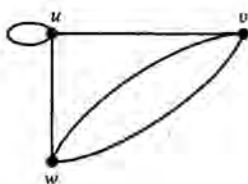


Figure 2.2

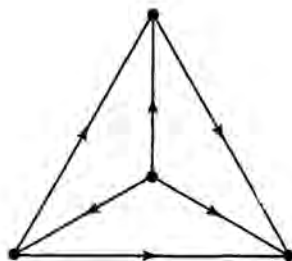


Figure 2.3

(2.5) DEFINITION. An orientation of a complete graph is called a *tournament*.

The reason for this name is that we can view an orientation on K_n as giving the results of a round-robin tournament with n competitors, with an arc from v_i to v_j meaning that the i^{th} competitor beat the j^{th} competitor.

For the remainder of this section, we will consider an algorithm due to the Dutch mathematician E. Dijkstra for solving an important optimization problem known as the “shortest path problem”. Although we will present this algorithm for digraphs, it is easy to see how to modify it in order to apply it to graphs. We need another definition.

(2.6) DEFINITION. A digraph D is called a *weighted digraph* if each arc a of D is assigned a positive real number $w(a)$, called the *weight* of a .

Suppose we wish to make a trip by airplane between two cities that are not connected by a direct flight; call the cities Metropolis and Smallville. Also suppose we want to do this in the most economical way possible. We could form a digraph with each vertex representing a city and with an arc going from city u to city v if and only if there is a flight from u to v . Then every possible sequence of connecting flights from Metropolis to Smallville

is represented by a directed path of this digraph. (To keep this fairly simple, we will not draw multiple arcs between cities to represent flights on different airlines.) We could then label each arc with the cost of the corresponding flight. This gives us a weighted digraph. We then must find the directed path from Metropolis to Smallville such that the sum of the weights of the arcs that make up the path is a minimum. By virtue of the following definition, such a path would be called a "shortest" path, although here the name "cheapest" path would be more appropriate.

(2.7) DEFINITIONS. Suppose D is a weighted digraph. If P is a directed path of D , then the *weighted length* of P is the sum of the weights of the arcs of P . If $u, v \in V(D)$ and v is reachable from u , then the *distance from u to v* , denoted $d(u, v)$, is the minimum of the weighted lengths of the u - v directed paths of D . If v is not reachable from u , then we set $d(u, v) = \infty$.

Now suppose D is a weighted simple digraph and $u_0 \in V(D)$. The Shortest Path Algorithm that we will describe will find the distance from u_0 to every other vertex of D and the shortest directed path from u_0 to every vertex that is reachable from u_0 . Our presentation follows that in Chartrand and Lesniak [7]. The key step in the algorithm is based on the following remarks.

Suppose $S \subseteq V(D)$ and put $\bar{S} = \{v \in V(D) : v \notin S\}$. Suppose $u_0 \in S$. Define $d(u_0, \bar{S})$ by

$$d(u_0, \bar{S}) = \min_{v' \in \bar{S}} \{d(u_0, v')\}.$$

Of course, since distances are natural numbers or infinity, this minimum exists (but may be ∞), and we may choose a vertex $v \in \bar{S}$ such that $d(u_0, v) = d(u_0, \bar{S})$. Suppose $d(u_0, v) < \infty$ and let $P : u_0, u_1, \dots, u_n, v$ denote a directed path in D of length $d(u_0, v)$. (Since D is simple, to specify a directed path we need not specify the edges.) Then we must have $u_i \in S$ for $i = 1, 2, \dots, n$ or else there would be a vertex in \bar{S} that would be closer to u_0 than v . Also, since P is a u_0 - v path of minimum weighted length, we must have that u_0, u_1, \dots, u_n is a u_0 - u_n path of minimum weighted length. We conclude that

$$d(u_0, \bar{S}) = \min_{u \in S, v \in \bar{S}} \{d(u_0, u) + w(u, v)\}, \quad (*)$$

where $w(u, v)$ is the weight of the arc from u to v .

Now for the algorithm. Fix a vertex u_0 of our simple digraph D . Suppose that D has $p > 1$ vertices. At each step of the algorithm, each vertex $v \neq u_0$ will be labeled either ∞ or with an ordered pair $(L(v), u)$. At the end of the algorithm the first member of the pair (i.e., $L(v)$) gives the distance from u_0 to v , and the second member of the pair gives the predecessor of v on a shortest u_0 - v path if v is reachable from u_0 .

(2.8) SHORTEST PATH ALGORITHM

Step 0. Set $i = 0, S_0 = \{u_0\}, L(u_0) = 0$, and $L(v) = \infty$ for all vertices $v \neq u_0$.

Step 1. Put $\bar{S}_i = \{v \in V(D) : v \notin S_i\}$. For each $v \in \bar{S}_i$, replace $L(v)$ by

$$\min_{u \in S_i} \{L(v), L(u) + w(u, v)\}$$

(compare with equation (*)). If this produces a new value of $L(v)$, then label v by the ordered pair $(L(v), u_i)$, where u_i is a vertex in S_i where the above minimum is attained.

Step 2. If $L(v) = \infty$ for all $v \in \bar{S}_i$, then stop; otherwise, determine

$$\min_{v \in \bar{S}_i} \{L(v)\}$$

and let u_{i+1} be a vertex where this minimum is attained.

Step 3. Put $S_{i+1} = S_i \cup \{u_{i+1}\}$.

Step 4. Replace i by $i + 1$. If i is now $|V(D)| - 1$, then stop; otherwise, go to Step 1.

At the termination of the algorithm, we will have $L(v) = d(u_0, v)$ for all $v \in V(D)$ and if $L(v) < \infty$, then

$$u_0 = w_0, w_1, w_2, \dots, w_k = v$$

is a shortest (i.e., minimum weighted length) u_0 - v directed path, where w_i is labeled $(L(w_i), w_{i-1})$ for $i = 1, 2, \dots, k$. If $L(v) = \infty$ at the termination of the algorithm, then v is unreachable from u_0 .

We will omit the proof that the algorithm accomplishes what we have stated above. The proof given in [7, pp. 32-33] for graphs may be easily adapted to the digraph case. The idea of the proof is to use induction on i together with the key remarks before the statement of (2.8). We will, of course, illustrate the algorithm with an example.

► (2.9) EXAMPLE. Suppose D is the weighted simple digraph shown in Figure 2.4.

We will find the shortest directed path from u_0 to each of the other vertices. The algorithm proceeds as follows.

Step 0. Set $i = 0, S_0 = \{u_0\}, L(u_0) = 0, L(v_i) = \infty$ for $i = 1, 2, 3, 4, 5$.

Step 1. Replace $L(v_1)$ by 2 and label v_1 with the ordered pair $(2, u_0)$. Replace $L(v_3)$ by 3 and label v_3 with $(3, u_0)$. (We still have $L(v_2) = L(v_4) = L(v_5) = \infty$.)

- Step 2. We have $\min_{v \in \bar{S}_0} \{L(v)\} = 2$ and we let $u_1 = v_1$.
- Step 3. Put $S_1 = \{u_0, v_1\}$.
- Step 4. Set $i = 1$ and go to Step 1.
- Step 1. Replace $L(v_4)$ with 7 and label v_4 with $(7, v_1)$. (We still have $L(v_3) = 3, L(v_2) = L(v_5) = \infty$.)
- Step 2. We have $\min_{v \in \bar{S}_1} \{L(v)\} = 3$ and we let $u_2 = v_3$.
- Step 3. Put $S_2 = \{u_0, v_1, v_3\}$.
- Step 4. Set $i = 2$ and go to Step 1.
- Step 1. Replace $L(v_4)$ by 6 and label v_4 with $(6, v_3)$. (Note that this is a change in the label that v_4 received during the previous run through Step 1.) Replace $L(v_5)$ by 4 and label v_5 with $(4, v_3)$.
- Step 2. We have $\min_{v \in \bar{S}_2} \{L(v)\} = 4$ and we let $u_3 = v_5$.
- Step 3. Put $S_3 = \{u_0, v_1, v_3, v_5\}$.
- Step 4. Set $i = 3$ and go to Step 1.
- Step 1. Replace $L(v_4)$ by 5 and label v_5 with $(5, v_5)$. (This is a second change in the label on v_4 .)
- Step 2. We have $\min_{v \in \bar{S}_3} \{L(v)\} = 5$ and we let $u_4 = v_4$.
- Step 3. Put $S_4 = \{u_0, v_1, v_3, v_5, v_4\}$.
- Step 4. Set $i = 4$ and go to Step 1.
- Step 1. The only vertex in \bar{S}_4 is v_2 and we still have $L(v_2) = \infty$, so the algorithm stops.

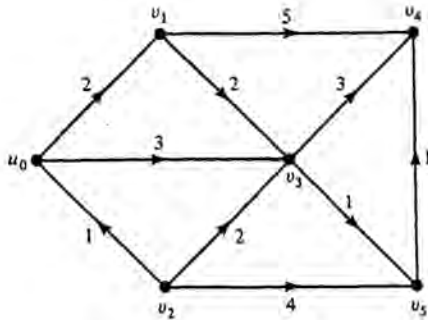


Figure 2.4

Figure 2.5 shows the labels that we now have on the vertices of D . From Figure 2.5, we can read off $d(u_0, v_i)$, and we can trace a shortest path from u_0 to $v_i, i \neq 2$, by working backwards from v_i and noting the second member of each label. For example, $d(u_0, v_4) = 5$, and a shortest path from u_0 to v_4 is u_0, v_3, v_5, v_4 (because the second member of the label on v_4 is v_5 , the

second member of the label on v_5 is v_3 , and the second member of the label on v_3 is u_0). Note that v_2 is unreachable from u_0 . We also remark that in this example it turned out that there was a unique shortest path from u_0 to each $v_i, i \neq 2$. In general, it may turn out that there are several paths from one vertex to another that have the same minimum weighted length. In that case, the algorithm will find one such path. ◀

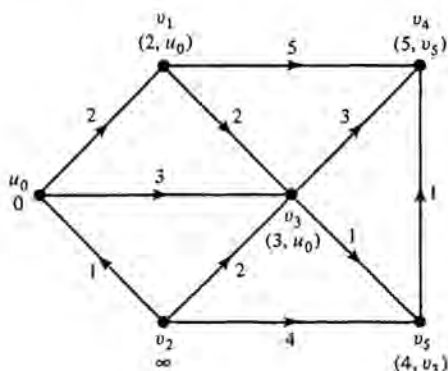


Figure 2.5

EXERCISES

† 2.1. If D is a digraph with q arcs, then show that

$$\sum_{v \in V(D)} d^-(v) = \sum_{v \in V(D)} d^+(v) = q.$$

† 2.2. Can a digraph contain an odd number of vertices of odd outdegree? Can a digraph contain an odd number of vertices of odd degree?

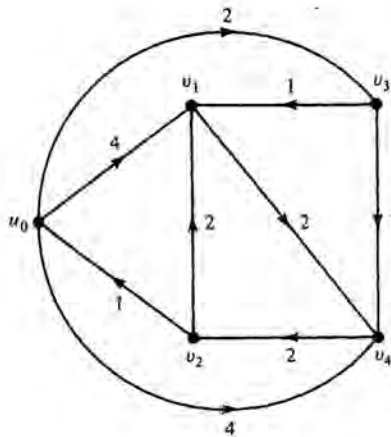
2.3. Determine the indegree and outdegree of each vertex of the tournament in Figure 2.3. Which competitor is the winner of this tournament?

2.4. Draw the two nonisomorphic tournaments with three vertices. Note that not every tournament has a clear winner.

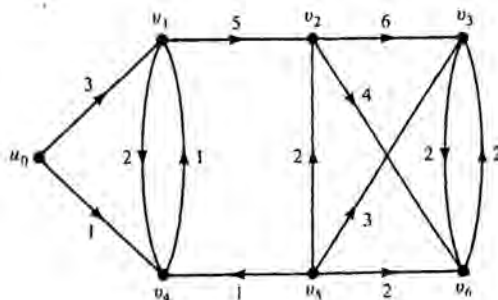
2.5. A simple digraph D is called *symmetric* if whenever (u, v) is an arc of D , then (v, u) is also an arc of D . If G is a simple graph, then the *associated digraph*, denoted $D(G)$, is the symmetric digraph obtained from G by

replacing each edge of G by two arcs pointing in opposite directions. Set up a one-to-one correspondence between symmetric simple digraphs with n vertices and simple (undirected) graphs with n vertices.

- † 2.6. A digraph D is called *strongly connected* (or *disconnected*) if each vertex of D is reachable from every other vertex of D .
- Prove or disprove: If D is strongly connected, then D is connected.
 - Prove or disprove: If D is connected, then D is strongly connected.
- † 2.7. Show that if every vertex of D has positive outdegree, then D contains a directed cycle.
- † 2.8. Apply the Shortest Path Algorithm to each of the following simple digraphs to determine $d(u_0, v_i)$ and the shortest path from u_0 to v_i if v_i is reachable from u_0 .
-



b)



- 2.9.** Prove that a simple connected digraph D has a closed Eulerian directed trail if and only if $d^-(v) = d^+(v)$ for all $v \in A(D)$.

3. Trees

In this section, we study a special type of graph that has nice properties and wide applications.

(3.1) DEFINITION. A *tree* is a connected graph with no cycles.

A graph with no cycles is also called *acyclic* or a *forest*, since its connected components (cf. Exercise 1.10) are trees. Notice that a graph with no cycles must be a simple graph (for a loop would be a cycle and if there were two distinct edges with the same endpoints, then they would form a cycle).

- **(3.2) EXAMPLES.** Examples of trees include family trees and trees which show all directories, subdirectories, and files on a computer system. Some other examples are:

1) Up to isomorphism, there is only one tree with one vertex, one tree with two vertices, and one tree with three vertices. As shown in Figure 3.1, there are two nonisomorphic trees with four vertices and three nonisomorphic trees with five vertices.

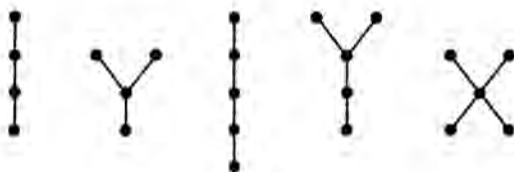


Figure 3.1

2) Trees are often used in computer science for diagramming sort and search procedures. For example, suppose we are given three distinct real numbers a, b , and c and we wish to sort them in increasing order.³³² A tree to accomplish this is shown in Figure 3.2. ◀

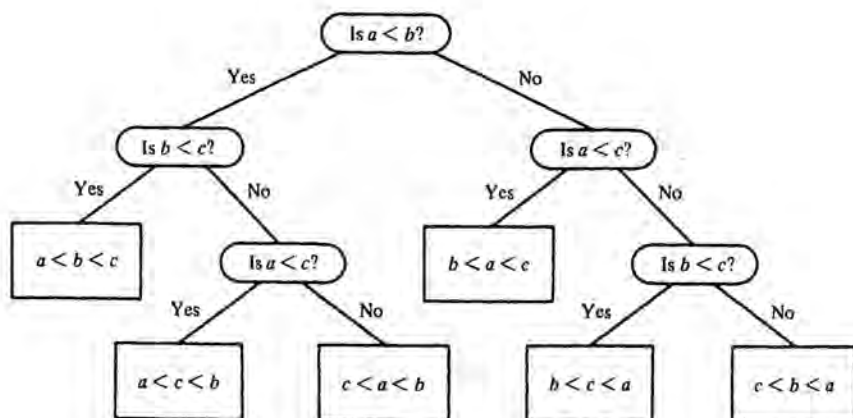


Figure 3.2

(3.3) LEMMA. A tree G with at least two vertices must have a vertex of degree 1.

PROOF. Since G is connected, every vertex of G has positive degree. If every vertex of G had degree at least 2, then G would contain a cycle by Exercise 1.5. Hence G must have at least one vertex of degree 1. ■

A vertex of degree 1 in a tree is called a *pendant vertex* or a *leaf*.

There are several equivalent ways to define trees. We demonstrate some of these equivalent conditions in the following result.

(3.4) THEOREM. Let G be a simple graph with $p \geq 2$ vertices and q edges. Then the following are equivalent:

- 1) G is a tree (i.e., G is connected with no cycles).
- 2) There exists a unique path between any two vertices of G .
- 3) G is connected and $p = q + 1$.
- 4) G has no cycles and $p = q + 1$.

PROOF. We will show that $(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (4) \Rightarrow (1)$. This demonstrates that each of the four statements implies the other three.

$(1) \Rightarrow (2)$: Since G is connected, there is a path between any two vertices of G . If there were two distinct paths between two vertices of G , then G would have a cycle by Exercise 1.6. Since G has no cycles, we conclude that there must be a unique path between any two vertices.

(2) \Rightarrow (3): Obviously G is connected. We will show that $p = q + 1$ by induction on p . If $p = 2$, then, since there is a unique path between the two vertices, it is clear that $q = 1$. Now suppose that (2) \Rightarrow (3) for all graphs with k vertices and let G be a graph with $k + 1$ vertices such that there exists a unique path between any two vertices of G . If all vertices of G have degree at least 2, then, by Exercise 1.5, G would contain a cycle and hence there would be two distinct paths between two vertices of G . Thus G must have a vertex v such that $d(v) = 1$. Let e denote the edge that is incident with v . Since there is a unique path between any two vertices of G , there is a unique path between any two vertices of the subgraph $G - v$. Since $G - v$ has k vertices, our induction hypothesis shows that $G - v$ has $k - 1$ edges. Hence G has k edges (the $k - 1$ edges of $G - v$ together with the edge e).

(3) \Rightarrow (4): We must show that G has no cycles. Again, we use induction on p . If $p = 2$ and $q = p - 1 = 1$, then clearly G has no cycles. Assume that (3) \Rightarrow (4) for all graphs with k vertices and let G be a graph with $k + 1$ vertices such that G is connected and G has k edges. If every vertex of G had degree at least 2, then, by Proposition (1.9), G would have at least $k + 1$ edges. Since G only has k edges, we conclude that G must have a vertex v such that $d(v) = 1$. Then the subgraph $G - v$ is connected and has $k - 1$ edges, so by our induction hypothesis, $G - v$ has no cycles. Since $d(v) = 1$, it is then not hard to see that G cannot have any cycles.

(4) \Rightarrow (1): We must show that G is connected. Again, we use induction on p . Clearly, if $p = 2$ and $q = p - 1 = 1$, then G is connected (G cannot have a loop since it has no cycles). Assume that (4) \Rightarrow (1) for all graphs with k vertices and let G be a graph with $k + 1$ vertices such that G has no cycles and G has k edges. As before, not every vertex of G can have degree at least 2. If G had no vertices of degree 1, then by deleting vertices of degree 0 we would arrive at a subgraph of G whose vertices all had degree at least 2. Hence this subgraph, and therefore G , would have a cycle. Thus we can conclude that G must have a vertex v such that $d(v) = 1$. Then $G - v$ has no cycles and has $k - 1$ edges, so by our induction hypothesis, $G - v$ is connected. But then G must be connected. ■

We next consider an application of trees. Suppose one wants to connect a number of cities by railroad tracks so that trains can get from any city to any other city (possibly going through some of the other cities on the way). Suppose that we know the cost of laying track between each pair of our cities. Which tracks should we lay to connect the cities at a minimum cost?

This translates into a graph theory problem as follows: If there are n cities, then we can draw the complete graph on n vertices, K_n , and the fact that we know the cost of laying track between any two of the cities means

that we can attach a positive real number $w(e)$ ($=$ cost) to each edge e of our graph. In analogy with Definition (2.6), we call the graph together with the function w a *weighted graph*. What we want now is a connected subgraph of this graph such that all n vertices are vertices of the subgraph (because we want to be able to get from any city to any other city by train) and we want the sum of the weights of the edges of this subgraph to be a minimum among all such subgraphs. This motivates the following definitions.

(3.5) DEFINITIONS

- 1) A subgraph H of a graph G is called a *spanning subgraph* if $V(H) = V(G)$. A spanning subgraph that is a tree is called a *spanning tree*.
- 2) If G is a weighted graph and H is a subgraph of G , then the *weight* of H , denoted $w(H)$, is the sum of the weights of all the edges in $E(H)$.

Suppose G is a connected graph. Then G has at least one spanning tree. We can see this as follows: If G is itself a tree, then we are done. If G is not a tree, then G has cycles. If we delete an edge from a cycle of G , then the resulting subgraph will still be connected and will be a spanning subgraph since we didn't delete any vertices. We can continue this process until we reach a connected spanning subgraph with no cycles (i.e., a spanning tree).

(3.6) DEFINITION. If G is a connected weighted graph, then a *minimum spanning tree* of G is a spanning tree T such that if T' is any other spanning tree of G , then $w(T) \leq w(T')$.

Now it should be evident that to solve the railroad connection problem above, what we need is a minimum spanning tree of our weighted graph. We will now describe an algorithm due to J. Kruskal for finding a minimum spanning tree in a connected weighted graph. This is an example of a so-called "greedy" algorithm — if we act in our best interests at each step, then in the end we attain our goal.

(3.7) MINIMUM SPANNING TREE ALGORITHM. Suppose G is a connected weighted graph with $p \geq 2$ vertices.

Step 0. Set $i = 0$ and put $E_0 = \emptyset$.

Step 1. Put $\bar{E}_i = \{e \in E(G) : e \notin E_i\}$. Determine $\min_{e \in \bar{E}_i} \{w(e)\}$ and choose, if possible, an edge $e_{i+1} \in \bar{E}_i$ such that $w(e_{i+1})$ is this minimum and such that the subgraph T_{i+1} of G induced by $E_i \cup \{e_{i+1}\}$ has no cycles. If such an edge exists, then put $E_{i+1} = E_i \cup \{e_{i+1}\}$. If no such edge exists, then stop and let $T = T_i$.

Step 2. Replace i by $i + 1$ and return to Step 1.

The subgraph T produced by this algorithm is a minimum spanning tree of G .

PROOF. Clearly, the subgraph T that is produced has no cycles. Also, it is not hard to see that T is a spanning subgraph of G , since if some vertex v of G were not in T , then an edge incident with v could be added in Step 1 of the algorithm. Furthermore, T must be connected; for if it were not, then there would be vertices u and v in T with no u - v path in T . But since there is a u - v path in G , we would be able to add an edge to T in Step 1 of the algorithm without creating a cycle in T . Thus, since T is connected, has no cycles, and is a spanning subgraph of G , we see that T is a spanning tree.

It remains to show that T is a spanning tree of minimum weight. Suppose not and among all minimum spanning trees of G , let T' be one having the maximum number of edges in common with the spanning tree T . Let e_1, e_2, \dots, e_{p-1} denote the edges of T . Since $T \neq T'$ and any two spanning trees of G have exactly $|v(G)| - 1$ edges, there exists an edge of T that is not an edge of T' . Let e_i be the first edge of T that is not an edge of T' . Add e_i to T' and call the resulting subgraph (of G) H . Then H has too many edges to be a tree, so H must have a cycle. Since T has no cycle, there must be an edge e_0 of this cycle in H such that e_0 is not an edge of T .

Now let T'' be the graph obtained from H by deleting e_0 . Then T'' is connected and has $p - 1$ edges (and p vertices), so T'' is another spanning tree. Since $w(T'') = w(T') + w(e_i) - w(e_0)$ and T' is a minimum spanning tree, we conclude that $w(e_0) \leq w(e_i)$. Now, $e_1, e_2, \dots, e_{i-1}, e_0$ are edges of the tree T'' . Hence the subgraph of G with edges $e_1, e_2, \dots, e_{i-1}, e_0$ does not have a cycle. By the choice of e_i in Step 1 of the algorithm, it follows that $w(e_0)$ cannot be less than $w(e_i)$. Thus we must have $w(e_i) = w(e_0)$ and so $w(T'') = w(T')$. Therefore T'' is also a minimum spanning tree. But T'' has one more edge (namely e_i) in common with T than does T' . This contradicts the choice of T' and shows that T must be a minimum spanning tree. ■

- **(3.8) EXAMPLE.** We will apply the Minimum Spanning Tree Algorithm to the connected weighted graph in Figure 3.3.

For e_1 , we take the edge v_1v_2 of weight 2. For e_2 , we can take either of the edges v_3v_4 or v_4v_6 , since each has weight 3. Let's take $e_2 = v_4v_6$. Then for e_3 we take v_3v_4 . For e_4 , we take the edge v_1v_6 . Now notice that we do not take the edge v_1v_4 for e_5 , even though this edge has the minimum weight among all unused edges, since we would then have the cycle $v_1v_6v_4v_1$. So, for e_5 we can take either the edge v_2v_6 or the edge v_5v_6 . Thus in this example there is more than one spanning tree of minimum weight ($= 20$). One of these trees is shown in Figure 3.4. ◀

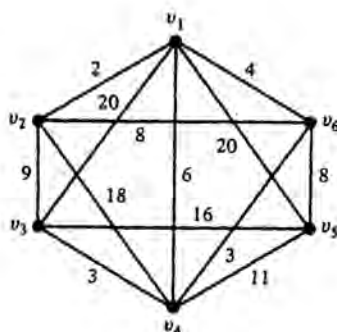


Figure 3.3

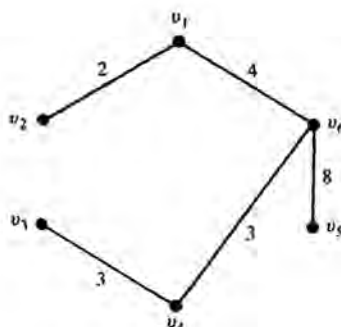


Figure 3.4

We close this section by discussing rooted trees.

(3.9) DEFINITION. A *directed tree* is a digraph whose underlying graph is a tree. A *rooted tree* is a directed tree T with a distinguished vertex r , called the *root*, such that T contains an r - v directed path for every vertex v of T .

It is customary to draw a rooted tree with the root at the top. Then, since all arcs are directed downward, one usually omits the arrows on the arcs and just draws the underlying graph. For $p \geq 3$, there are more nonisomorphic rooted trees with p vertices than there are trees with p vertices. For example, there is only one (up to isomorphism) tree with three vertices, but there are two nonisomorphic rooted trees with three vertices. We show these in Figure 3.5(a) with the orientations indicated on the arcs and in Figure 3.5(b) in the more customary way without arrows.

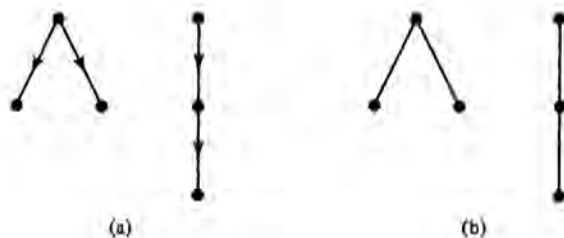


Figure 3.5

Suppose T is a rooted tree with root r . There is some special terminology, derived from family trees, for vertices of rooted trees. For any vertex $v \neq r$, the unique vertex u that is adjacent to v is called the *parent* of v . Any vertices adjacent from v are called *children* of v .

A special type of rooted tree that commonly arises is a *binary tree*. This is a rooted tree such that the root has degree 2 and all other vertices have degree either 1 or 3; i.e., the root has two children and each vertex other than the root has either two children or no children. The tree in Figure 3.2 may be viewed as a binary tree.

Given a connected simple graph G and a vertex u_0 of G , there are two reasonable strategies to follow to construct a rooted tree, with u_0 as the root, whose underlying graph is a spanning tree of G .

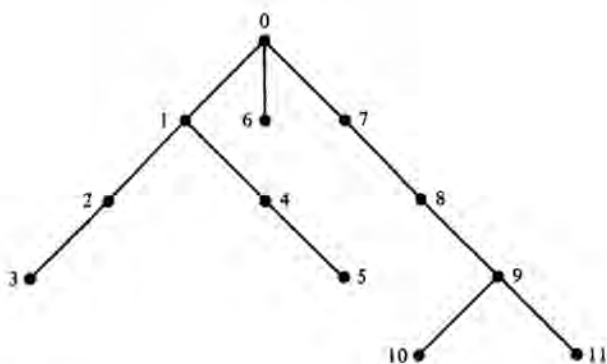
The first method, called the *depth-first search procedure*, can be described as follows: Take u_0 as the root. Form the leftmost path (branch) in the rooted tree by successively adding edges of G such that each new edge joins the (previous) last vertex in this path to an unused vertex of G . When no more edges can be added to this branch, back up to the next-to-last vertex and, if possible, form a new branch by the same procedure beginning with this vertex. If this is not possible, then back up to the previous vertex. By continuing this process, we construct the branches of a rooted tree from left to right.

The second method, called the *breadth-first search procedure*, can be described as follows: Take u_0 as the root. Next add all edges of G that are incident to u_0 as branches from left to right. Next, go to the leftmost vertex adjacent from the root and add (from left to right) all edges of G incident with this vertex. Then go to the next vertex on the right that is adjacent from the root and add all edges of G incident to this vertex. Continue this process for each vertex that is incident from the root. Then proceed to the "grandchildren" of the root and repeat this procedure from left to right. In

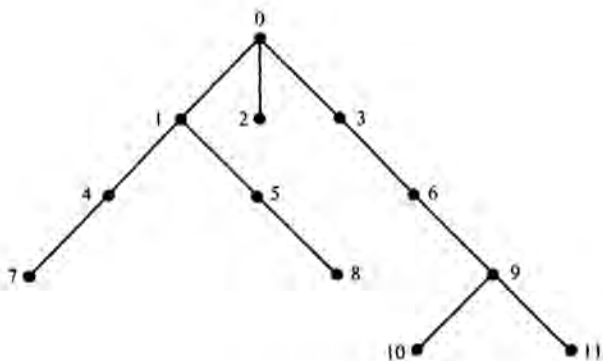
this way, we construct the tree from top to bottom, constructing each new “level” from left to right.

These procedures may also be used to label, or index, the vertices of a rooted tree.

- **(3.10) EXAMPLE.** In Figure 3.6(a), we have used the depth-first search procedure to label the vertices of a rooted tree from 0 to 11. In Figure 3.6(b), we have used the breadth-first search procedure to label the vertices of the same rooted tree. ◀



(a) Depth-first



(b) Breadth-first

Figure 3.6

EXERCISES

- † 3.1. Strengthen Lemma (3.3) by showing that a tree with at least two vertices must have at least two pendant vertices.
- 3.2. There are six nonisomorphic trees with six vertices. Draw them.
- † 3.3. An edge e of a graph G is called a *bridge*, or *cut edge*, if the subgraph $G - e$ has more connected components (cf. Exercise 1.10) than does G .
- Show that an edge e is a bridge if and only if e is on no cycle of G .
 - Show that a connected graph is a tree if and only if every edge is a bridge.
- † 3.4. Suppose that G is a connected graph. If $u, v \in V(G)$, then the *distance from u to v* , denoted $d(u, v)$, is the minimum of the lengths of the u - v paths in G . The *eccentricity* of a vertex v , denoted $e(v)$, is the number $\max_{u \in V(G)} \{d(u, v)\}$. A *center* of G is a vertex of minimum eccentricity.
- For each of the trees in Figure 3.1, find the eccentricity of each vertex and find all centers.
 - Prove that a tree has either one center or two adjacent centers.
- † 3.5. Find a minimum spanning tree of the weighted graph in Figure 3.7.

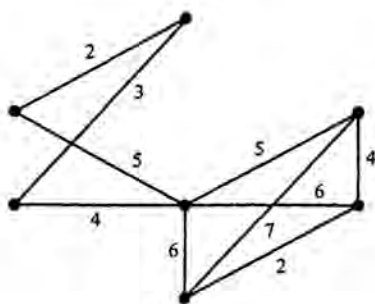


Figure 3.7

- † 3.6. Find three distinct minimum spanning trees of the weighted graph in Figure 3.8.

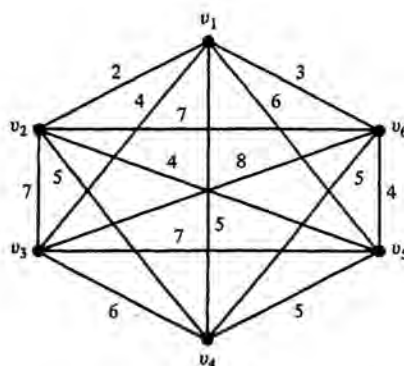


Figure 3.8

- 3.7.** Suppose u_0 is a vertex of the complete graph on n vertices K_n .
- Describe the rooted tree with root u_0 and underlying graph a spanning tree of K_n that would be constructed from K_n by the depth-first search procedure. What is the degree of each vertex of this tree?
 - Describe the rooted tree with root u_0 and underlying graph a spanning tree of K_n that would be constructed from K_n by using the breadth-first search procedure. What is the degree of each vertex of this tree?

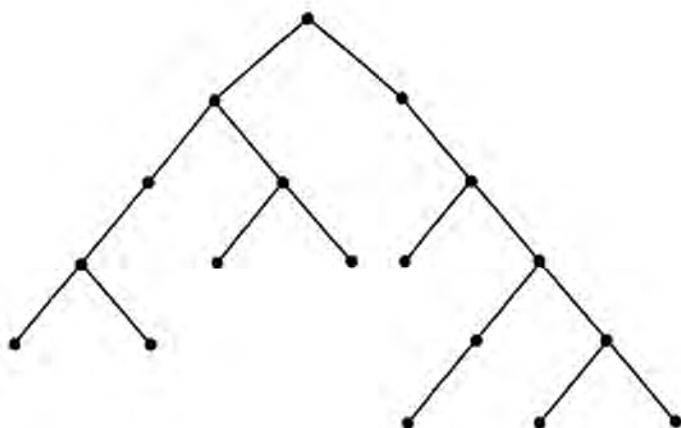


Figure 3.9

- † 3.8. Label the vertices of the binary tree in Figure 3.9 by using the (a) depth-first search procedure and (b) breadth-first search procedure.

4. Networks

Suppose a company wishes to transport a quantity of some product or commodity from its source of production to its final destination. Suppose that there are several possible routes between these two points and that there are intermediate stopping points between the source and the final destination. Further suppose that each link between two stopping points has a certain capacity; that is, can carry a certain amount of the product. For example, an oil company may be transporting oil from a well to some final destination by means of a network of pipelines, and the pipelines that connect various pumping stations may have different capacities for the amount that can flow through them. Or the product may be transported sometimes by truck, sometimes by train and sometimes by air, and the capacity of each link would depend on the size of the trucks, railroad cars, or cargo holds. It is natural to assume that the company would like to choose the route that will allow the greatest amount of the product to "flow" from the source to the final destination. What we have described becomes the problem in graph theory of finding a maximum flow in a network.

(4.1) DEFINITION. A network $N = (D, c, x, y)$ is a weighted digraph D with two distinguished distinct vertices x and y , called the *source* and *sink*, respectively, and a weight function c (with domain $A(D)$, the set of arcs of the digraph D) that is called the *capacity function*.

We will assume throughout this section that c maps $A(D)$ into the set of positive *integers*. The theory is not much more difficult if we assume c takes rational numbers as values, but the situation becomes quite complicated if we allow c to take real numbers (in particular irrational numbers) as values. One should think of the capacity of an arc as measuring the maximum rate at which a product or commodity can be transported along it.

The source vertex x should be viewed as the source of the product or commodity (perhaps a factory or an oil well or a farm) and the sink vertex y should be viewed as the final destination. All the other vertices of D are referred to as *intermediate vertices*.

It may not be possible to utilize each arc to its capacity; that is, the amount of product transported along an arc may have to be less than the capacity of the arc. The amount of product actually transported along each arc gives what is called a *flow*. But before formally defining a flow, we need to introduce a little more notation.

(4.2) DEFINITIONS

- 1) Suppose $S, T \subseteq V = V(D)$. Then we will let (S, T) denote the set of all arcs in $A(D)$ such that the tail of the arc is in S and the head of the arc is in T . In particular, if $u \in V$, then (u, V) will denote the set of all arcs in $A(D)$ with tail at u and (V, u) will denote the set of all arcs in $A(D)$ with head at u .
- 2) Suppose f is a function defined on $A(D)$. Then $f(S, T)$ is defined by

$$f(S, T) = \sum_{a \in (S, T)} f(a).$$

For the rest of this section, we let N be the network (D, c, x, y) . One of the main definitions of this section is the following.

(4.3) DEFINITION. A *flow* f in N is a function $f : A(D) \rightarrow \mathbb{Z}$ such that

- 1) $0 \leq f(a) \leq c(a)$ for all $a \in A(D)$ and
- 2) $f(u, V) = f(V, u)$ for all intermediate vertices u (i.e., for all vertices u such that $u \neq x, y$).

The second condition in the definition of a flow just says that the amount of the product that comes into any intermediate vertex must be the same as the amount of the product that goes out from that vertex. This is a very natural conservation requirement.

- **(4.4) EXAMPLES.** Note that the function that assigns 0 to every arc is a (trivial) flow. In Figures 4.1(a) and 4.1(b), we show two flows on a network N . Each arc a of the digraph is labeled $c(a), f(a)$. Note that the conservation requirement is met at each intermediate vertex in these two flows. ◀

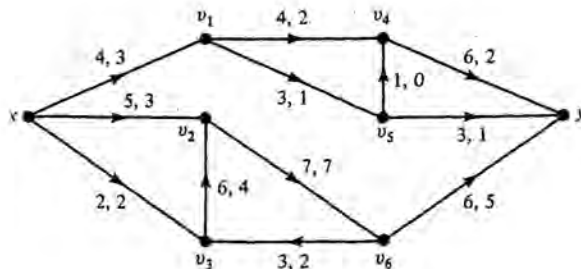


Figure 4.1(a)

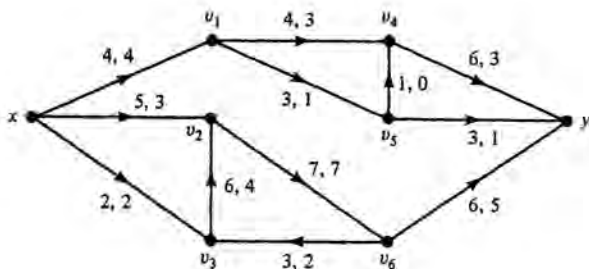


Figure 4.1(b)

Because of the conservation condition satisfied by a flow at each intermediate vertex, it is plausible that the net amount of product that leaves the source should equal the net amount of product that reaches the sink. This is the content of the next proposition. We note that although in most examples all arcs involving x will point away from x and all arcs involving y will point toward y , this is not part of the definition of a network. Hence, in the next proposition we must consider arcs with head x (i.e., the set (V, x)) and arcs with tail y (i.e., the set (y, V)).

(4.5) PROPOSITION. Suppose f is a flow in the network N . Then

$$f(x, V) - f(V, x) = f(V, y) - f(y, V).$$

PROOF. First note that we have

$$\sum_{v \in V} f(v, V) = \sum_{v \in V} f(V, v) \quad (*)$$

since both sides equal $f(V, V)$. But, since f is a flow, we have $f(v, V) = f(V, v)$ for all intermediate vertices v . Subtracting these common terms from both sides of $(*)$, we are left with

$$f(x, V) + f(y, V) = f(V, x) + f(V, y),$$

which is what we wanted to show. ■

(4.6) DEFINITION. The *value of a flow* f , denoted $\text{val } f$, is defined by

$$\text{val } f = f(x, V) - f(V, x).$$

The value of f is thus the net flow out of the source, and Proposition (4.5) shows that this is the same as the net flow into the sink. The value of the flow in Figure 4.1(a) is 8 and the value of the flow in Figure 4.1(b) is 9.

(4.7) DEFINITION. A flow f in N is called a *maximum flow* if $\text{val } f \geq \text{val } f'$ for all flows f' in N .

The central problem of this section is to find a maximum flow in a network.

It turns out that to attack the problem of finding a maximum flow, we need first to introduce the concept of a cut in a network. These two concepts will be related by the so-called Max-Flow Min-Cut Theorem.

As we did in an earlier section of this chapter, if $S \subseteq V$, then we will let \bar{S} denote the complement of S in V ; i.e., $\bar{S} = \{v \in V : v \notin S\}$. The next definition also uses notation introduced in Definition (4.2).

(4.8) DEFINITIONS. A *cut* in N is a set (S, \bar{S}) of arcs such that $S \subseteq V$ and the source x is in S and the sink y is in \bar{S} . If $K = (S, \bar{S})$ is a cut in N , then the *capacity* of K , denoted $\text{cap } K$, is defined by $\text{cap } K = c(S, \bar{S})$. A cut K is called a *minimum cut* if $\text{cap } K \leq \text{cap } K'$ for all cuts K' in N .

► **(4.9) EXAMPLE.** Consider the network of Figure 4.1. To form a cut, we need a subset S of the set of vertices such that $x \in S$ and $y \notin S$. For each of v_1, v_2, \dots, v_6 , we have a choice of either including this vertex in S or excluding this vertex from S . Therefore, the number of cuts in N is $2^6 = 64$.

If we take $S_1 = \{x\}$, then the set $K_1 = (S_1, \bar{S}_1)$ consists of the three arcs (x, v_1) , (x, v_2) , and (x, v_3) , and the capacity of this cut is $c(x, v_1) + c(x, v_2) + c(x, v_3) = 4 + 5 + 2 = 11$. For another example, if $S_2 = \{x, v_1, v_2, v_3\}$, then the set $K_2 = (S_2, \bar{S}_2)$ consists of the arcs (v_1, v_4) , (v_1, v_5) , and (v_2, v_6) . (Note that the arc (v_6, v_3) is not in K_2 since the arrow points the “wrong way.”) The capacity of K_2 is 14. ◀

The reason for the name “cut” is that if the arcs in a cut are deleted from D , then y is no longer reachable from x . It is reasonable then that the value of any flow cannot exceed the capacity of any cut. This is the content of the next result.

(4.10) PROPOSITION. Suppose f is a flow in N and $K = (S, \bar{S})$ is a cut in N . Then

$$\text{val } f = f(S, \bar{S}) - f(\bar{S}, S) \leq \text{cap } K.$$

PROOF. Since $x \in S$ and all other vertices in S are intermediate vertices, we have

$$\sum_{v \in S} (f(v, V) - f(V, v)) = f(x, V) - f(V, x) = \text{val } f.$$

On the other hand, we have

$$\begin{aligned}
 \sum_{v \in S} (f(v, V) - f(V, v)) &= f(S, V) - f(V, S) \\
 &= f(S, S \cup \bar{S}) - f(S \cup \bar{S}, S) \\
 &= f(S, S) + f(S, \bar{S}) - (f(S, S) + f(\bar{S}, S)) \\
 &= f(S, \bar{S}) - f(\bar{S}, S).
 \end{aligned}$$

Hence we have $\text{val } f = f(S, \bar{S}) - f(\bar{S}, S)$. Since $f(S, \bar{S}) \leq \text{cap } K$ and $f(\bar{S}, S) \geq 0$, it follows that $\text{val } f \leq \text{cap } K$. ■

As an immediate consequence of the fact that any flow cannot exceed any cut, we have the following corollary.

(4.11) COROLLARY. Suppose f is a flow and K is a cut in N . If $\text{val } f = \text{cap } K$, then f is a maximum flow and K is a minimum cut.

(4.12) COROLLARY. Suppose f is a flow and $K = (S, \bar{S})$ is a cut in N . If $f(a) = c(a)$ for all $a \in (S, \bar{S})$ and $f(a) = 0$ for all $a \in (\bar{S}, S)$, then f is a maximum flow and K is a minimum cut.

PROOF. The hypotheses imply that $f(S, \bar{S}) - f(\bar{S}, S) = \text{cap } K$. By Proposition (4.10), we have $\text{val } f = f(S, \bar{S}) - f(\bar{S}, S)$. It follows that $\text{val } f = \text{cap } K$ and we are done by Corollary (4.11). ■

From the previous results, we know that the value of a maximum flow is bounded by the capacity of a minimum cut, and if we should happen to find a flow and cut such that the value of the flow is the capacity of the cut, then we are sure that we have a maximum flow and a minimum cut. At this point, it seems that it still might be possible that the value of a maximum flow could be strictly less than the capacity of a minimum cut. However, the Max-Flow Min-Cut Theorem says that the value of a maximum flow is in fact always equal to the capacity of a minimum cut. Before proving this theorem, we will introduce the concept of an f -augmenting semipath. Our presentation follows that of Chartrand and Lesniak [7].

(4.13) DEFINITION. Suppose D is a digraph and $u, v \in V(D)$. A u - v *semipath* P is a finite, alternating sequence

$$P: u = u_0, a_1, u_1, a_2, \dots, u_{n-1}, a_n, u_n = v$$

346

of vertices and arcs of D such that $u_i \neq u_j$ if $i \neq j$ and, for each $i = 1, 2, \dots, n$, the arc a_i is either (u_i, u_{i-1}) or (u_{i-1}, u_i) .

In other words, a u - v semipath is a u - v path in the underlying graph of D , but it may not be a directed path in D since some arcs may point the "wrong way." For example, in the digraph in Figure 4.1, an x - v_6 semipath is $x, (x, v_3), v_3, (v_6, v_3), v_6$.

(4.14) DEFINITION. Let f be a flow in N . A u - v semipath $u = u_0, a_1, u_1, \dots, u_{n-1}, a_n, u_n = v$ in D is said to be f -unsaturated if, for $1 \leq i \leq n$,

- (1) $f(a_i) < c(a_i)$ whenever $a_i = (u_{i-1}, u_i)$ and
- (2) $f(a_i) > 0$ whenever $a_i = (u_i, u_{i-1})$.

An f -unsaturated semipath from the source x to the sink y is called an f -augmenting semipath.

► **(4.15) EXAMPLES.** Let f be the flow in Figure 4.1(a). Then

- 1) The semipath $x, (x, v_3), v_3, (v_6, v_3), v_6$ is not f -unsaturated, since $f(x, v_3) = c(x, v_3)$.
- 2) The semipath $x, (x, v_1), v_1, (v_1, v_4), v_4, (v_4, y), y$ is an f -augmenting semipath (which is actually a directed path in D). ◀

To explain the reason for the name " f -augmenting semipath," we will show that if there exists such a semipath, then we may change the definition of the flow on some of the arcs of D in such a way that the new flow will have a greater value.

So, suppose that f is a flow and that

$$P : x = u_0, a_1, u_2, a_2, \dots, u_{n-1}, a_n, u_n = y$$

is an f -augmenting semipath. If $a_i = (u_{i-1}, u_i)$, then put $s_i = c(a_i) - f(a_i)$. We think of s_i as being the "slack" along a_i ; i.e., it might be possible for s_i more units of the product to flow along a_i . If $a_i = (u_i, u_{i-1})$, then put $s_i = f(a_i)$. Put $s = \min_{1 \leq i \leq n} \{s_i\}$. Define a new flow f^* in N by

$$(4.16) \quad f^*(a) = \begin{cases} f(a) + s & \text{if } a = (u_{i-1}, u_i) \text{ for some } i, 1 \leq i \leq n \\ f(a) - s & \text{if } a = (u_i, u_{i-1}) \text{ for some } i, 1 \leq i \leq n \\ f(a) & \text{if } a \text{ is not an arc of } P. \end{cases}$$

We have called f^* a flow, but the conservation requirement at each intermediate vertex needs to be verified. We leave this as an exercise at the end of this section.

► **(4.17) EXAMPLE.** For the flow f in Figure 4.1(a), the semipath

$$x, (x, v_1), v_1, (v_1, v_4), v_4, (v_4, y), y$$

is an f -augmenting semipath. The "slack" s is 1 for this semipath and the new flow of greater value f^* is the flow in Figure 4.1(b). ◀

The following theorem is due to Ford and Fulkerson [12].

(4.18) THEOREM. A flow f in N is a maximum flow if and only if there is no f -augmenting semipath in D .

PROOF. We have just seen that if there is an f -augmenting semipath, then we may define a flow f^* such that $\text{val } f^* > \text{val } f$. This shows that if f is a maximum flow, then there is no f -augmenting semipath.

Conversely, suppose that there is no f -augmenting semipath. Define a cut (S, \bar{S}) by putting

$$S = \{v \in V : \text{there exists an } f\text{-unsaturated } x\text{-}v \text{ semipath in } D\} \cup \{x\}.$$

Note that $y \notin S$ since there is no f -augmenting semipath. Suppose $a = (v, w) \in (S, \bar{S})$. Then we claim that $f(a) = c(a)$. Indeed, if $f(a) < c(a)$, then by adjoining a and w to an f -unsaturated x - v semipath, we would have an f -unsaturated x - w semipath, and w would be in S , which is a contradiction. By similar reasoning, one can see that if $a = (w, v) \in (\bar{S}, S)$, then we must have $f(a) = 0$. It then follows from Corollary (4.12) that f is a maximum flow. ■

(4.19) THEOREM (Max-Cut Min-Flow Theorem). In any network, the value of a maximum flow equals the capacity of a minimum cut.

PROOF. This follows from the proof of Theorem (4.18), for if f is a maximum flow, then we get a cut with capacity equal to the value of f by taking S to be the set consisting of x and all vertices v such that there is an f -unsaturated x - v path. By Corollary (4.11), this cut must be a minimum cut. ■

It is now not difficult to give an algorithm for determining a maximum flow.

(4.20) MAXIMUM FLOW ALGORITHM

- Step 1. Begin with any flow f in N . (This initial f could even be the flow with $f(a) = 0$ for all a .)
- Step 2. Determine if there exists an f -augmenting semipath. If not, then f is a maximum flow, so stop. 348
- Step 3. If there is an f -augmenting semipath, then define the flow f^* as in (4.16). (Then $\text{val } f^* > \text{val } f$.) Replace f with f^* and go to Step 2.

We have been a little vague at Step 2 of the above algorithm. Of course, one could examine every x - y semipath to see if any of them are f -augmenting, but this would not give an effective algorithm. A good algorithm for implementing the entire search for a maximum flow was given by Edmonds and Karp [11]. This procedure searches for a shortest f -augmenting semipath.

Our treatment again follows that in Chartrand and Lesniak [7]. The algorithm proceeds by a sequence of labelings of the vertices of D . A vertex v receives a label only if there is an f -unsaturated x - v semipath in D . The label given to such a vertex v is an ordered pair as follows: if u is the vertex preceding v on the f -unsaturated x - v semipath P , then the first member of the label on v is $u+$ if the arc preceding v on P is (u, v) and is $v-$ if the arc preceding v on P is (v, u) . The second member of the label on v measures the slack in the semipath up to v . As a vertex receives a label, it is added to a list L .

(4.21) BETTER ALGORITHM FOR DETERMINING A MAXIMUM FLOW

Step 1. Begin with any flow f in N . Label x with the ordered pair $(-, \infty)$ and put $L = \{x\}$.

Step 2. If L is empty, then f is a maximum flow, so stop. If L is nonempty, then select and remove the first element of L , call it v , and suppose the label on v is $(u+, s(v))$ or $(u-, s(v))$ (or $(-, s(x) = \infty)$ if $v = x$).
 a) To all vertices w that are unlabeled and such that $(v, w) \in A(D)$ and $f(v, w) < c(v, w)$, assign the label $(v+, s(w))$ where

$$s(w) = \min\{s(v), c(v, w) - f(v, w)\}$$

and add w to the end of the list L .

b) To all vertices w that are unlabeled and such that $(w, v) \in A(D)$ and $f(w, v) > 0$, assign the label $(v-, s(w))$ where

$$s(w) = \min\{s(v), f(w, v)\}$$

and add w to the end of the list L .

Step 3. If the sink y has been labeled, then go to Step 4; otherwise, go to Step 2.

Step 4. The labels now describe an f -augmenting semipath

$$P: x = u_0, a_1, u_1, a_2, \dots, u_{n-1}, a_n, u_n = y$$

where, for $1 \leq i \leq n$, the vertex u_i is labeled $(u_{i-1}+, s(u_i))$ if $a_i = (u_{i-1}, u_i)$ and is labeled $(u_{i-1}-, s(u_i))$ if $a_i = (u_i, u_{i-1})$. Then,

as in (4.16), replace $f(a_i)$ by $f(a_i) + s(y)$ if $a_i = (u_{i-1}, u_i)$ and by $f(a_i) - s(y)$ if $a_i = (u_i, u_{i-1})$. (All arcs not in P have unchanged f values.) This defines a new flow (still denoted by f) in N .

Step 5. Discard the labels on all vertices except x , put $L = \{x\}$, and go to Step 2.

Again, we leave it as an exercise to see that the new function produced in Step 4 is indeed a flow. It is not hard to see that this algorithm will produce a flow with greater value so long as there exists an augmenting semipath. The algorithm will stop when there is no augmenting semipath, which, by Theorem (4.18), means we have found a maximum flow.

► **(4.22) EXAMPLE.** We will apply Algorithm (4.21) to the network in Figure 4.1 with the initial flow f being the one given in Figure 4.1(a).

- Step 1. We begin with the flow in Figure 4.1(a), we label x with $(-, \infty)$, and we put $L = \{x\}$.
- Step 2. We remove x from L .
- We label v_1 with $(x+, 1)$ and add v_1 to the end of L . We label v_2 with $(x+, 2)$ and add v_2 to the end of L . (Note that there is no slack on the arc (x, v_3) .)
 - There are no vertices v with $(v, x) \in A(D)$.
- Step 3. y has not been labeled, so we return to Step 2.
- Step 2. We remove v_1 from L (so L now consists only of v_2).
- We label v_4 with $(v_1+, 1)$ and add v_4 to the end of L . We label v_5 with $(v_1+, 1)$ and add v_5 to the end of L .
 - Again, 2(b) does not apply.
- Step 3. y is not labeled, so we return to Step 2. (L is now v_2, v_4, v_5 .)
- Step 2. We remove v_2 from L .
- There is no slack on the arc (v_2, v_6) , so we proceed to 2(b).
 - We label v_3 with $(v_2-, 2)$, since $s(v_2) = 2 < f(v_3, v_2)$, and we add v_3 to the end of L .
- Step 2. We remove v_4 from L .
- We label y with $(v_4+, 1)$ and add y to the end of L .
 - Note that v_5 is already labeled, so we proceed to Step 3.
- Step 3. y has been labeled, so we proceed to Step 4.
- Step 4. We have found the f -augmenting semipath

$$x, (x, v_1), v_1, (v_1, v_4), v_4, (v_4, y), y.$$

We replace f by the flow obtained by increasing the value of f by 1 on each of the ("forward") arcs of this semipath. The new flow is then the flow in Figure 4.1(b).

Step 5. We discard all labels except the one on x , put $L = \{x\}$, and proceed to Step 2 with this new flow.

We leave it as an exercise to carry through the steps of the algorithm for this new flow f (the flow in Figure 4.1(b)). The reader should find the f -augmenting semipath $x, (x, v_2), v_2, (v_3, v_2), v_3, (v_6, v_3), v_6, (v_6, y), y$. Using this f -augmenting semipath, we may define a new flow by increasing the value by 1 on the “forward” arcs (x, v_2) and (v_6, y) and by decreasing the value by 1 on the “backward” arcs (v_3, v_2) and (v_6, v_3) . The resulting flow is shown in Figure 4.2. Another pass through the algorithm shows that there are now no augmenting semipaths, so that this flow is a maximum flow with value equal to 10. The corresponding minimum cut (of capacity 10, of course) is (S, \bar{S}) , where $S = \{x, v_2, v_3, v_6\}$. ◀

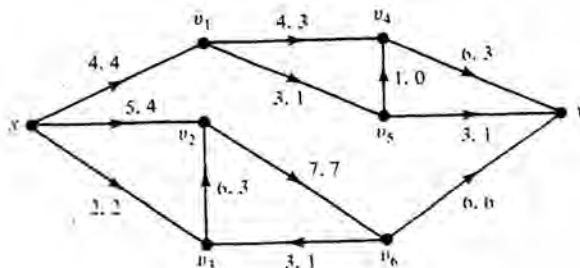


Figure 4.2

EXERCISES

- † 4.1. If a network N has n intermediate vertices, then how many cuts in N are there?
- † 4.2. Find another maximum flow in the network in Figure 4.2 but show that there exists a unique minimum cut in this network.
- 4.3. a) Show that the function f^* defined in (4.16) is indeed a flow.
b) Show that the function defined in Step 4 of Algorithm (4.21) is indeed a flow.
- 4.4. Complete Example (4.22).
- † 4.5. Find all cuts and the capacity of each cut in the network in Figure 4.3. Also find a maximum flow in this network.

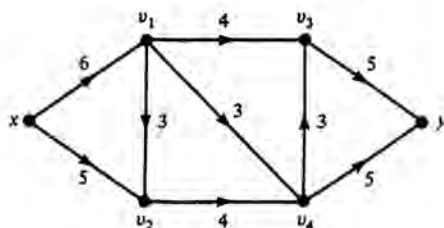


Figure 4.3

- † 4.6. Apply Algorithm (4.21) to the network in Figure 4.4 to find a maximum flow, using the flow given in the figure as your initial flow. Also find a minimum cut.

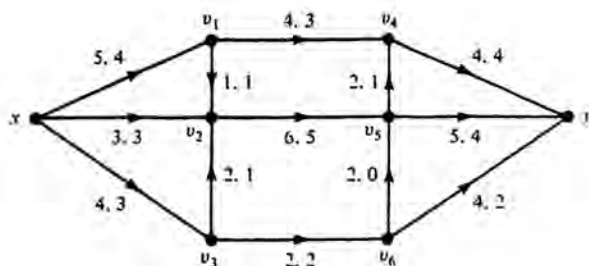


Figure 4.4

- 4.7. Prove the converse of Corollary (4.12) — if f is a maximum flow and $K = (S, \bar{S})$ is a minimum cut, then $f(a) = c(a)$ for all $a \in S$ and $f(a) = 0$ for all $a \in \bar{S}$. (Hint: Use Proposition (4.10) and the Max-Flow Min-Cut Theorem.)
- 4.8. Although we have defined a network to have only one source and one sink, sometimes one may want to consider more general networks that have several sources and sinks. Suppose

$$N' = (D, c, \{x_1, x_2, \dots, x_k\}, \{y_1, y_2, \dots, y_l\})$$

is such a generalized network with k sources x_1, x_2, \dots, x_k and l sinks y_1, y_2, \dots, y_l . Assume that the sets $\{x_1, x_2, \dots, x_k\}$ and $\{y_1, y_2, \dots, y_l\}$

are disjoint. A flow in N' is again an integer-valued function f' on $A(D)$ such that $0 \leq f'(a) \leq c(a)$ for all $a \in A(D)$ and $f'(v, V) = f'(V, v)$ for all intermediate vertices v . Define a new network N , as in this section, by adjoining to N' two vertices x and y and arcs from x to x_i , $1 \leq i \leq k$, and arcs from y_j to y , $1 \leq j \leq l$. On each of these new arcs, let the capacity be ∞ (or some suitably very large integer). Show that there is a 1-1 correspondence between the flows of N' and the flows of N such that if the flow f' in N' corresponds to the flow f in N , then $\text{val } f' = \text{val } f$.

- 4.9.** Let $N = (D, c, x, y)$ be a network such that $c(a) = 1$ for all $a \in A(D)$. Show that:
- The value of a maximum flow in N is equal to the maximum number of x - y directed paths in D such that no two such paths have an arc in common.
 - The capacity of a minimum cut is equal to the minimum number of arcs that must be deleted from D so that y is not reachable from x (via a directed path).
- 4.10.** Let x and y be two vertices of a simple digraph D . Prove that the maximum number of x - y directed paths in D such that no two such paths have an arc in common is equal to the minimum number of arcs that must be deleted from D so that y is not reachable from x . (Hint: Use Exercise 4.9.)
- 4.11.** Let x and y be vertices of a simple graph G . Prove that the number of x - y paths in G such that no two such paths have an edge in common is equal to the minimum number of edges of G that must be deleted so that x and y are not connected. (Hint: Consider the associated digraph of G , as in Exercise 2.5 and use Exercise 4.10.)

References

1. Aigner, Martin. *Combinatorial Theory*. Springer-Verlag, New York, 1979.
2. Bavel, Z. *Introduction to the Theory of Automata*. Reston, Reston, VA, 1983.
3. Blahut, Richard E. *Theory and Practice of Error Control Codes*. Addison-Wesley, Reading, MA, 1983.
4. Bollobàs, Bela. *Graph Theory: An Introductory Course*. Springer-Verlag, New York, 1979.
5. Bondy, J.A. and U.S.R. Murty. *Graph Theory with Applications*. North-Holland, New York, 1976.
6. Brualdi, Richard. *Introductory Combinatorics*. North-Holland, New York, 1977.
7. Chartrand, Gary and Linda Lesniak. *Graphs and Digraphs*, 2nd ed. Wadsworth, Belmont, CA, 1986.
8. Davis, Philip J. and Reuben Hersh. *The Mathematical Experience*. Houghton Mifflin, Boston, 1981.
9. Diffie, Whitfield and Martin E. Hellman. "New Directions in Cryptography." *IEEE Trans. Inform. Theory* IT-22, No. 6 (1976), 135-145.
10. Dornhoff, Larry L. and Franz E. Hohn. *Applied Modern Algebra*. Macmillan, New York, 1978.
11. Edmonds, J. and R.M. Karp. "Theoretic Improvements in Algorithmic Efficiency for Network Flow Problems." *J. Assoc. Comput. Mach.* 19 (1972), 248-264.
12. Ford, L.R., Jr. and D.R. Fulkerson. "Maximal Flow Through a Network." *Canad. J. Math.* 8 (1956), 399-404.
13. Harary, F. and E.M. Palmer. *Graphical Enumeration*. Academic Press, New York, 1973.
14. Harrison, M.A. *Introduction to Switching and Automata Theory*. McGraw-Hill, New York, 1966.

15. Hartmanis, J. and R.E. Stearns. *Algebraic Structure of Sequential Machines*. Prentice-Hall, Englewood Cliffs, NJ, 1966.
16. Herstein, I.N. *Topics in Algebra*, 2nd ed. Wiley, New York, 1975.
17. Hill, Raymond. *A First Course in Coding Theory*. Oxford Univ. Press, New York, 1986.
18. Knuth, Donald E. *The Art of Computer Programming, Vol. 2*. Addison-Wesley, Reading, MA, 1969.
19. Lang, Serge. *Undergraduate Algebra*. Springer-Verlag, New York, 1987.
20. Lenstra, H.W. "Factoring Integers with Elliptic Curves." *Annals of Math.* 126 (1987), 649-673.
21. Lidl, Rudolf and Günter Pilz. *Applied Abstract Algebra*. Springer-Verlag, New York, 1984.
22. Mackiw, George. *Applications of Abstract Algebra*. Wiley, New York, 1985.
23. MacWilliams, F.J. and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.
24. Mendelson, Elliot. *Schaum's Outline of Theory and Problems of Boolean Algebra and Switching Circuits*. McGraw-Hill, New York, 1970.
25. Miller, William. "Even and Odd Permutations." *Mathematics Association of Two-year Colleges* 5 (1971), p. 32.
26. Pless, Vera. *Introduction to the Theory of Error-Correcting Codes*. Wiley-Interscience, New York, 1982.
27. Pomerance, Carl. "The Search for Prime Numbers." *Scientific American*, December 1982, 136-147.
28. Rivest, R.L., A. Shamir, and L. Adelman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." *Comm. ACM* 21 (1978), 120-126.
29. Roberts, Fred. *Applied Combinatorics*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

Chapter I

I.1

- 1.1. a) $S \cap T = \{3, 6, 9\}$. b) Explicitly write out $S \cup T$ or use Proposition (1.1).
- 1.2. $B^2 = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ and $\mathcal{P}(B) = \{\emptyset, \{0\}, \{1\}, B\}$.
- 1.3. If $S = T$, then clearly $S \subseteq T$ and $T \subseteq S$. Conversely, if $S \subseteq T$ and $T \subseteq S$, then every element of S is in T and every element of T is in S , which implies that S and T have exactly the same elements.
- 1.4. Suppose $S \subseteq T$. Then if $x \in S \cup T$, we would have $x \in T$, hence $S \cup T \subseteq T$. The fact that $T \subseteq S \cup T$ follows immediately from the definitions. Conversely, suppose $S \cup T = T$. Then, since S is obviously a subset of $S \cup T$, we may conclude that $S \subseteq T$.
- 1.6. We use Exercise 1.3. Suppose $x \in A \cap (B \cup C)$. Then $x \in A$ and either $x \in B$ or $x \in C$. Hence either $x \in (A \cap B)$ or $x \in (A \cap C)$. Conversely, suppose either $x \in (A \cap B)$ or $x \in (A \cap C)$. Then x must be in A and x is also either in B or in C . Thus $x \in A \cap (B \cup C)$.
- 1.8. a) $(x, y) \in (A \times B) \cap (A \times C) \Leftrightarrow x \in A \text{ and } y \in (B \cap C) \Leftrightarrow (x, y) \in A \times (B \cap C)$.
- 1.13. Write $A \cup B \cup C = A \cup (B \cup C)$ and apply Proposition (1.1) and Exercise 1.6.
- 1.14. Use the Pigeonhole Principle with the 12 months as the pigeonholes and the people as the pigeons.
- 1.16. Assume each pigeonhole contains no more than $\lfloor (n-1)/m \rfloor$ pigeons. Since there are m pigeonholes, there could then be at most $(n-1)$ pigeons, a contradiction.

1.2

- 2.1. The formula is clearly true for $n = 1$. Assume the formula true for $n = k$. We must now prove it's true for $n = k + 1$. We have

$$\begin{aligned} 1^3 + \cdots + k^3 + (k+1)^3 &= (1^3 + \cdots + k^3) + (k+1)^3 \\ &= [k(k+1)/2]^2 + (k+1)^3 \\ &= (k+1)^2[k^2/4 + k + 1] \\ &= (k+1)^2[(k^2 + 4k + 4)/4] = [(k+1)(k+2)/2]^2. \end{aligned}$$

- 2.4. a) To establish the $n = 1$ case, we must show that $F_2^2 - F_1 F_3 = -1$. We have $F_1 = 1, F_2 = 1$, and $F_3 = 2$ and indeed $1 - 2 = -1$. Now we assume the formula true for $n = k$, and we must prove it true for $n = k + 1$. We have

$$\begin{aligned} F_{k+2}^2 - F_{k+1} F_{k+3} &= F_{k+2}^2 - F_{k+1}(F_{k+2} + F_{k+1}) \\ &= F_{k+2}^2 - F_{k+1} F_{k+2} - F_{k+1}^2 \\ &= F_{k+2}(F_{k+2} - F_{k+1}) - F_{k+1}^2 \\ &= F_{k+2} F_k - F_{k+1}^2 = -(F_{k+1}^2 - F_k F_{k+2}) \\ &= -(-1)^k = (-1)^{k+1}. \end{aligned}$$

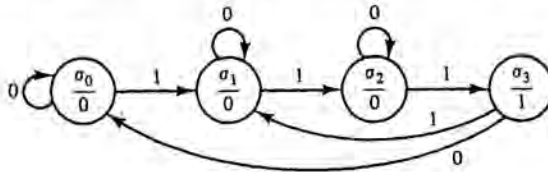
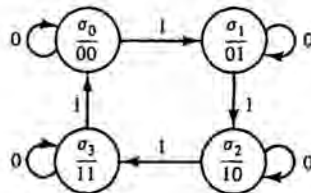
- b) Note that since $F_{3n} = F_{3n-1} + F_{3n-2}$, once we show that F_{3n-1} and F_{3n-2} are odd for all $n \geq 1$, we can conclude that F_{3n} is even for all $n \geq 1$ (since odd + odd = even). The $n = 1$ case amounts to verifying that F_2 and F_1 are both odd, and this is true since $F_2 = F_1 = 1$. Now assume that F_{3k-2} and F_{3k-1} are both odd. We must show that F_{3k+1} and F_{3k+2} are both odd. Since $F_{3k} = F_{3k-1} + F_{3k-2}$, we know that F_{3k} is even. Then $F_{3k+1} = F_{3k} + F_{3k-1}$ is odd since it's the sum of an even and an odd, and similarly $F_{3k+2} = F_{3k+1} + F_{3k}$ is also odd.

- 2.6. a) $q = 7, r = 2$. b) $q = -8, r = 3$. c) $q = 6, r = 0$. d) $q = r = 0$.
 2.7. There exist $k, l \in \mathbb{Z}$ such that $a = dk, b = dl$. Then $c = d(l + k)$.
 2.9. Since $F_5 = 5$, the statement is true for $n = 1$. Assume $5 | F_{5k}$. We have $F_{5k+5} = F_{5k+4} + F_{5k+3} = 2F_{5k+3} + F_{5k+2} = 3F_{5k+2} + 2F_{5k+1} = 5F_{5k+1} + 3F_{5k}$. Now use Exercise 2.7.
 2.12. a) $6 = 1 \cdot 138 - 11 \cdot 12$. b) $6 = 1 \cdot 138 + 11 \cdot (-12)$. c) $10 = 5 \cdot 210 - 8 \cdot 130$.
 d) $1 = 4 \cdot 64 - 15 \cdot 17$. e) $1 = 461 \cdot 1211 - 90 \cdot 6203$.
 2.13. The only positive divisors of p are 1 and p , so the only possibilities for (a, p) are 1 and p . If p does not divide a , then (a, p) must be 1.
 2.15. We must show that the properties in Definition (2.8) are satisfied. Put $d = p_1^{\min\{s_1, t_1\}} \cdots p_r^{\min\{s_r, t_r\}}$. Then it is not hard to see that $d | a$ and $d | b$ (since $p_i^m | p_i^n$ if $m \leq n$). Now suppose $d' | a$ and $d' | b$. Express d' as a product of primes. Then each prime in this factorization of d' divides a and b , so by Corollary (2.20), no prime other than p_1, p_2, \dots, p_r can occur in the prime factorization of d' . Also, if $p_i^{m_i}$ occurs in the prime factorization of d' , then $p_i^{3s_i} | p_i^{s_i}$ and $p_i^{m_i} | p_i^{t_i}$, so we must have $m_i \leq s_i$ and $m_i \leq t_i$. But then $p_i^{m_i} | p_i^{\min\{s_i, t_i\}}$. It follows that $d' | d$.

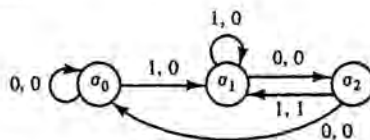
- 2.17.** Write each of the m_i 's and n as a product of primes. Since the m_i 's are pairwise relatively prime, no prime can occur simultaneously in the prime factorizations of m_i and m_j if $i \neq j$. Thus $m_1 \cdots m_k | n$ if and only if whenever $p_i^{s_i}$ occurs in one of the prime factorizations of the m_i 's, then $p_i^{t_i}$, for $t_i \geq s_i$, occurs in the prime factorization of n .
- 2.18.** The proof of part (c) is similar to the proof in Exercise 2.15. For part (d), one can use part (c) and Proposition (2.22).

1.3

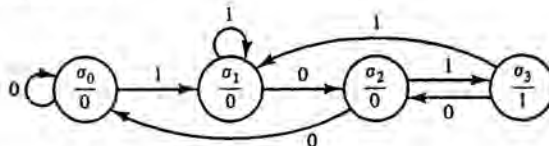
- 3.1.** a) 001010. b) 0110011. c) This machine delays each digit in the input string by one place when it produces an output string.

3.2.**3.3.**

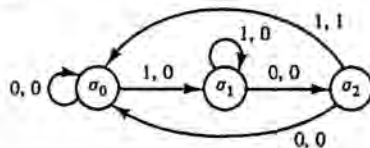
- 3.4. a)** Mealy machine:



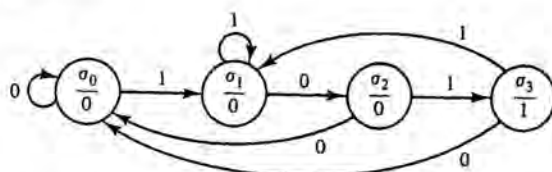
Moore machine:



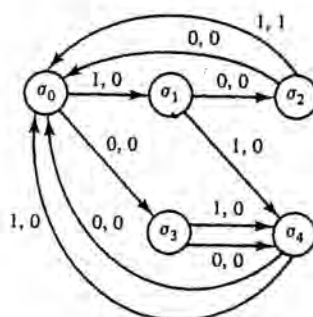
- b)** Mealy machine:



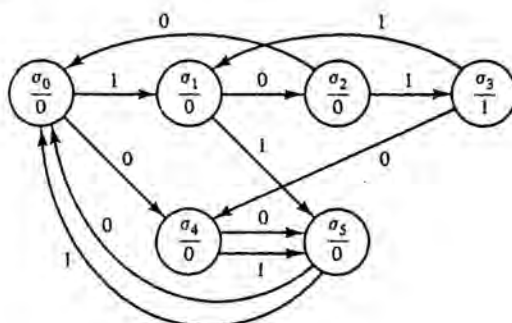
Moore machine:



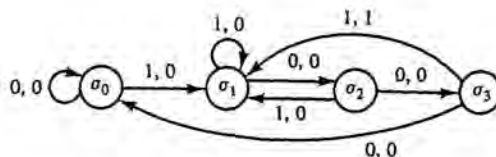
3.5. Mealy machine:



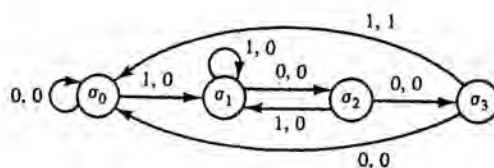
Moore machine:



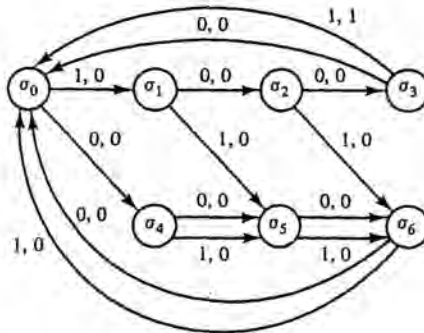
3.6. a)



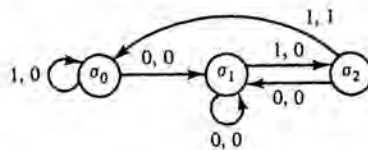
b)



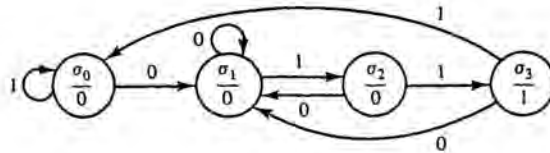
3.7.



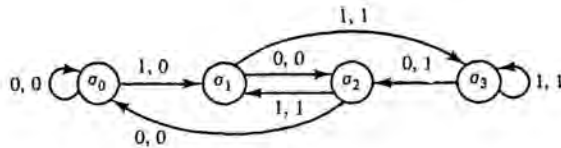
3.8. Mealy machine:



Moore machine:



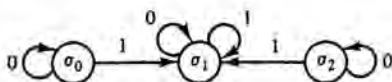
3.9.



Chapter II

II.1

- 1.2. There are several possible correct answers. We give one correct answer for each part.
- a) $R = \{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1), (1, 3), (3, 1)\}$. Then we have $(2, 1) \in R$ and $(1, 3) \in R$, but $(2, 3) \notin R$.
- b) $R = \{(1, 1), (2, 2), (3, 3), (1, 2)\}$.
- 1.3. Since $0 = n - n$ is even, R is reflexive. R is symmetric because if $n - m$ is even, then $m - n = -(n - m)$ is also even. R is transitive, since if $n - m$ is even and if $m - l$ is even, then $n - l = (n - m) + (m - l)$ is even.
- 1.4. R is symmetric, but neither reflexive, nor transitive.
- 1.5. a) (We omit outputs in this machine since they are irrelevant to this problem.)



- 1.6. Suppose $f(a) = f(b)$. Then $a^3 = b^3$, which implies that $a = b$.
- 1.7. Suppose $b \in \mathbb{Z}$. Then $b = f(b-1)$.
- 1.8. If we let the elements of B be the pigeonholes and the elements of A be the pigeons, then this is just a restatement of part (3) of the Pigeonhole Principle (I.1.9). (We think of the mapping f as assigning each pigeon to some hole.)
- 1.9. Since f is onto, every $b \in B$ occurs as the first member of an ordered pair in R . Suppose (b, a_1) and (b, a_2) are both in R . We must show that $a_1 = a_2$. By definition of R , we have $b = f(a_1)$ and $b = f(a_2)$. Since f is 1-1, this implies that $a_1 = a_2$.
- 1.10. $((f \circ g) \circ h)(s) = f(g(h(s))) = (f \circ (g \circ h))(s)$.
- 1.11. Let $A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$, $B = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}$, $C = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$. Show explicitly that $A(BC) = (AB)C$. Matrix multiplication is not commutative since, for example, $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ while $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$.
- 1.12. Yes. (One can use (I.2.22).)
- 1.13. $(n \circ m) \circ l = (n + m - nm) \circ l = n + m - nm + l - (n + m - nm)l = n + m + l - nm - nl - ml + nml = n \circ (m \circ l)$.
- 1.14. This operation is clearly commutative from inspection of the table. It can also be seen to be associative by checking the possible cases.

II.2

2.1. a) Transitive, not reflexive, not symmetric, $M_R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

b) Reflexive, symmetric, not transitive, $M_R = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$.

c) Reflexive, symmetric, transitive, $M_R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$.

d) Reflexive, symmetric, transitive, $M_R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

e) Symmetric, not reflexive, not transitive, $M_R = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$.

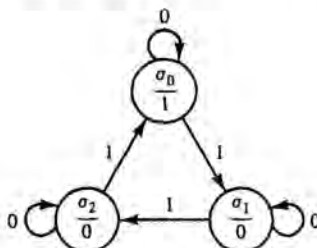
- 2.2. a) There are n^2 entries in an $n \times n$ matrix and each entry may be 0 or 1.
 b) If the relation is reflexive, then the entries on the main diagonal of the matrix of the relation must be 1's. Thus we have $n^2 - n$ "free" entries, which each may be 0 or 1.
 c) If the relation is symmetric, then its matrix must be symmetric. Therefore, once the upper right triangle of the matrix is filled, the lower left triangle of the matrix is determined. Hence, there are n "free" entries in the first row, $n - 1$ free entries in the second row, and so on up to 1 free entry in the last row. The total number of free entries is $1 + 2 + \cdots + n = n(n + 1)/2$, and each of these free entries may be 0 or 1.

II.3

- 3.1. a) Not symmetric.
 b) An equivalence relation, $[a] = \{a, -a\}$.
 c) An equivalence relation, $[a] = \{a, -(a + 1)\}$.
 d) Not reflexive, not transitive.
 e) Not transitive.
 f) An equivalence relation with the two equivalence classes $\{1, 2\}$ and $\{3, 4\}$.
 g) Not reflexive.
- 3.2. a) $R = \{(1, 1), (1, 4), (1, 5), (4, 1), (4, 4), (4, 5), (5, 1), (5, 4), (5, 5), (2, 2), (2, 3), (3, 2), (3, 3)\}$.
 b) $R = \{(1, 1), (2, 2), (3, 3), (3, 4), (4, 3), (4, 4), (5, 5)\}$.
- 3.4. Since $n|0$, we have $a \equiv_n a$ for every $a \in \mathbf{Z}$, showing that this relation is reflexive. Suppose $a \equiv_n b$. Then $(b - a) = nk$ for some $k \in \mathbf{Z}$. Hence $(a - b) = n(-k)$, so $n|(a - b)$ and $b \equiv_n a$. Finally, suppose $a \equiv_n b$ and $b \equiv_n c$. Then there exist $k, l \in \mathbf{Z}$ such that $(b - a) = nk$ and $(c - b) = nl$. Adding these equations, we obtain $(c - a) = n(k + l)$, so $n|(c - a)$ and $a \equiv_n c$.
- 3.6. The question asks: How many different partitions are there of a four-element set? There is the partition consisting only of the set itself; the partition consisting of the four singleton subsets; four partitions each consisting of one singleton subset and one subset containing the other three elements; $\binom{4}{2} = 6$ partitions each consisting of one subset containing two elements and two singleton subsets; and three partitions each consisting of two subsets containing two elements each. Thus there are 15 distinct equivalence relations on a four-element set.

II.4

- 4.3. The minimal machine equivalent to M would have just one state with the output associated to every state of M as its associated output and with every input keeping the machine in this state.
- 4.4. One example is



- 4.5. a) $\Pi_0 = \{\{\sigma_0, \sigma_3\}, \{\sigma_1, \sigma_2\}\}$. But then σ_0 and σ_1 are not 1-equivalent, since $\tau(\sigma_0, 1) = \sigma_0$ and $\tau(\sigma_3, 1) = \sigma_2$. Also, σ_1 and σ_2 are not 1-equivalent since $\tau(\sigma_1, 0) = \sigma_2$ and $\tau(\sigma_2, 0) = \sigma_0$. Therefore,

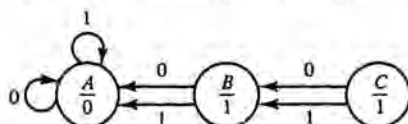
$$\Pi_1 = \{\{\sigma_0\}, \{\sigma_1\}, \{\sigma_2\}, \{\sigma_3\}\}.$$

Hence this machine is minimal and $\tilde{M} = M$.

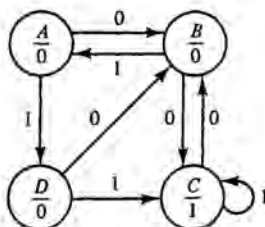
- b) Π_0 is the same here as in part (a) above. But here, $\Pi_1 = \Pi_0$. Hence the algorithm stops. The quotient machine is the same as the one in Figure 4.4.
- c) Π_0 is the same as in parts (a) and (b). But here,

$$\Pi_1 = \{\{\sigma_0, \sigma_3\}, \{\sigma_1\}, \{\sigma_2\}\}.$$

If we put $A = \{\sigma_0, \sigma_3\}$, $B = \{\sigma_2\}$, and $C = \{\sigma_1\}$, then the quotient machine looks like



- d) $\Pi_0 = \{\{\sigma_0, \sigma_3, \sigma_4, \sigma_6\}, \{\sigma_1, \sigma_2, \sigma_5\}\}$. Now, σ_0 is not 1-equivalent to σ_3 or σ_4 since $\tau(\sigma_0, 0) = \sigma_3$ and $\tau(\sigma_3, 0) = \tau(\sigma_4, 0) = \sigma_2$. Also, σ_0 is not 1-equivalent to σ_6 since $\tau(\sigma_0, 1) = \sigma_6$ and $\tau(\sigma_6, 1) = \sigma_2$. It is easy to see that σ_3 and σ_4 are 1-equivalent and neither is 1-equivalent to σ_6 . It is also easy to see that σ_1, σ_2 , and σ_5 are all 1-equivalent. Therefore, $\Pi_1 = \{\{\sigma_0\}, \{\sigma_6\}, \{\sigma_3, \sigma_4\}, \{\sigma_1, \sigma_2, \sigma_5\}\}$. It is straightforward to see that $\Pi_2 = \Pi_1$. If we put $A = \{\sigma_0\}$, $B = \{\sigma_3, \sigma_4\}$, $C = \{\sigma_1, \sigma_2, \sigma_5\}$, and $D = \{\sigma_6\}$, then the quotient machine looks like



- 4.7. M_1 and M_2 must have the same number of states and are really the same machine except for possible relabeling of states.
- 4.8. M_1 and M_2 cannot be equivalent. If they were, then a minimal machine with two states would be equivalent to a minimal machine with three states, which is a contradiction.

Chapter III

III.1

1.1.

Element of S_3	ϵ	σ	σ^2	τ	$\sigma\tau$	$\tau\sigma$
Least r	1	3	3	2	2	2

1.2. a) We can verify closure by forming the following table:

	ϵ	σ	σ^2
ϵ	ϵ	σ	σ^2
σ	σ	σ^2	ϵ
σ^2	σ^2	ϵ	σ

- b) $\{\epsilon, \tau\}, \{\epsilon, \sigma\tau\}, \{\epsilon, \tau\sigma\}$.
 c) $(\sigma\tau)(\tau\sigma) = \sigma^2$.

1.4.

Element of D_4	ϵ	α	α^2	α^3	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$
Least r	1	4	2	4	2	2	2	2

1.5. a) From Table (1.8), we have the following table:

	ϵ	α^2	β	$\alpha^2\beta$
ϵ	ϵ	α^2	β	$\alpha^2\beta$
α^2	α^2	ϵ	$\alpha^2\beta$	β
β	β	$\alpha^2\beta$	ϵ	α^2
$\alpha^2\beta$	$\alpha^2\beta$	β	α^2	ϵ

- b) $\{\epsilon, \alpha, \alpha^2, \alpha^3\}$ or $\{\epsilon, \alpha^2, \alpha\beta, \alpha^3\beta\}$.
- 1.8. For each letter, there are n choices. Therefore, there are n^r words of length r .
- 1.9. a) Call the relation R . If \hat{x} is a string, then we clearly have $\hat{x}R\hat{x}$. If $\hat{x}R\hat{y}$, then obviously we also have $\hat{y}R\hat{x}$. Finally, if $\hat{x}R\hat{y}$ and $\hat{y}R\hat{z}$, then \hat{x} and \hat{z} have the same parity (since both have the same parity as \hat{y}). Therefore, we have $\hat{x}R\hat{z}$.
- b) Suppose \hat{x}_1 and \hat{x}_2 are both even. If \hat{y}_1 and \hat{y}_2 are both even, then $\hat{x}_1\hat{y}_1$ and $\hat{x}_2\hat{y}_2$ are both even. If \hat{y}_1 and \hat{y}_2 are both odd, then $\hat{x}_1\hat{y}_1$ and $\hat{x}_2\hat{y}_2$ are both odd. A similar argument applies if \hat{x}_1 and \hat{x}_2 are both odd.
- c) No. If the input string is 100, then the parity check machine will output 111, while if the input string is 111, then the parity check machine will output 101.

III.2

- 2.1. a) Group.
 b) Semigroup, but not a monoid.
 c) Monoid, but not a group.
 d) Semigroup, but not a monoid.
 e) Subtraction is not associative, so this is not even a semigroup.
- 2.3. The rows and columns of the multiplication table must form a symmetric matrix.
- 2.4. Suppose b and c are both inverses of a and let e denote the identity in the monoid. Then $b = be = b(ac) = (ba)c = ec = c$.
- 2.5. Using Proposition (2.8), the table must be

	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

This is an abelian group. We have $a^2 = b, a^3 = e, a^{-1} = b$.

- 2.7. Suppose $a, b \in G$. Since $(ab)^2 = a^2b^2$, we have $abab = aabb$. Since G is a group, we may cancel an a from each side on the left and a b from each side on the right, yielding $ba = ab$.
- 2.8. $-1 = 3$ in this group. The table is

$+_4$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

- 2.9. $2^2 = 0$ in this monoid. The table is

\cdot_4	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

- 2.14. Suppose $ab = e$. Then, since $a^{-1} \in A$, we may multiply on the left on each side of this equation by a^{-1} , obtaining $b = a^{-1}$. If $ba = e$, then we multiply on the right by a^{-1} .
- 2.15. If $g^2 = e$ for all $g \in G$, then $g = g^{-1}$ for all $g \in G$. Then if $a, b \in G$, we have $ab = (ab)^{-1} = b^{-1}a^{-1} = ba$.

III.3

3.4. The multiplication table for S is

\circ	f_1	f_2	f_3	f_4
f_1	f_1	f_2	f_3	f_4
f_2	f_2	f_1	f_4	f_3
f_3	f_3	f_3	f_3	f_3
f_4	f_4	f_4	f_4	f_4

Note that S is actually a monoid, with f_1 being the identity. The subsemigroups of S are $\{f_1\}$, $\{f_3\}$, $\{f_4\}$, $\{f_1, f_2\}$, $\{f_1, f_3\}$, $\{f_1, f_4\}$, $\{f_3, f_4\}$, and S itself.

- 3.5. Suppose H_1 and H_2 are subgroups of G . We will show that $H_1 \cap H_2$ satisfies the criterion in Proposition (3.8). Suppose $a, b \in H_1 \cap H_2$. Then $a, b \in H_1$ and $a, b \in H_2$. Since H_1 and H_2 are subgroups, we have $ab^{-1} \in H_1$ and $ab^{-1} \in H_2$. Therefore, $ab^{-1} \in H_1 \cap H_2$.
- 3.6. Consider the subgroups $\{\epsilon, \sigma, \sigma^2\}$ and $\{\epsilon, \tau\}$ of S_3 . The union of these two subgroups is not closed under product since $\sigma\tau$ is not in the union.
- 3.7. $(\mathbf{Z}, +)$ is a cyclic group (generated by either 1 or -1). Thus by Proposition (3.19) every subgroup must be cyclic. Hence any subgroup is of the form $(n) = \{mn : m \in \mathbf{Z}\}$ for some $n \in \mathbf{Z}$.
- 3.8. By the Division Algorithm, we may write $k = mq + r$ where $0 \leq r < m$. But then $g^r = g^{k-mq} = g^k(g^m)^{-q} = ee^{-q} = e$. Hence r must be 0, since $r < m$ and m is the order of g .
- 3.9. $(\epsilon) = \{\epsilon\}$, $(\alpha) = \{\epsilon, \alpha, \alpha^2, \alpha^3\} = (\alpha^3)$, $(\alpha^2) = \{\epsilon, \alpha^2\}$, $(\beta) = \{\epsilon, \beta\}$, $(\alpha\beta) = \{\epsilon, \alpha\beta\}$, $(\alpha^2\beta) = \{\epsilon, \alpha^2\beta\}$, $(\alpha^3\beta) = \{\epsilon, \alpha^3\beta\}$.
- 3.10. Suppose $m|n$. Then $n = mk$ for some $k \in \mathbf{P}$. The cyclic subgroup generated by g^k is then $\{g^k, g^{2k}, \dots, g^{mk} = g^n = e\}$. These m elements must be distinct; for if $g^{ik} = g^{jk}$ for $1 \leq i < j \leq m$, then we would have $g^{(j-i)k} = e$. But $(j-i)k < mk = n$, so that would contradict the fact that the order of g is n . Conversely, suppose H is a subgroup of G of order m . Then H must be cyclic by Proposition (3.19). If g' is a generator of H , then $o(g') = m$ by Proposition (3.17). Now use Exercise 3.8.
- 3.11. Put $l = o(g^t)$. Since $(g^t)^m = (g^m)^t = e$, we have that $l|m$ by Exercise 3.8. Since $g^{tl} = (g^t)^l = e$, we also have that $m|tl$. Since m and t are relatively prime and $m|tl$, we have $m|l$ by Proposition (I.2.18). Since $l|m$ and $m|l$, we must have $m = l$.
- 3.13.

Element	0	1	2	3	4	5	6	7	8	9	10	11
Order	1	12	6	4	3	12	2	12	3	4	6	12

From the table, or from Example (3.13.2), we see that the generators of \mathbf{Z}_{12} are 1, 5, 7, and 11. Every subgroup must be cyclic, and the distinct cyclic subgroups are (0) , $(2) = \{0, 2, 4, 6, 8, 10\}$, $(3) = \{0, 3, 6, 9\}$, $(4) = \{0, 4, 8\}$, $(6) = \{0, 6\}$, and \mathbf{Z}_{12} itself.

III.4

- 4.1. The left cosets of H are $\{\epsilon, \sigma\tau\}$, $\{\sigma, \tau\sigma\}$, and $\{\tau, \sigma^2\}$. The right cosets of H are $\{\epsilon, \sigma\tau\}$, $\{\sigma, \tau\}$, and $\{\sigma^2, \tau\sigma\}$.
- 4.2. The left cosets are the same as the right cosets. They are H and $\{\beta, \alpha\beta, \alpha^2\beta, \alpha^3\beta\}$.
- 4.4. The order of a subgroup must divide 8, so the possible orders are 1, 2, 4, and 8. A subgroup of order 1 is $\{\epsilon\}$, a subgroup of order 2 is $\langle\beta\rangle$, a subgroup of order 4 is $\langle\alpha\rangle$, and a subgroup of order 8 is D_4 itself.
- 4.5. The order of each element of G must divide 4, hence the possible orders are 1, 2, and 4. If there is an element of order 4, then it generates G and G is cyclic. If not, then every element not equal to the identity has order 2.
- 4.6. We have $H = \{e, g^3, g^6, g^9\}$. The cosets (left and right cosets are equal here) are H , $\{g, g^4, g^7, g^{10}\}$, and $\{g^2, g^5, g^8, g^{11}\}$.
- 4.8. A proper subgroup of a group of order 15 can have order 1, 3, or 5. Now apply Corollary (4.11).
- 4.9. If $a \in H \cap K$, then $o(a)$ must divide $|H|$ and $|K|$ by Corollary (4.9). The only positive divisor of 7 and 12 is 1.
- 4.10. If $g \in H$, then clearly $g^2 \in H$. Suppose $g \notin H$. Then $gH \neq H$. Also, $g^2H \neq gH$ or else we would have $g^2 = gh$ for some $h \in H$, which would imply $g = h$. Since $g^2H \neq gH$ and there are only two distinct left cosets of H , we must have $g^2H = H$. Then $g^2 \in H$.
- 4.11. We have $[G : K] = |G|/|K|$, $|G| = |H|[G : H]$, and $|H| = |K|[H : K]$. In D_4 , consider $K = \{\epsilon, \alpha^2\}$ and $H = \{\epsilon, \alpha, \alpha^2, \alpha^3\}$.

III.5

- 5.1. a) If we let R denote the equivalence relation corresponding to this partition, then we have $0R1$ and $3R4$, but $0 \cdot_6 3 = 0$ is not equivalent to $1 \cdot_6 4 = 4$.
- b) Let R_1 be the equivalence relation corresponding to the partition $S_1 = \{0, 2, 4\}$, $S_2 = \{1, 3, 5\}$. Note that the product of any two elements in S_1 is in S_1 , the product of any two elements in S_2 is in S_2 , and the product of an element in S_1 and an element in S_2 is in S_1 .
- c) Let R_2 be the equivalence relation corresponding to the partition $S_1 = \{0, 3\}$, $S_2 = \{1, 5\}$, $S_3 = \{2, 4\}$. Each of these sets is closed under \cdot_6 , the product of an element in S_1 with an element in either S_2 or S_3 is in S_1 , and the product of an element in S_2 with an element in S_3 is in S_3 .
- 5.3. a) No matter in which state you start, you end in σ_1 if the input string ends in 0 and in σ_2 if the input string ends in 1.
- b) There are two elements in this semigroup, $[0]$ and $[1]$. The equivalence class $[0]$ is the class of all inputs that end in 0 and the transition mapping T_0 associated to any such input is $\sigma_1 \mapsto \sigma_1, \sigma_2 \mapsto \sigma_1$. The equivalence class $[1]$ is the class of all inputs ending in 1 and the transition mapping T_1 associated to any such input is $\sigma_1 \mapsto \sigma_2, \sigma_2 \mapsto \sigma_2$. Since $T_0 \diamond T_1 = T_1$ and $T_1 \diamond T_0 = T_0$, the multiplication table for the semigroup of this machine is

then

	[0]	[1]
[0]	[0]	[1]
[1]	[0]	[1]

This is not a group; in fact, it is not even a monoid since there is no identity.

- 5.4. The two distinct left cosets of H must be H and the elements in G that are not in H , since the left cosets partition G . The right cosets must be the same as the left cosets (since one right coset is also H and the two distinct right cosets partition G).
- 5.5. a) Suppose H is normal and suppose $g \in G$ and $h \in H$. Then $gh \in gH = Hg$, so we have $gh = h'g$ for some $h' \in H$. Thus $ghg^{-1} = h' \in H$. Conversely, suppose $ghg^{-1} \in H$ for all $g \in G$ and $h \in H$. Then given $gh \in gH$, we have $ghg^{-1} = h' \in H$. Hence $gh = h'g \in Hg$, which shows that $gH \subseteq Hg$. Similarly, if $hg \in Hg$, then $g^{-1}h(g^{-1})^{-1} = g^{-1}hg = h'' \in H$. Hence $hg = gh'' \in gH$.
- 5.6. First, one must show that H is a subgroup of G — use Proposition (3.8) to do this. Next, if $A = \begin{bmatrix} a & b \\ 0 & d \end{bmatrix} \in G$, then an arbitrary element of HA looks like $\begin{bmatrix} 1 & b_1 \\ 0 & 1 \end{bmatrix} A = \begin{bmatrix} a & b+b_1d \\ 0 & d \end{bmatrix}$. This matrix also equals $A \begin{bmatrix} 1 & b_1d/a \\ 0 & 1 \end{bmatrix}$. Hence every element in HA is in AH . Similarly, one can show that every element in AH is in HA . One may also use Exercise 5.5(a).
- 5.7. Obviously, $\{\epsilon\}$ and D_4 itself are normal subgroups. By Exercise 5.4, any subgroup of index 2 is normal, so the subgroups $\langle \alpha \rangle$ and $\langle \epsilon, \beta, \alpha^2, \alpha^2\beta \rangle$ are normal. Also, since α^2 commutes with every element, the subgroup $\langle \epsilon, \alpha^2 \rangle$ is normal. The other subgroups of D_4 of order 2 are not normal.
- 5.8. Let $K = \langle \epsilon, \beta \rangle$, $H = \langle \epsilon, \beta, \alpha^2, \alpha^2\beta \rangle$. Then K is normal in H since K has index 2 in H , and H is normal in D_4 since H has index 2 in D_4 , but note that K is not normal in D_4 (for example, $\alpha K \neq K\alpha$).
- 5.9. First note that H is normal in G since G is abelian. Now, if aH and bH are in G/H , then $aHbH = (ab)H = (ba)H$ (since G is abelian) and $(ba)H = bHaH$.
- 5.10. Note that $(aH)^{o(a)} = (a^{o(a)})H = eH = H$, the identity in G/H . It follows from Exercise 3.8 that the order of aH must then divide $o(a)$. An example of this situation would be: take $G = D_4$, $H = \langle \epsilon, \alpha^2, \beta, \alpha^2\beta \rangle$, $a = \alpha$. Then $o(\alpha) = 4$ while the order of αH in G/H is 2 (since $\alpha^2 \in H$).
- 5.11'. First, note that a cyclic group is abelian, so any subgroup is normal. Let g be a generator for G . Then any element in G is of the form g^r for some r . Hence any coset in G/H is of the form g^rH for some r . Since $g^rH = (gH)^r$, it follows that gH generates G/H .
- 5.12. Suppose H_1 and H_2 are normal subgroups of G and put $K = H_1 \cap H_2$. By Exercise 3.5, we know that K is a subgroup of G . If $g \in G$ and $k \in K$, then by Exercise 5.5(a), we have $gkg^{-1} \in H_1$ and $gkg^{-1} \in H_2$. Hence $gkg^{-1} \in K$ and K is normal, again by Exercise 5.5(a).

III.6

- 6.2. The multiplication table for the semigroup (monoid) of mappings from $\{0, 1\}$

to itself was computed in the answer to Exercise 3.4. This semigroup is not commutative (e.g., $f_3 f_4 \neq f_4 f_3$), so that it cannot be isomorphic to $(\mathbf{Z}_4, +_4)$ or to (\mathbf{Z}_4, \cdot_4) by Exercise 6.8 in this exercise set.

- 6.4. For $n = 1$, we have $\varphi(a_1) = \varphi(a_1)$. Suppose the statement is true for $n = k$. Then

$$\begin{aligned}\varphi(a_1 \circ a_2 \circ \cdots \circ a_k \circ a_{k+1}) &= \varphi((a_1 \circ \cdots \circ a_k) \circ a_{k+1}) \\ &= \varphi(a_1 \circ \cdots \circ a_k) * \varphi(a_{k+1}) \\ &\quad (\text{since } \varphi \text{ is a homomorphism}) \\ &= (\varphi(a_1) * \cdots * \varphi(a_k)) * \varphi(a_{k+1}) \\ &\quad (\text{by our induction hypothesis}) \\ &= \varphi(a_1) * \cdots * \varphi(a_{k+1}),\end{aligned}$$

completing the induction.

- 6.6. One automorphism of a group is always the identity map. For $(\mathbf{Z}_6, +_6)$ another automorphism is obtained by sending g to $5g$ for all $g \in \mathbf{Z}_6$.
- 6.7. Suppose $s_1, s_2 \in S$. Then

$$\begin{aligned}(\psi \circ \varphi)(s_1 * s_2) &= \psi(\varphi(s_1 * s_2)) = \psi(\varphi(s_1) \circ \varphi(s_2)) \\ &= \psi(\varphi(s_1)) \cdot \psi(\varphi(s_2)) = (\psi \circ \varphi)(s_1) \cdot (\psi \circ \varphi)(s_2).\end{aligned}$$

- 6.9. First we must show that φ_a is a homomorphism. We have

$$\varphi_a(g_1)\varphi_a(g_2) = ag_1a^{-1}ag_2a^{-1} = a(g_1g_2)a^{-1} = \varphi_a(g_1g_2).$$

Next we must see that φ_a is 1-1. If $\varphi_a(g_1) = \varphi_a(g_2)$, then $ag_1a^{-1} = ag_2a^{-1}$, so by cancellation in G , we have $g_1 = g_2$. Finally, we must see that φ_a is onto. If $g \in G$, then $g = aa^{-1}gaa^{-1}$, so $g = \varphi_a(a^{-1}ga)$.

- 6.10. First we must see that φ is a homomorphism. We have $\varphi(ab) = (ab)^{-1} = b^{-1}a^{-1} = a^{-1}b^{-1} = \varphi(a)\varphi(b)$, where we have used the fact that G is abelian. To see that φ is 1-1, suppose $\varphi(a) = \varphi(b)$. Then we would have $a^{-1} = b^{-1}$. Taking the inverse of each side shows that $a = b$. To see that φ is onto, note that if $a \in G$, then $a = \varphi(a^{-1})$. Thus φ is an isomorphism from G to itself.
- 6.12. a) Suppose $a, b \in \varphi^{-1}(\tilde{H})$. Then $\varphi(a), \varphi(b) \in \tilde{H}$. We must show that $a \circ b^{-1} \in \varphi^{-1}(\tilde{H})$. This means that the image of $a \circ b^{-1}$ under φ must be in \tilde{H} . Since φ is a homomorphism, we have $\varphi(a \circ b^{-1}) = \varphi(a) * \varphi(b^{-1}) = \varphi(a) * (\varphi(b))^{-1}$ and this product is in \tilde{H} since \tilde{H} is a subgroup. Note that if $a \in \text{Ker}(\varphi)$, then $\varphi(a) = \bar{e} \in \tilde{H}$ since \tilde{H} is a subgroup. Thus $\text{Ker}(\varphi) \subseteq \varphi^{-1}(\tilde{H})$.
- b) Suppose $g \in G$ and $h \in \varphi^{-1}(\tilde{H})$. We must show that $g \circ h \circ g^{-1} \in \varphi^{-1}(\tilde{H})$, so we look at the image of this element under φ . We have $\varphi(g \circ h \circ g^{-1}) = \varphi(g) * \varphi(h) * (\varphi(g))^{-1}$, and this is in \tilde{H} since \tilde{H} is normal and $\varphi(h) \in \tilde{H}$.
- 6.13. Suppose $\bar{g} \in \bar{G}$ and $\varphi(h) \in \varphi(H)$. Since φ is surjective, there exists $g \in G$ such that $\varphi(g) = \bar{g}$. Then $\bar{g} * \varphi(h) * \bar{g}^{-1} = \varphi(g \circ h \circ g^{-1}) \in \varphi(H)$, since $g \circ h \circ g^{-1} \in H$ because H is normal.

III.7

7.1.

	T_0	T_1	T_{00}	T_{01}	T_{10}	T_{11}
σ_0	σ_0	σ_1	σ_0	σ_1	σ_0	σ_1
σ_1	σ_0	σ_1	σ_0	σ_1	σ_0	σ_1

The algorithm stops since every transition map corresponding to an input of length 2 equals a map corresponding to an input of length 1. The multiplication table for the semigroup of this machine is then

\diamond	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1

The machine of this semigroup looks just like the machine with which we began.

7.2.

	T_0	T_1	T_{00}	T_{01}	T_{10}	T_{11}
σ_0	σ_0	σ_1	σ_0	σ_1	σ_1	σ_2
σ_1	σ_1	σ_2	σ_1	σ_2	σ_2	σ_0
σ_2	σ_2	σ_0	σ_2	σ_0	σ_0	σ_1

The algorithm does not stop here, since T_{11} is a new map, but it is easy to see that we get no new maps for input strings of length 3. The transition map associated to an input string just depends on the number of 1's mod 3 in the input string. The multiplication for the semigroup of this machine is

\diamond	T_0	T_1	T_{11}
T_0	T_0	T_1	T_{11}
T_1	T_1	T_{11}	T_0
T_{11}	T_{11}	T_0	T_1

This semigroup is actually a cyclic group of order 3, which is isomorphic to $(\mathbb{Z}_3, +_3)$. The machine $\mathcal{A}_X(\mathcal{T}(M))$ of this group looks just like the one with which we began.

7.3.

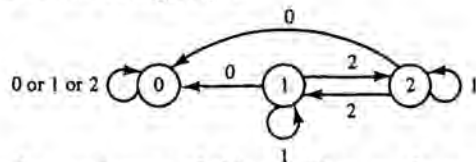
	T_0	T_1	T_{00}	T_{01}	T_{10}	T_{11}
σ_1	σ_1	σ_2	σ_1	σ_1	σ_2	σ_3
σ_2	σ_1	σ_3	σ_1	σ_1	σ_2	σ_3
σ_3	σ_1	σ_3	σ_1	σ_1	σ_2	σ_3

The algorithm does not stop here, but if one checks, one sees that no input string of length 3 gives rise to a new map. (To make this a little easier, note that, in this machine, any input string ending in 0 gives rise to the map T_0 , any input string ending in 01 gives rise to the map T_{01} , and any input string ending in 11 gives rise to the map T_{11} .) The semigroup of this machine is then

\diamond	T_0	T_1	T_{01}	T_{11}
T_0	T_0	T_{01}	T_{01}	T_{11}
T_1	T_0	T_{11}	T_{01}	T_{11}
T_{01}	T_0	T_{11}	T_{01}	T_{11}
T_{11}	T_0	T_{11}	T_{01}	T_{11}

The machine $\mathcal{A}_X(T(M))$ does not look like the original machine, since it has four states. We must introduce an equivalence relation by setting the states corresponding to T_1 and T_{01} equivalent to each other; i.e., we partition the states into the three subsets $\{T_0\}$, $\{T_1, T_{01}\}$, $\{T_{11}\}$. The resulting quotient machine looks just like the one with which we began.

7.6. The machine of this semigroup is



To compute the semigroup of this machine, we form the table:

	T_0	T_1	T_2	T_{00}	T_{01}	T_{02}	T_{10}	T_{11}	T_{12}	T_{20}	T_{21}	T_{22}
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	0	0	0	0	1	2	0	2	1
2	0	2	1	0	0	0	0	2	1	0	1	2

The algorithm stops here. The multiplication table for the semigroup of this machine is

\circ	T_0	T_1	T_2
T_0	T_0	T_0	T_0
T_1	T_0	T_1	T_2
T_2	T_0	T_2	T_1

which is a semigroup isomorphic to (\mathbb{Z}_3, \cdot_3) .

III.8

8.2. a) Let α denote rotation by 72° , let ϵ denote the identity, and let β denote a reflection. Note that $\beta\alpha = \alpha^4\beta$ by the remarks after Definition (8.2). The multiplication table for D_5 then is

	ϵ	α	α^2	α^3	α^4	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$	$\alpha^4\beta$
ϵ	ϵ	α	α^2	α^3	α^4	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$	$\alpha^4\beta$
α	α	α^2	α^3	α^4	ϵ	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$	$\alpha^4\beta$	β
α^2	α^2	α^3	α^4	ϵ	α	$\alpha^2\beta$	$\alpha^3\beta$	$\alpha^4\beta$	β	$\alpha\beta$
α^3	α^3	α^4	ϵ	α	α^2	$\alpha^3\beta$	$\alpha^4\beta$	β	$\alpha\beta$	$\alpha^2\beta$
α^4	α^4	ϵ	α	α^2	α^3	$\alpha^4\beta$	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$
β	β	$\alpha^4\beta$	$\alpha^3\beta$	$\alpha^2\beta$	$\alpha\beta$	ϵ	α^4	α^3	α^2	α
$\alpha\beta$	$\alpha\beta$	β	$\alpha^4\beta$	$\alpha^3\beta$	$\alpha^2\beta$	α	ϵ	α^4	α^3	α^2
$\alpha^2\beta$	$\alpha^2\beta$	$\alpha\beta$	β	$\alpha^4\beta$	$\alpha^3\beta$	α^2	α	ϵ	α^4	α^3
$\alpha^3\beta$	$\alpha^3\beta$	$\alpha^2\beta$	$\alpha\beta$	β	$\alpha^4\beta$	α^3	α^2	α	ϵ	α^4
$\alpha^4\beta$	$\alpha^4\beta$	$\alpha^3\beta$	$\alpha^2\beta$	$\alpha\beta$	β	α^4	α^3	α^2	α	ϵ

b) Note that since the order of D_5 is 10, the only possible orders of subgroups of D_5 are 1,2,5, and 10. Any subgroups of order 2 and 5 must be cyclic

since 2 and 5 are primes. It is then easy to see that the subgroups of D_5 are (ϵ) , (α) , (β) , $(\alpha\beta)$, $(\alpha^2\beta)$, $(\alpha^3\beta)$, $(\alpha^4\beta)$, and D_5 itself.

- c) The rotation α may be represented by the cycle $(1,2,3,4,5)$ of length 5. The reflection β may be represented by $(1)(2,5)(3,4)$. The permutation representing each of the other elements of D_5 may now be found by multiplication.

8.3. a)

$$\rho\tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}, \quad \tau\rho = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}$$

b)

$$\rho^2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}, \quad \rho^3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}, \quad \rho^4 = \epsilon$$

c) $\rho^{-1} = \rho^3, \tau^{-1} = \tau.$

- 8.4. $\sigma = (1, 2, 4)(3, 5, 6, 8)(7, 10)(9)$. The order of σ is 12, which is the least common multiple of 3, 4, and 2.

8.5. $(1, 2, 4, 5)(2, 4, 3) = (1, 2, 5)(3, 4).$

8.7. a) $\sigma\rho = (1, 5, 4)(2)(3), \rho\sigma = (1)(2, 4, 3)(5).$

b) σ has order 6 and ρ has order 4.

c) $\sigma^{-1} = (1, 4, 2)(3, 5), \rho^{-1} = (1, 2, 5, 3)(4).$

- 8.12. a) Given an integer m in $\{1, 2, \dots, n\}$, if $m = \sigma(a_i)$ for some $i = 1, 2, \dots, r$, then the image of m under the composition $\sigma \circ (a_1, a_2, \dots, a_r) \circ \sigma^{-1}$ is $\sigma(a_{i+1})$ if $i < r$ and $\sigma(a_1)$ if $i = r$. If m does not equal any of the $\sigma(a_i)$, then $\sigma^{-1}(m)$ does not equal any of the a_i , so (a_1, a_2, \dots, a_r) leaves $\sigma^{-1}(m)$ fixed. Thus the composition $\sigma \circ (a_1, a_2, \dots, a_r) \circ \sigma^{-1}$ will leave m fixed. Therefore, this composition is precisely the cycle $(\sigma(a_1), \sigma(a_2), \dots, \sigma(a_r))$.
- b) Using part (a) we see that $\sigma(1) = 1, \sigma(2) = 4, \sigma(3) = 3$, and $\sigma(4) = 2$. Thus σ is the permutation $(2, 4)$.
- c) From part (a) we see that $\sigma(1, 2, 3)\sigma^{-1}$ must be a cycle of length 3. No cycle of length 3 can equal the product of two disjoint transpositions (since, for example, a cycle of length 3 has order 3 and the product of two disjoint transpositions has order 2).

III.9

- 9.2. The group of symmetries of a regular pentagon is D_5 . This group consists of the identity, four nontrivial rotations, and five reflections. The identity, of course, fixes all $3^5 = 243$ possible 3-colored pentagons. Each nontrivial rotation will fix only those pentagons that have all vertices the same color, hence the fixed set of each nontrivial rotation contains 3 elements. Each reflection will fix $3^3 = 27$ of the 3-colored pentagons. By Burnside's Theorem, the number of distinct orbits is $(243 + 12 + 135)/10 = 39$.
- 9.4. The group of symmetries here is a cyclic group of order 4, $G = \{\epsilon, \alpha, \alpha^2, \alpha^3\}$, generated by a rotation α by 90° . We have $|X_\epsilon| = 3^4, |X_\alpha| = 3 = |X_{\alpha^3}|$, and $|X_{\alpha^2}| = 3^2$. By Burnside's Theorem, the number of different wheels is $(81 + 6 + 9)/4 = 24$.

- 9.6. The elements of D_6 are $\epsilon, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \beta, \alpha\beta, \alpha^2\beta, \alpha^3\beta, \alpha^4\beta, \alpha^5\beta$, where α is rotation by 60° and β is a reflection. We have $|X_\epsilon| = 2^6 = 64$. The rotations α and α^5 only fix the compounds with all hydrogen or all chlorine atoms. The rotations α^2 and α^4 , which have order 3, fix these two compounds and also fix the two other compounds in which hydrogens and chlorines alternate. The rotation α^3 fixes $2^3 = 8$ compounds since we are free to choose either a chlorine or a hydrogen for three consecutive carbons. There are two types of reflections. There are three reflections with axis through two vertices. These each fix $2^4 = 16$ compounds, since we are free to choose either a hydrogen or a chlorine at each of the two vertices on the axis, and we are free to choose a hydrogen or a chlorine at two of the other vertices. There are also three reflections with axis through the midpoints of opposite edges. These each fix $2^3 = 8$ compounds. The total number of different compounds is then $(64 + 4 + 8 + 8 + 48 + 24)/12 = 13$.
- 9.7. a) We have $|X_\epsilon| = 3^5 = 243$. If τ denotes the nontrivial symmetry of the tie (which may be viewed as rotation by 180°), then $|X_\tau| = 3^3 = 27$ since the first and last bands must be colored the same and the second and fourth bands must be colored the same. Thus by Burnside's Theorem, the number of different neckties is $(243 + 27)/2 = 135$.
- b) If two adjacent bands must be colored differently, then $|X_\epsilon| = 3 \cdot 2^4 = 48$ since the first band may be colored in any of three ways, but subsequent bands may be colored in either of two ways. Also, $|X_\tau| = 3 \cdot 2^2 = 12$. Thus the number of different neckties is $(48 + 12)/2 = 30$.
- 9.8. The key here is to determine correctly the symmetry group of such a box. This box is not a square and the group of symmetries is not D_4 . There are four symmetries of this box: the identity (call it ϵ), reflection about the vertical axis through the middle of the box (call it α), reflection about the horizontal axis through the middle of the box (call it β), and rotation by 180° , which equals the composition $\alpha\beta$. The group of symmetries is thus the group with four elements in which each element other than the identity has order 2 (the Klein four-group). We then have $|X_\epsilon| = 2^6 = 64$, $|X_\alpha| = 2^3 = 8$, $|X_\beta| = 2^4 = 16$ (since the single wires on either end can be either color), and $|X_{\alpha\beta}| = 2^3 = 8$. Thus there are $(64 + 8 + 16 + 8)/4 = 24$ different boxes. If we allow three possible colors, then the number of different boxes is $(3^6 + 3^3 + 3^4 + 3^3)/4 = 216$.
- 9.9. We have $|X_\epsilon| = 16$, $|X_\rho| = |X_{\rho^3}| = 2$, and $|X_{\rho^2}| = 4$. Thus the number of distinct orbits is $(16 + 2 + 2 + 4)/4 = 6$.

III.10

- 10.2. a) $(3^6 + 3 \cdot 3^4 + 12 \cdot 3^3 + 8 \cdot 3^2)/24 = 57$.
- b) The pattern inventory is $[(r+b+w)^6 + 3(r+b+w)^2(r^2+b^2+w^2)^2 + 6(r+b+w)^2(r^4+b^4+w^4) + 6(r^2+b^2+w^2)^3 + 8(r^3+b^3+w^3)^2]/24$. In $(r+b+w)^6$, there is the term $60r^3b^2w$ and in $3(r+b+w)^2(r^2+b^2+w^2)^2$, there is the term $12r^3b^2w$. In the other expressions in the pattern inventory, there are no terms in r^3b^2w . Hence the number of different cubes with 3 red faces, 2 blue faces, and 1 white face is $(60 + 12)/24 = 3$. 374
- 10.5. D_5 consists of the identity, with cycle index X_1^5 , four nontrivial rotations, each with cycle index X_5 , and five reflections, each with cycle index $X_1X_2^2$. The

cycle index of D_5 is then $(X_1^5 + 4X_5 + 5X_1X_2^2)/10$. Use this cycle index to answer the questions in this exercise.

- 10.6. Compare with Example (10.12.2).
 10.7. The number $(m^6 + 3m^4 + 12m^3 + 8m^2)/24$ is the number of different m -colorings of the faces of a cube. This number is obviously an integer.
 10.8. a) $(X_1^4 + 8X_1X_3 + 3X_2^2)/12$.
 b) $(X_1^4 + 8X_1X_3 + 3X_2^2)/12$.
 c) $(X_1^6 + 8X_3^2 + 3X_1^2X_2^2)/12$.

III.11

- 11.5. a) Write ρ and μ as products of disjoint cycles. Notice that if $\rho = \tau_1 \circ \tau_2 \circ \cdots \circ \tau_m$, then we may write $\sigma \circ \rho \circ \sigma^{-1} = (\sigma \circ \tau_1 \circ \sigma^{-1}) \circ (\sigma \circ \tau_2 \circ \sigma^{-1}) \circ \cdots \circ (\sigma \circ \tau_m \circ \sigma^{-1})$. It follows that if ρ has d_i cycles of length i , for $i = 1, 2, \dots, n$, in its representation as the product of disjoint cycles, then so will $\sigma \circ \rho \circ \sigma^{-1}$, for any $\sigma \in S_n$, by Exercise 8.12(a). Thus, if μ is conjugate to ρ , then μ must have the same cycle index as ρ . Conversely, suppose ρ and μ have the same cycle index. Suppose $\rho = \tau_1 \circ \tau_2 \circ \cdots \circ \tau_m$ is the representation of ρ as the product of disjoint cycles. Then corresponding to each τ_j , $j = 1, 2, \dots, m$, there exists a cycle ν_j of the same length in the representation of μ as a product of cycles (and this correspondence is 1-1 and onto). By Exercise 8.12(a), we may define a permutation σ such that $\sigma \circ \tau_j \circ \sigma^{-1} = \nu_j$ for $j = 1, 2, \dots, m$. (One σ works for all j because the τ_j 's are disjoint and the ν_j 's are disjoint.) It follows that $\sigma \circ \rho \circ \sigma^{-1} = \mu$.
 b) This follows easily from part (a).
 11.6. $(1, 2)(3, 4)$ and $(1, 2)(3, 5)$ is one possible answer.
 11.7. The centralizer G_x of x has index 2 by Lemma (11.5). Thus G_x is normal by Exercise 5.4.

Chapter IV

IV.1

- 1.1. a) $a(-b) + ab = a(-b + b) = a0 = 0$, so $a(-b) = -(ab)$. Also, $(-a)b + ab = (-a + a)b = 0b = 0$, so $(-a)b = -(ab)$ as well.
 b) By (a), we have $(-a)(-b) = -(a(-b)) = -(-(ab)) = ab$.
 1.6. a) If $x - y \in S$ for all $x, y \in S$, then $(S, +)$ is a subgroup of the additive group of R by Proposition (III.3.8). If $xy \in S$ for all $x, y \in S$, then (S, \cdot) is a subsemigroup of (R, \cdot) . Finally, we must check that the distributive laws hold, but since they hold for every $a, b, c \in R$, they certainly hold for every $a, b, c \in S$.
 b) We use part (a). If $x = 2n$ and $y = 2m$ are in $2\mathbb{Z}$, then $x - y = 2(n - m) \in 2\mathbb{Z}$ and $xy = 2(2nm) \in 2\mathbb{Z}$.
 1.8. It suffices to show that a field has no zero divisors other than zero. Suppose F is a field and suppose $a, b \in F$ with $ab = 0$ and $a \neq 0$. Then $a^{-1} \in F$ and

multiplying both sides by a^{-1} shows that b must be 0. This implies that there are no nonzero zero divisors in F .

- 1.9. Let U denote the set of units in R . We must show that the restriction of the multiplication on R defines an operation on U ; i.e., that U is closed under multiplication. Suppose $x, y \in U$. Then $(xy)^{-1} = y^{-1}x^{-1}$, so we have $xy \in U$. This shows that the set of units is a subsemigroup of (R, \cdot) . It is easy to see that U is then a group with 1 as its identity and with x^{-1} as the inverse of x if $x \in U$.
- 1.11. a) $2 \cdot \bar{3} = \bar{1}, 3 \cdot \bar{3} = \bar{4}, 4 \cdot \bar{3} = \bar{2}, 5 \cdot \bar{3} = \bar{0}$.
 b) $\bar{3}^2 = \bar{4}, \bar{3}^3 = \bar{2}, \bar{3}^4 = \bar{1}, \bar{3}^5 = \bar{3}$.
 c) It suffices to show that each nonzero element has a multiplicative inverse. We have $\bar{1}^{-1} = \bar{1}, \bar{2}^{-1} = \bar{3}, \bar{3}^{-1} = \bar{2}, \bar{4}^{-1} = \bar{4}$.
- 1.12. a) We have $r + r = (r + r)^2 = r^2 + 2r^2 + r^2 = r + 2r + r$. Therefore, $2r = 0$, which means that $r + r = 0$. Thus, $r = -r$.
 b) Suppose $r, s \in R$. Then $(r + s)^2 = r + s$, while on the other hand, $(r + s)^2 = r^2 + rs + sr + s^2 = r + rs + sr + s$. Hence $rs + sr = 0$, which implies that $rs = -sr = sr$ (using (a)).
 c) Recall that "multiplication" in $\mathcal{P}(S)$ means intersection of subsets. If A is any subset of S , then $A \cap A = A$.
- 1.13. a) We may assume $r \neq 0$. Let n be the least positive integer (necessarily at least 2) such that $r^n = 0$. Then $r \cdot r^{n-1} = 0$ and r^{n-1} is not 0. Therefore r is a zero divisor.
 b) The nonzero zero divisors in \mathbf{Z}_8 are 2, 4, and 6 so we must check these to see if they are nilpotents. Indeed, they all are since in \mathbf{Z}_8 we have $2^3 = 0$, $4^2 = 0$, and $6^3 = 0$.
- 1.16. The zero divisors are 0, 2, 3, 4, 6, 8, 9 and 10. The units are 1, 5, 7, and 11.

IV.2

- 2.5. b) We cannot apply Corollary (2.9) here since $(5, 1000) = 5$ and not 1. If you start computing powers of 125 mod 1000, you will find that odd powers of 125 are congruent to 125 mod 1000 and even powers of 125 are congruent to 625 mod 1000. Therefore, the last three digits of $(125)^{134}$ are 625.
- 2.7. a) $10 \equiv 1 \pmod{9}$, so any power of 10 is congruent to 1 mod 9.
 b) If $m = a_n a_{n-1} \cdots a_0$ is the decimal representation of m , then $m = 10^n a_n + 10^{n-1} a_{n-1} + \cdots + a_0$. By part (a), m is congruent to 0 mod 9 if and only if the sum of the digits of m is congruent to 0 mod 9.
- 2.8. We have $2^6 = 64 \equiv 1 \pmod{63}$. Hence $2^{63} = 2^{60} 2^3 = (2^6)^{10} 2^3 \equiv 2^3 \pmod{63}$. Thus 2^{63} is congruent to 8, not to 2, mod 63.
- 2.9. a) $x = \bar{6}$. b) $x = \bar{6}$. c) $x = \bar{3}$ or $\bar{4}$.
- 2.11. There must exist positive integers i and j , with $i < j$, such that $i \cdot 1 = j \cdot 1$. Then we have $(j - i) \cdot 1 = 0$.
- 2.12. $nr = n(1 \cdot r) = (n \cdot 1)r = 0r = 0$.
- 2.13. Suppose R is an integral domain of characteristic $k > 0$. If k is ~~396~~ a prime, then we may write $k = mn$ with $1 < m, n < k$. Then we have $0 = k1 = (mn)1 = (m1)(n1)$. Since R is an integral domain, we must have $m1 = 0$ or

$n1 = 0$. But this contradicts the fact that k is the least positive integer such that $k1 = 0$.

- 2.15. Using the Euclidean Algorithm, one shows that $1 = 16(35) - 13(43)$. Hence, in \mathbf{Z}_{43} , we have $35^{-1} = 16$.
- 2.16. Let the elements of the field be $0, 1, a, b$. The additive group must be a group of order 4 in which each nonzero element has order 2 (the Klein four-group). The nonzero elements $1, a, b$ form a group under multiplication, which must then be a cyclic group of order 3. Therefore, we must have $a^2 = b, b^2 = a, ab = ba = 1$.
- 2.17. Since the coefficient of X^3 must be 1 and the coefficients of X^2, X and the constant term can be either 0 or 1, there are $2^3 = 8$ cubic polynomials over \mathbf{Z}_2 . We just need to check each one to see if it has 0 or 1 as a root. (See Proposition (6.19).) The irreducible ones are $X^3 + X + 1$ and $X^3 + X^2 + 1$.

IV.3

- 3.2. a) If $1 \in I$, then $r = r \cdot 1 \in I$ for every $r \in R$.
 b) Suppose u is a unit and $u \in I$. Then $1 = u^{-1}u \in I$, so $I = R$ by (a).
 c) Suppose F is a field and I is an ideal of F . If $I \neq \{0\}$, then there exists a nonzero element in I . But in a field every nonzero element is a unit. Hence we would have $I = F$ by (b).
- 3.3. a) Suppose I and J are ideals of a ring R . We know that the intersection of two subgroups is a subgroup, so $I \cap J$ is a subgroup of $(R, +)$. Now suppose $r \in R$ and $x \in I \cap J$. Then $rx \in I$ and $rx \in J$, hence $rx \in I \cap J$. Similarly, $rx \in I \cap J$.
 b) In \mathbf{Z} , consider the union of (2) and (3). This is not an ideal because it is not even a subgroup of \mathbf{Z} since 3 and 2 are in the union, but $1 = 3 - 2$ is not in the union.
- 3.12. a) Since the additive group \mathbf{Z}_n is cyclic, any subgroup must be cyclic by Proposition (III.3.19). It is easy to see the cyclic subgroup generated by \bar{a} is the same as the principal ideal (\bar{a}) .
 b) Suppose I is a nonzero ideal of \mathbf{Z}_n . By part (a), any ideal I in \mathbf{Z}_n is principal, so of the form $I = (\bar{a})$. If $(a, n) = 1$, then \bar{a} is a unit in \mathbf{Z}_n , so $(\bar{a}) = \mathbf{Z}_n$ by Exercise 3.2. If $(a, n) \neq 1$, then let b denote the greatest common divisor of a and n . We claim that $(\bar{a}) = (\bar{b})$ in \mathbf{Z}_n . Indeed, $a = bk$ for some k , so $\bar{a} = \bar{b}\bar{k}$. This means that \bar{a} is in the ideal (\bar{b}) , which implies that $(\bar{a}) \subseteq (\bar{b})$. On the other hand, there exist integers s and t such that $b = sa + tn$. Hence, we have $\bar{b} = \bar{s}\bar{a}$, which shows that $\bar{b} \in (\bar{a})$. It follows that $(\bar{b}) \subseteq (\bar{a})$ and hence that $(\bar{a}) = (\bar{b})$. Thus the ideal I equals (\bar{b}) , where b is a positive divisor of n .
 c) The distinct ideals of \mathbf{Z}_{24} are $(0), (\bar{2}), (\bar{3}), (\bar{4}), (\bar{6}), (\bar{8}), (\bar{12})$, and \mathbf{Z}_{24} itself.
- 3.14. a) Take $[0]_4$ and $[2]_4$ to $[0]_2$ and take $[1]_4$ and $[3]_4$ to $[1]_2$. It is not hard to check that this is a ring homomorphism.
 b) If φ were such a homomorphism, then we would have to have $\varphi([1]_2 + [1]_2) = [1]_4 + [1]_4 = [2]_4$. But $[1]_2 + [1]_2 = [0]_2$ and $\varphi([0]_2)$ must equal $[0]_4 \neq [2]_4$.

- 3.15. φ is a group homomorphism since $\varphi(a+b) = 3(a+b) = 3a+3b = \varphi(a) + \varphi(b)$. However, φ is not a ring homomorphism since $3 = \varphi(1) = \varphi(1 \cdot 1)$ while $\varphi(1)\varphi(1) = 3 \cdot 3 = 9$.

IV.4

- 4.1. The positive integers less than or equal to p^r that are not relatively prime to p^r are $p, 2p, 3p, \dots, p^{r-1}p$.
- 4.2. (b_1, b_2, \dots, b_k) is a unit if and only if there exists (c_1, c_2, \dots, c_k) such that

$$(b_1c_1, b_2c_2, \dots, b_kc_k) = (1, 1, \dots, 1).$$

- 4.3. If r is a unit, then there exists r' such that $rr' = 1 = r'r$. By Exercise 3.6, we have $\psi(1_R) = 1_S$. Hence $1_S = \psi(1_R) = \psi(rr') = \psi(r)\psi(r')$, and similarly we have $1_S = \psi(r')\psi(r)$. Conversely, if there exists $s \in S$ such that $\psi(r)s = 1_S = s\psi(r)$, then there exists $r' \in R$ such that $\psi(r') = s$. We then have $\psi(rr') = \psi(r)s = 1_S = \psi(r'r)$. Since ψ is 1-1 and we know that $\psi(1_R) = 1_S$, we may conclude that $rr' = r'r = 1_R$.
- 4.4. In the notation of Theorem (4.3), we have $m_1 = 2, m_2 = 3$, and $m_3 = 5$. We then take $c_1 = 15, c_2 = 10$, and $c_3 = 6$. The smallest positive solution is then the remainder when $(15)(1) + (10)(1) + (6)(3)$ is divided by $m = 30$. Thus the smallest positive solution is 13 and the next smallest positive solution is $13 + 30 = 43$.
- 4.5. Here $m_1 = 3, m_2 = 5$, and $m_3 = 7$. We have $c_1 = 70, c_2 = 21$, and $c_3 = 15$. The smallest positive solution is the remainder when $(70)(2) + (21)(3) + (15)(4)$ is divided by 105. Thus the smallest positive solution is 53 and the next smallest solution is 158.
- 4.7. Notice that 3 and 6 are not relatively prime. The integers that have remainders 2 and 3 when divided by 3 and 4, respectively, are congruent to 11 mod 12. But if an integer is congruent to 11 mod 12, then it is congruent to 5 mod 6, so there is no solution to the given problem.
- 4.8. 11.
- 4.10. In the notation of Example (4.5), we have

$$\begin{aligned}\bar{\psi}(18) &= ([2]_4, [0]_3, [3]_5, [4]_7, [7]_{11}) \\ \bar{\psi}(48) &= ([0]_4, [0]_3, [3]_5, [6]_7, [4]_{11}) \\ \bar{\psi}(54) &= ([2]_4, [0]_3, [4]_5, [5]_7, [10]_{11}) \\ \bar{\psi}(22) &= ([2]_4, [1]_3, [2]_5, [1]_7, [0]_{11}) \\ \bar{\psi}(32) &= ([0]_4, [2]_3, [2]_5, [4]_7, [10]_{11}) \\ \bar{\psi}(60) &= ([0]_4, [0]_3, [0]_5, [4]_7, [5]_{11}).\end{aligned}$$

If we put $N = 18 \cdot 48 + 54 \cdot 22 + 32 \cdot 60$, then $\bar{\psi}(N) = ([0]_4, [0]_3, [2]_5, [3]_7, [1]_{11})$. Using the values for c_1, \dots, c_5 in Example (4.5), we see that N is the remainder when $(3696)(2) + (2640)(3) + 2520$ is divided by 4620. Thus $N = 3972$.

IV.5

- 5.1. a) Use Lemma (2.8). b) Mimic the proof of Theorem (5.4).
- 5.2. a) 17. b) 269.
- c) First, to compute $3^{5 \cdot 17} = 3^{85} \bmod 377$, note that mod 377 we have $3^2 = 9$, $3^4 = 81$, $3^8 \equiv 152$, $3^{16} \equiv 107$, $3^{32} \equiv 139$, $3^{64} \equiv 94$. Then $3^{85} = 3^{64} 3^{16} 3^4 3^1 \equiv 3 \bmod 377$. Next, we need to compute $3^{5 \cdot 269} = 3^{1345}$. This can be simplified by using Corollary (2.9), which says here that $3^{336} \equiv 1 \bmod 377$. Since $1345 = 4(336) + 1$, we have $3^{1345} = (3^{336})^4 \cdot 3 \equiv 3 \bmod 377$.
- 5.3. a) Note that $\varphi(n) = n - (p + q) + 1$. Since we know n and $\varphi(n)$, we know $p + q$. Then from the equation $(p - q)^2 = (p + q)^2 - 4n$, we can find $p - q$. Since we know $p + q$ and $p - q$, it is easy to solve for p and q .
- b) If n were a prime, then $\varphi(n)$ would be $n - 1$. Then given an encryption exponent e , anyone could solve for the corresponding decryption exponent.
- 5.4. a) $d = 3$. b) 05. c) 0715 (= "GO").
- 5.5. HI.
- 5.6. The integers 4, 5, 6, 9, 10, 11, and 14 all have this property.
- 5.9. a) $d_1 = 13$.
- b) You compute $26^{13} \bmod 77$, which is 75, and then send $75^5 \bmod 91$, which is 17.
- c) Note that User 2's decryption exponent is 5. User 2 computes $17^5 \bmod 91$, which is 75, and then User 2 computes $75^7 \bmod 77$, which is 26 (= "Z").

IV.6

- 6.3. a) $q(X) = \bar{2}X + \bar{2}$, $r(X) = X + \bar{2}$. b) $q(X) = \bar{2}X + 1$, $r(X) = \bar{2}X$.
- 6.4. $X^2 + \bar{2}$, $X^2 + \bar{3}$, $X^2 + X + 1$, $X^2 + X + \bar{2}$, $X^2 + \bar{2}X + \bar{3}$, $X^2 + \bar{2}X + \bar{4}$, $X^2 + \bar{3}X + \bar{3}$, $X^2 + \bar{3}X + \bar{4}$, $X^2 + \bar{4}X + 1$, $X^2 + \bar{4}X + \bar{2}$.
- 6.5. There are p monic linear polynomials. Hence there are p monic quadratics that are squares of linear polynomials, and there are $p(p - 1)/2$ monic quadratics that are the product of two distinct linear polynomials. Thus the number of irreducible monic quadratic polynomials is $p^2 - p - p(p - 1)/2 = p(p - 1)/2$.
- 6.7. 1, $\bar{5}$, $\bar{7}$ (note that \mathbf{Z}_{12} is not a field, so Corollary (6.14) does not apply).
- 6.10. The GCD is $X + \bar{2}$ (remember that the GCD must be monic), and we have $X + \bar{2} = \bar{2}(X + 1)(X^4 + \bar{2}X^2 + \bar{2}X + 1) + (X^2 + 1)(X^3 + X^2 + X)$.
- 6.13. If $p(X)$ does not divide $f(X)$, then, since $p(X)$ is irreducible, we may infer that the only common divisors of $p(X)$ and $f(X)$ are constants.

IV.7

- 7.1. a) 27. b) 25. c) 125.
- 7.2. The elements in K are 0, 1, $\bar{2}$, α , $\alpha + 1$, $\alpha + \bar{2}$, $\bar{2}\alpha$, $\bar{2}\alpha + 1$, and $\bar{2}\alpha + \bar{2}$. Note that $\alpha^2 + 1 = 0$, so $\alpha^2 = \bar{2}$.
- a) $\bar{2}\alpha$. b) $\alpha + \bar{2}$. c) $\bar{2}\alpha + \bar{2}$.
- 7.3. $\bar{2}\alpha^3 + \alpha + 1$.
- 7.5. See that $\alpha + 1$ is a generator. (Remember that $\alpha^2 = 2$.)

- 7.9. a) Use the Division Algorithm for polynomials in $F[X]$ and the minimality of $\deg(f)$.
 b) Use (a) and Exercise 6.1.
 c) If $f(X) = g(X)h(X)$ and if $f(\alpha) = 0$, then either $g(\alpha) = 0$ or $h(\alpha) = 0$.
 d) Show that the kernel of φ is the ideal $(p(X))$ and use Corollary (3.12).

IV.8

- 8.1. 2, 3, 4, 5, 7, 8, 9, 11, 13, 16, 17, 19.
 8.2. Since $64 = 2^6$, the subfields, by Theorem (8.10), are $GF(2)$, $GF(4)$, $GF(8)$, and $GF(64)$ itself.
 8.4. The group of nonzero elements in $GF(8)$ has order 7, hence every nonzero element except the element 1 must generate it by Corollary (III.4.11).
 8.6. Let α denote the image of X in $\mathbb{Z}_3[X]/(X^2 + X + \bar{2})$. Note that $\alpha^2 + \alpha + \bar{2} = 0$, hence $\alpha^2 = -\alpha - \bar{2} = \bar{2}\alpha + 1$. Some computation then shows that α is a primitive element. The other generators of the cyclic group of order 8 of nonzero elements are $\alpha^3 = \bar{2}\alpha + \bar{2}$, $\alpha^5 = \bar{2}\alpha$, and $\alpha^7 = \alpha + 1$.

IV.9

- 9.4. Here is one possible answer. We compute the matrices A_1 and A_2 in the proof of Theorem (9.9). Writing 7 in place of 0 and omitting bars over elements of $GF(7)$, we have

$$A_1 = \begin{bmatrix} 2 & 3 & 4 & 5 & 6 & 7 & 1 \\ 3 & 4 & 5 & 6 & 7 & 1 & 2 \\ 4 & 5 & 6 & 7 & 1 & 2 & 3 \\ 5 & 6 & 7 & 1 & 2 & 3 & 4 \\ 6 & 7 & 1 & 2 & 3 & 4 & 5 \\ 7 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 3 & 4 & 5 & 6 & 7 & 1 & 2 \\ 5 & 6 & 7 & 1 & 2 & 3 & 4 \\ 7 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 7 & 1 \\ 4 & 5 & 6 & 7 & 1 & 2 & 3 \\ 6 & 7 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}.$$

- 9.6. bk is the sum of the cardinalities of the blocks. Since each of the elements in V occurs in exactly r blocks, this sum must equal vr .
 9.10. We may take $\bar{2}$ as a primitive element in $GF(13)$. The blocks are then $B_{0j} = \{1 + \alpha_j, \bar{3} + \alpha_j, \bar{9} + \alpha_j\}$ and $B_{1j} = \{\bar{2} + \alpha_j, \bar{6} + \alpha_j, \bar{5} + \alpha_j\}$ where $j = 1, 2, \dots, 13$ and $\alpha_1, \alpha_2, \dots, \alpha_{13}$ are the elements $1, \bar{2}, \bar{3}, \dots, \bar{12}, 0$ of $GF(13)$.

Chapter V

V.1

- 1.2. A four-times repetition code cannot correct two errors, but a five-times repetition code can (since three occurrences of the correct message will be received accurately).
- 1.3. a) 10. b) 01. c) 11. d) 01.
- 1.4. No. For example, if 10000 is received, the receiver does not know if the codeword transmitted was 00000 or 10001.
- 1.6. This code is 1-error-correcting since each codeword differs from each other codeword in at least four places. It cannot correct two errors. (Suppose 011000 is received.)

V.2

- 2.1. Each codeword \hat{c} may be written in a unique way in the form $\hat{c} = a_1\hat{c}_1 + a_2\hat{c}_2 + \cdots + a_k\hat{c}_k$, where $\hat{c}_1, \dots, \hat{c}_k$ is a basis for the code. Since each $a_i \in GF(q)$, there are q possibilities for each coefficient.
- 2.2. a) The eight codewords are 00000, 10010, 01011, 00101, 11001, 10111, 01110, and 11100.
- b) A parity check matrix is $H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$.
- 2.3. a) The nine codewords are 0000, 1022, 0112, 2011, 0221, 1101, 2120, 1210, and 2202.
- b) A parity check matrix for C is $H = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$.
- 2.5. a) $\begin{bmatrix} 1 & 0 & 2 & 2 \\ 0 & 1 & 2 & 0 \end{bmatrix}$.
- b) $H = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$.
- 2.8. a)

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

- b) Yes, since $H[111100]^t = [000]^t$.
- 2.10. a) Suppose $\sigma, \tau \in \mathcal{G}(C)$. By Proposition (III.3.11), it suffices to show that $\sigma\tau \in \mathcal{G}(C)$. If $\hat{c} \in C$, then we have $\sigma\tau(\hat{c}) = \sigma(\tau(\hat{c})) = \sigma(\hat{c}')$ for some codeword \hat{c}' (since $\tau \in \mathcal{G}(C)$). Since $\sigma \in \mathcal{G}(C)$, we know that $\sigma(\hat{c}') = \hat{c}''$ for some $\hat{c}'' \in C$. This shows that $\sigma\tau \in \mathcal{G}(C)$.
- b) The codewords of C are 000 and 111. Every permutation in S_3 leaves these two strings fixed. Therefore, $\mathcal{G}(C) = S_3$.
- c) The codewords of C are 000, 101, 011, 110. Then $\mathcal{G}(C) = S_3$.
- d) The codewords of C are 000, 100, 101, 001. Then $\mathcal{G}(C) = \{\epsilon, (13)\}$.

V.3

- 3.2. $d(C) = 2$ (101000 is a codeword).
- 3.4. Let S denote the set of codewords of even weight and let T denote the set of codewords of odd weight. If T is empty, then there is nothing to show, so suppose \hat{c}_1 is a codeword of odd weight. Define a mapping $f: S \rightarrow T$ by $f(\hat{c}) = \hat{c} + \hat{c}_1$. Then f is 1-1 by the cancellation law in C and f is onto since if \hat{c}' is any odd codeword, then $f(\hat{c}_1 + \hat{c}') = \hat{c}'$. Therefore, the sets S and T have the same number of elements.
- 3.7. a) Take, for example, $H = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$ as a parity check matrix.
 b) No. The parity check matrix would have to be a 2×4 matrix with distinct nonzero columns; but there are only three distinct nonzero vectors in $GF(2)^2$.
- 3.8. a) 1100110. b) 1100110. c) 0001111.
- 3.9. The columns of H are precisely the seven nonzero vectors in $GF(2)^3$.
- 3.10. a) 02211. b) 11001 c) 01122.

V.4

- 4.2. a) Yes. b) No; it will be decoded as 01101.
- 4.3. a) Yes. b) No; it will be decoded as 1012. c) No; it will be decoded as 0211.
- 4.7. One possible syndrome table is

Coset Leader	Syndrome
00000	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
00001	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
00002	$\begin{bmatrix} 0 \\ 2 \end{bmatrix}$
00010	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
00020	$\begin{bmatrix} 2 \\ 0 \end{bmatrix}$
00100	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
00200	$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$
10000	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$
20000	$\begin{bmatrix} 2 \\ 2 \end{bmatrix}$

We would then decode 11202 as 11200, 20112 as 20012, and 20201 as 20101.

V.5

- 5.3. a) 2021. b) 1201. c) 2210.
- 5.5. $H_{2,5} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 \end{bmatrix}$.
- 5.6. a) 111000000000000. b) 000111000011000.
- 5.8. a) For $t = 1$, we have $\binom{8}{0} + \binom{8}{1} = 9$. For $t = 2$, we have $\binom{8}{0} + \binom{8}{1} + \binom{8}{2} = 36$. For $t = 3$, we have $\binom{8}{0} + \binom{8}{1} + \binom{8}{2} + \binom{8}{3} = 92$. Since 9, 36, and 92 are not powers of 2, there cannot be a binary perfect code of length 8.
- b) Yes. It is Ham(2,7), which is an (8,6,3) code over $GF(7)$.

V.6

- 6.2. a) A cyclic shift of a string does not change its weight.
 b) $H = [1 \ 1 \ \dots \ 1]$.
 c) $g(X) = X - 1, h(X) = X^{n-1} + X^{n-2} + \dots + 1$.
- 6.3. $G = \begin{bmatrix} 1 & 0 & \dots & 1 & 0 \\ 0 & 1 & \dots & 0 & 1 \end{bmatrix}$.
- 6.6. Use the factorization $X^4 - 1 = (X - 1)(X + 1)(X^2 + 1)$.
- 6.7. Use the factorization $X^7 - 1 = (X - 1)(X^3 + X^2 + 1)(X^3 + X + 1)$.
- 6.9. 2^m .
- 6.10. It suffices to show that $X^{11} + X^{10} + X^6 + X^5 + X^4 + X^2 + 1$ divides $X^{23} - 1$.
- 6.13. Take

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \end{bmatrix}.$$

Then the resulting code has minimum distance 5 by Theorem (3.10) and Exercise 6.12.

Chapter VI

VI.1

- 1.4. We will show that $a \vee (b \vee c) = (a \vee b) \vee c$ by applying Lemma (1.4'). To do this, we will see that $a \wedge (a \vee (b \vee c)) = a \wedge ((a \vee b) \vee c)$ and $\bar{a} \wedge (a \vee (b \vee c)) = \bar{a} \wedge ((a \vee b) \vee c)$. Exercise 1.3 shows that $a \wedge (a \vee (b \vee c)) = a$. We also have

$$\begin{aligned} a \wedge ((a \vee b) \vee c) &= (a \wedge (a \vee b)) \vee (a \wedge c) \text{ by distributivity} \\ &= a \vee (a \wedge c) \text{ by Exercise 1.3} \\ &= a \text{ by Exercise 1.3 again.} \end{aligned}$$

Now, $\bar{a} \wedge (a \vee (b \vee c)) = (\bar{a} \wedge a) \vee (\bar{a} \wedge (b \vee c)) = 0 \vee (\bar{a} \wedge (b \vee c)) = \bar{a} \wedge (b \vee c)$.
 And

$$\begin{aligned} \bar{a} \wedge ((a \vee b) \vee c) &= (\bar{a} \wedge (a \vee b)) \vee (\bar{a} \wedge c) \\ &= ((\bar{a} \wedge a) \vee (\bar{a} \wedge b)) \vee (\bar{a} \wedge c) \\ &= (0 \vee (\bar{a} \wedge b)) \vee (\bar{a} \wedge c) = (\bar{a} \wedge b) \vee (\bar{a} \wedge c) \\ &= \bar{a} \wedge (b \vee c). \end{aligned}$$

Lemma (1.4') may now be applied to show that \vee is associative. The associativity of \wedge follows by duality.

- 1.6. $\bar{a} = \bar{a} \vee (a \wedge b) = (\bar{a} \vee a) \wedge (\bar{a} \vee b) = \bar{a} \vee b = (\bar{a} \vee b) \wedge (a \vee b) = ((\bar{a} \vee b) \wedge a) \vee ((\bar{a} \vee b) \wedge b) = (b \wedge a) \vee b = b$.

1.8. We have

$$\begin{aligned}(a \wedge b) \vee (\bar{a} \vee \bar{b}) &= ((a \wedge b) \vee \bar{a}) \vee \bar{b} \\ &= ((a \vee \bar{a}) \wedge (b \vee \bar{a})) \vee \bar{b} = (b \vee \bar{a}) \vee \bar{b} \\ &= (b \vee \bar{b}) \vee \bar{a} = 1 \vee \bar{a} = 1.\end{aligned}$$

We also have

$$\begin{aligned}(a \wedge b) \wedge (\bar{a} \vee \bar{b}) &= ((a \wedge b) \wedge \bar{a}) \vee ((a \wedge b) \wedge \bar{b}) \\ &= ((a \wedge \bar{a}) \wedge b) \vee (a \wedge (b \wedge \bar{b})) \\ &= 0 \vee 0 = 0.\end{aligned}$$

It follows from Exercise 1.6 that $\bar{a} \vee \bar{b}$ is the complement of $a \wedge b$. The other DeMorgan Law follows by duality.

1.10. This follows from Lemma (III.2.4).

1.14. Note that $x_1 \wedge x_2 \wedge x_3$ is closed only when x_1, x_2 , and x_3 are all closed, and $x_1 \wedge \bar{x}_2 \wedge x_3$ is closed only when x_1 and x_3 are closed and x_2 is open. Therefore, E is closed precisely when x_1 and x_3 are closed (and it doesn't matter if x_2 is open or closed). Thus, E has the same closure table as $x_1 \wedge x_3$.

VI.2

2.1. Use Exercise 1.3.

2.2. Use Exercises 1.8 and 2.1.

2.7. There are nine positive divisors of 36 and nine is not a power of 2.

2.9. Theorem (2.13) shows that a Boolean algebra with six atoms must have 2^6 elements.

VI.3

3.2. c) $x_2 \wedge x_3$.

3.4. See Lemma (4.3) in the next section.

3.6. a) If $E = (x_1 \vee x_2) \wedge x_3$, then f_E takes the value 1 on the triples $(0, 1, 1)$, $(1, 0, 1)$, and $(1, 1, 1)$. Therefore,

$$f_E = m_3 \vee m_5 \vee m_7 = \sum(3, 5, 7).$$

b) $\sum(0, 1, 4, 5, 6, 7)$.

c) $\sum(0, 1, 2, 3, 4, 5, 6, 7)$ (i.e., $f_E = 1$).

d) $\sum(0, 1, 2, 3, 5, 6, 7)$.

3.7. Suppose $f \leq g$ and suppose $f(b_1, b_2, \dots, b_n) = 1$. Then

$$1 \wedge g(b_1, b_2, \dots, b_n) = 1$$

(from the definition of \leq), so we must have $g(b_1, b_2, \dots, b_n) = 1$. Conversely, to see that $f \wedge g = f$, we need just worry about the times when $f = 1$ (for if $f(b_1, b_2, \dots, b_n) = 0$, then we have $0 \wedge g = 0$ regardless of whether $g(b_1, b_2, \dots, b_n) = 0$ or 1). If g is 1 whenever f is 1, then $f \wedge g$ will be 1 ($= f$) whenever f is 1.

VI.4

- 4.1. a) D.n.f. = $\bar{x}_1 x_2 \vee x_1 x_2$. Prime implicant = x_2 .
 c) D.n.f. = $\bar{x}_1 x_2 \vee x_1 \bar{x}_2$ and these two products are the prime implicants.
- 4.2. a) Yes, since $(\bar{x}_1 \vee x_2)(x_1 \vee x_2) \equiv (\bar{x}_1 \vee x_2)(\bar{x}_1 \bar{x}_2) \equiv \bar{x}_1 \bar{x}_2 \vee x_2(\bar{x}_1 \bar{x}_2) \equiv \bar{x}_1 \bar{x}_2 \vee 0 \equiv \bar{x}_1 \bar{x}_2$.
 b) No, since $(\bar{x}_1 \vee x_2)\bar{x}_2 \equiv (x_1 \bar{x}_2)\bar{x}_2 \equiv x_1 \bar{x}_2$ (so $x_1 \bar{x}_2$ is the d.n.f. of the second expression in (b)).
- 4.3. a) D.n.f. = $x_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 x_2 x_3 \vee \bar{x}_1 \bar{x}_2 x_3$; prime implicants are $\bar{x}_2 x_3, x_1 \bar{x}_2, x_1 x_3$.
 c) D.n.f. = $E_{(5)} \vee E_{(6)} \vee E_{(7)}$; prime implicants are $x_1 x_3, x_1 x_2$.
 e) D.n.f. = $E_{(0)} \vee E_{(2)} \vee E_{(3)} \vee E_{(4)} \vee E_{(5)} \vee E_{(7)}$; prime implicants are $\bar{x}_1 \bar{x}_3, \bar{x}_2 \bar{x}_3, \bar{x}_1 x_2, x_2 x_3, x_1 \bar{x}_2, x_1 x_3$.
- 4.4. c) Applying Algorithm (4.13), we obtain the following table.

0011 ✓	0-11	- -11
0111 ✓	-011 ✓	
1000	-111 ✓	
1011 ✓	111-	
1110 ✓		
1111 ✓		

Thus the prime implicants are $x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4, \bar{x}_1 x_3 \bar{x}_4, x_1 x_2 x_3, x_3 x_4$.

- 4.6. a) The d.n.f. of \bar{x}_2 is $x_1 \bar{x}_2 \vee \bar{x}_1 \bar{x}_2$. We then have

$$x_2 \equiv \bar{x}_2 \equiv (\overline{x_1 \bar{x}_2 \vee \bar{x}_1 \bar{x}_2}) \equiv (\bar{x}_1 \vee x_2)(x_1 \vee x_2).$$

This is the conjunctive normal form of x_2 .

VI.5

- 5.2. a) The original expression is the minimal sum of products equivalent to itself.
 c) $x_1 x_3 \vee x_1 x_2$.
 e) This is similar to Example (5.5.2). The original expression is one of the minimal sums of products equivalent to itself. It is also equivalent to $\bar{x}_2 \bar{x}_3 \vee \bar{x}_1 x_2 \vee x_1 x_3$.
- 5.3. c) $(x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4) \vee (x_1 x_2 x_3) \vee (x_3 x_4)$.
- 5.4. a) To find the minimal product of sums expression equivalent to E , find the minimal sum of products expression equivalent \bar{E} and then take the complement.
- 5.5. a) No change from 5.2(a).
 c) $\bar{x}_1 x_2 \bar{x}_3$ is now a prime implicant of $E \vee x_1 \bar{x}_3 \vee x_2 \bar{x}_3$, but the minimal sum of products is still $x_1 x_3 \vee x_1 x_2$.
 e) The prime implicants of $E \vee x_1 \bar{x}_3 \vee x_2 \bar{x}_3$ are $\bar{x}_2 \bar{x}_3, x_2$, and x_1 . Subject to the don't care conditions, the expression is equivalent to $x_1 \vee x_2 \vee \bar{x}_2 \bar{x}_3$.

Chapter VII

VII.1

- 1.1. a) $n - 1$.
 b) Use Proposition (1.9) and (a).
 c) Since G is simple, the number of edges in G cannot exceed the number of edges in K_p .
 d) For K_4 , see Figure 2.3. K_5 cannot be drawn in the plane so that its edges only intersect at their ends.
- 1.3. Use Corollary (1.10).
- 1.5. If G has a loop, then that loop is a cycle, so assume G has no loops. Start at a vertex v_1 and take an edge e_1 such that $\psi_G(e_1) = \{v_1, v_2\}$ with $v_2 \neq v_1$. If there is another edge joining v_1 and v_2 , then you are done. If not, then as the next edge in your cycle take an edge e_2 with $\psi_G(e_2) = \{v_2, v_3\}$ and $v_3 \neq v_1, v_2$. Continue this process. Since G is finite, this process must stop. Use the fact that $d(v) \geq 2$ for all v to show that this process must stop at a vertex that has already appeared in the cycle.
- 1.6. Let $P_1 : u = u_0, e_1, u_1, \dots, e_n, u_n = v$ and $P_2 : u = v_0, f_1, v_1, \dots, v_n = v$ be two distinct paths. Then there exists k such that $u_i = v_i$ for $i \leq k$, but $u_{k+1} \neq v_{k+1}$. Show that there is a cycle that begins and ends with u_k .
- 1.7. If v_i occurs twice in a u - v walk, then delete the vertices and edges between the two occurrences, including one occurrence of v_i .
- 1.9. Suppose not and let G be a simple graph with the minimum number of vertices such that $q > \binom{p-1}{2}$ and G is disconnected. Let v be a vertex of G . Argue that there must be an edge e which joins v to another vertex. Now delete v and e ; the resulting graph must be connected by the choice of G .
- 1.12. Suppose not and let G be a graph with the minimal number of vertices p such that G is connected and G has fewer than $p - 1$ edges. Show that G must have a vertex of degree 1.

VII.2

- 2.1. Each arc has precisely one tail and one head.
- 2.2. A digraph with only one vertex and one loop has an odd number of vertices of odd outdegree. There must be an even number of vertices of odd degree by Corollary (1.10).
- 2.6. a) If there exists a u - v directed path, then there clearly exists a u - v path in the underlying graph.
 b) Figure 2.1 gives a counterexample to this since u is not reachable from v (or from w).
- 2.7. See the hint for Exercise 1.5.

2.8. a)

Vertex	Distance from u_0	Shortest Path from u_0
v_1	3	u_0, v_3, v_1
v_2	5	u_0, v_3, v_4, v_2
v_3	2	u_0, v_3
v_4	3	u_0, v_3, v_4

b)

Vertex	Distance from u_0	Shortest Path from u_0
v_1	2	u_0, v_4, v_2
v_2	7	u_0, v_4, v_1, v_2
v_3	13	u_0, v_4, v_1, v_2, v_3
v_4	1	u_0, v_4
v_5	∞	unreachable
v_6	11	u_0, v_4, v_1, v_2, v_6

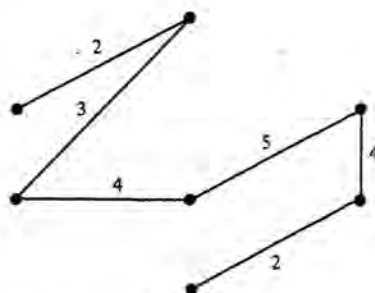
VII.3

3.1. This follows from Corollary (1.10) and Lemma (3.3).

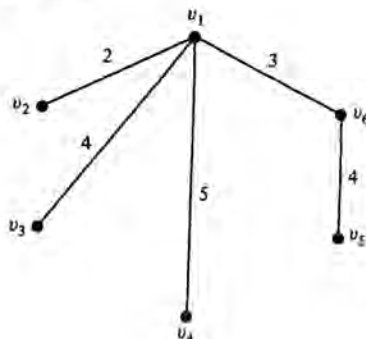
3.3. a) If e is a bridge, then there exist vertices u and v of G such that u and v are connected in G , but not connected in $G - e$.

3.4. b) Note that the vertices at the maximum distance from a fixed vertex must be leaves of the tree. Use induction on the number of vertices and delete all leaves.

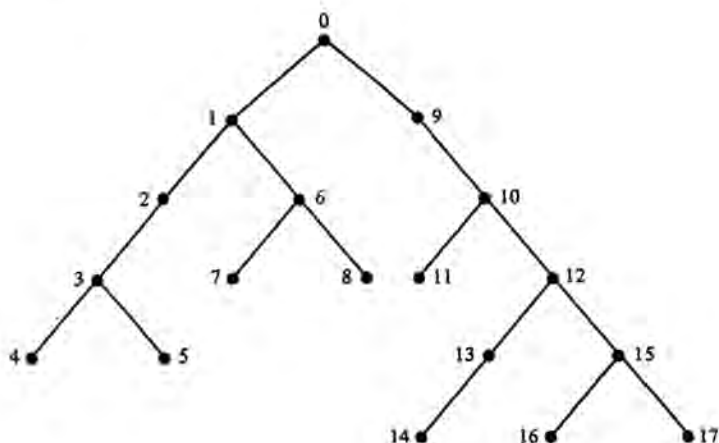
3.5.



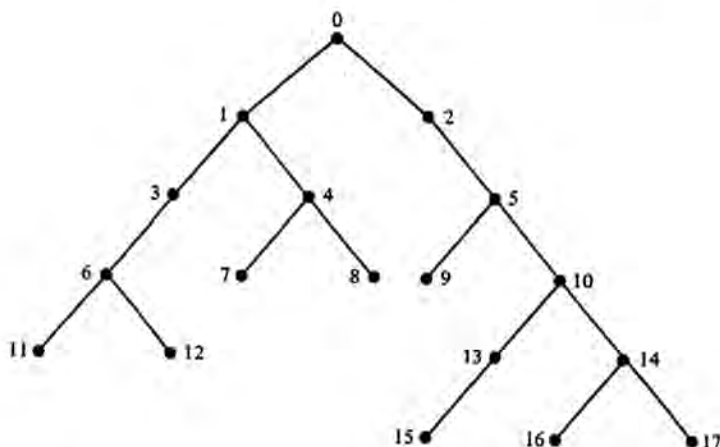
3.6. One minimum spanning tree is



3.8. a)



b)



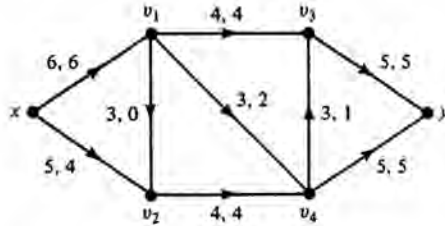
VII.4

- 4.1. If $N = (S, \bar{S})$, then the source must be in S and the sink must be in \bar{S} . For each intermediate vertex, one has the choice of either including it in S or including it in \bar{S} . Therefore, the number of distinct cuts is 2^n .
- 4.2. One can get another maximum flow f' by assigning

$$f'(v_1, v_4) = 2, f'(v_1, v_5) = 2, \text{ and } f'(v_5, v_4) = 1,$$

with $f'(a) = f(a)$ for all other arcs a (where f is the flow in Figure 4.2). Another maximum flow f'' can be obtained by assigning $f''(x, v_3) = 1$, $f''(x, v_2) = 5$, and $f''(v_3, v_2) = 1$, with $f''(a)$ equal to either $f(a)$ or $f'(a)$ for all other arcs a . The minimum cut corresponding to each of these maximum flows is always $\{x, v_2, v_3, v_6\}$. (Note that for f'' , even though the arc (x, v_2) is now saturated, the semipath $x, (x, v_3), v_3, (v_3, v_2), v_2$ is f -unsaturated.)

- 4.5. For this network, there is a unique maximum flow, shown below, and two minimum cuts (for S , take either $\{x, v_2\}$ or $\{x, v_1, v_2, v_3, v_4\}$).



- 4.6. For the maximum flow, assign

$$f(x, v_1) = 5, f(v_1, v_4) = 4, f(v_3, v_4) = 0, f(v_3, y) = 5,$$

with all other arcs having the flow value as shown in Figure 4.4. The corresponding minimum cut is (S, \bar{S}) , where $S = \{x, v_1, v_2, v_3, v_4, v_5\}$. (Note that $x, (x, v_3), v_3, (v_3, v_2), v_2, (v_2, v_5), v_5, (v_5, v_4), v_4, (v_4, v_1), v_1$ is an f -unsaturated x - v_1 semipath.)

Index

- Abelian group, 62
- Abelian groups, Fundamental theorem
 - of finite, 136
- Absorption law, 272
- Acceptor language, 106
- Action of a group on a set, 118
- Acyclic, 322
- Adder
 - full, 282
 - half, 281
- Adjacent, 306, 314
- Adjacency matrix, 307, 315
- Algebraic element, 191
- Algorithm
 - Division, 10, 175
 - Euclidean, 11, 180
 - for computing minimal machine, 49
 - for computing semigroup of a machine, 99
 - for constructing splitting field, 189
 - for constructing standard array, 236
 - for constructing syndrome table, 239
 - greedy, 325
 - maximum flow, 338, 339
 - minimum spanning tree, 325
 - Quine-McCluskey, 293, 298, 300
 - shortest path, 318
- Alphabet, 2
- Alternating group, 112, 113
- AND gate, 280
- Antisymmetric, 30
- Arc, 314
- Associative, 31
- Atom, 275
- Augmenting semipath, 337
- Automaton, 20
- Automorphism
 - of a field, 192
 - of a group, 89
- BCH code, 259
- Binary
 - algebra, 55
 - operation, 31
 - tree, 328
- Bijection, 31

- Binomial coefficient, 5
- Bipartite graph, 312
- Breadth-first search, 328
- Bridge, 330
- Bridges of Königsberg, 305
- Block design, 205
- Boolean algebra, 266
- Boolean product of matrices, 35
- Boolean ring, 145, 273
- Bound
 - Gilbert-Varshamov, 235
 - Hamming, 248
 - Singleton, 235
 - sphere-packing, 248
- Burnside's theorem, 122

- Cancellation laws, 64
- Capacity, 335
 - function, 332
- Cardinality, 2
- Cartesian product, 3
- Cauchy's theorem, 88, 138
- Cayley's theorem, 113
- Center
 - of a graph, 330
 - of a group, 71
- Centralizer, 75
- Channel, 210
- Characteristic, 149
- Check polynomial, 256
- Children, 328
- Chinese remainder theorem, 158, 160
- Ciphertext, 165
- Circuit
 - of a graph, 309
 - parallel, 270
 - series, 270
- Class equation, 135
- Code
 - BCH, 259
 - binary, 214
 - binary Golay, 249, 263
 - block, 214
 - cyclic, 252
 - dual, 224
 - equivalent, 217
 - error-correcting, 211
 - error-detecting, 210
 - extended, 225
 - Golay, 249, 263
 - group, 215
 - Hamming, 235, 244
 - linear, 212, 215
 - parity check, 210, 225
 - perfect, 248
 - repetition, 210
 - self-dual, 224
 - ternary, 215
 - ternary Golay, 249, 263
- Code, group of, 225
- Codeword, 210, 214
- Codomain, 19
- Colorable, 313
- Commutative, 31
 - diagram, 93
 - group, 62
 - monoid, 62
 - ring, 141
 - semigroup, 62
- Complete graph, 311
- Complement, 265
- Composition, 29
- Concatenation, 60
- Congruence relation, 81
- Conjugacy class, 134
- Conjugate, 134
- Conjunction, 266
- Conjunctive normal form, 296
- Connected, 309
 - components, 312
- Content, 182
- Contradiction, 270 392
- Core, 297

- Coset, 76
- Coset decoding, 236
- Coset leader, 236
- Counter, binary mod 3, 23
- Cover
 - of a finite-state machine, 45
 - of a product of literals, 297
- Cryptography, 164
- Cryptosystem, public-key, 165
- Cut, 335
 - edge, 330
 - minimum, 335
 - vertex, 312
- Cycle (in a permutation group), 108
- Cycle (of a graph), 309
 - directed, 315
 - Hamiltonian, 313
- Cycle index, 128
- Cyclic
 - code, 252
 - group, 72
 - semigroup, 69
- Cyclic shift, 252

- Decoding
 - coset, 236
 - maximum likelihood, 228
 - nearest neighbor, 228
- Decoding table, 236
- Degree
 - of a field extension, 185
 - of a polynomial, 172
 - of a vertex, 308, 315
- Decryption, RSA, 167
- Defining table of a machine, 21
- Delay machine, 26
- DeMorgan's laws, 273
- Depth-first search, 328
- Derivative (of a polynomial), 193
- Design, block, 205
- Digraph, 314
 - associated, 320
 - connected, 315
 - disconnected, 321
 - of a relation, 37
 - of a machine, 21
 - simple, 314
 - strongly connected, 321
 - symmetric, 320
 - weighted, 316
- Dihedral group, 59, 106
- Direct product
 - of groups, 68
 - of rings, 159
- Directed
 - cycle, 315
 - graph. *See* Digraph.
 - path, 315
 - trail, 315
 - tree, 327
 - walk, 315
- Disjoint cycle representation, 109
- Disjoint cycles, 108
- Disjunction, 266
- Disjunctive normal form, 290
- Distance
 - Hamming, 226
 - in weighted digraph, 317
- Distributive law, 141, 266
- Division algorithm
 - for integers, 10
 - for polynomials, 175
- Divisor, 10
- Domain of a mapping, 19
- Domain, integral, 142
- Don't care conditions, 302
- Dual code, 224
- Duality, principle of, 266

- Eccentricity, 330
- Edge, 306
 - cut, 330

- directed, 314
- Eisenstein's criterion, 182
- Element
 - algebraic, 191
 - identity, 62
 - order of, 73
 - primitive, 197
- Encryption, RSA, 166
- Enumeration theory, 125
- Epimorphism, 89
- Equivalence class, 40
- Equivalence relation, 30, 39
- Equivalent
 - circuits, 272
 - codes, 217
 - expressions, 269, 290
 - gating networks, 283
 - machines, 46
 - states, 46
- Error-correcting code, 211
- Error-detecting code, 210
- Essential prime implicant, 297
- Euclidean algorithm
 - for integers, 12
 - for polynomials, 180
- Euler, L., 201, 305
- Euler ϕ function, 147
- Euler's theorem, 148
- Eulerian trail, 309
- Even
 - permutation, 111
 - string, 22
- Exponents, laws of, 65
- Extended code, 225
- Factor group, 86
- Factor semigroup, 81
- Factor theorem, 177
- Fast adders, 162
- Fermat's Theorem, 148
- Fibonacci sequence, 16
- Field, 141
 - finite, 147, 185, 192
 - Galois, 147, 195
 - quotient, 146
 - splitting, 187
- Field extension, 185
- Finite abelian groups, fundamental theorem of, 136
- Finite field, 147, 185, 192
- Finite-state machine, 20
- First isomorphism theorem, 93, 155
- Fixed set, 122
- Flow, 333
 - maximum, 335
 - value of, 334
- Forest, 322
- Four-color Theorem, 313
- Four-group, Klein, 67, 116
- Frobenius automorphism, 199
- Full adder, 282
- Function, 19, 30
 - capacity, 332
 - output, 20
 - switching, 284
- Fundamental theorem of algebra, 183
- Fundamental theorem of arithmetic, 14
- Fundamental theorem of finite abelian groups, 136
- G-equivalence, 119
- Galois field, 147, 195
- Galois group, 192
- Gate
 - AND, 280
 - logic, 280
 - NOT, 281
 - OR, 280
- Gating network, 280
- Gauss' lemma, 182
- General linear group, 94

- Generator, 72
 - matrix, 216
 - polynomial, 254
- Generators of a group, 107
- Gilbert-Varshamov bound, 235
- Golay code, 249, 263
- Graph, 306
 - acyclic, 322
 - bipartite, 312
 - complete, 311
 - connected, 309
 - directed, 314
 - k-colorable, 313
 - planar, 313
 - simple, 306
 - weighted, 325
- Graphs, isomorphic, 307
- Greatest common divisor (GCD)
 - for integers, 10
 - for polynomials, 179
- Greatest lower bound, 278
- Greedy algorithm, 325
- Group, 62
 - abelian, 62
 - alternating, 112, 113
 - cyclic, 72
 - dihedral, 59, 106
 - factor, 86
 - Galois, 192
 - Klein four-, 67, 116, 137
 - octahedral, 131
 - permutation, 57, 108
 - quotient, 86
 - symmetric, 57, 108
 - tetrahedral, 133
- Group axioms, 62
- Group code, 215
- Group of a code, 225
- Hamming, R., 209
 - bound, 248
 - code, 235, 244
 - distance, 226
- Hardy, G. H., 164
- Hasse diagram, 37
- Head, 314
- Homomorphism
 - of binary algebras, 89
 - of Boolean algebras, 277
 - of groups, 89
 - of monoids, 89
 - of rings, 154
 - of semigroups, 89
- Huntington, E. V., 266
- Ideal, 152
 - maximal, 157
 - prime, 156
 - principal, 154
- Idempotent element, 76
- Idempotent law, 272
- Identity element, 62, 266
- Image, 19
- Implicant, 292
 - essential prime, 297
 - prime, 292
- Incidence mapping, 306, 314
- Incident, 306, 314
- Inclusion-exclusion, principle of, 18
- Indegree, 315
- Index (of a subgroup), 78
- Index, cycle, 128
- Induction, mathematical, 8, 9
- Information digits, 212
- Injective, 30
- Inner automorphism, 95
- Input alphabet, 20
- Integers modulo n , 146
- Integral domain, 142
- Intersection, 2
- Half-adder, 281
- Hamiltonian cycle, 313

- Inventory, pattern, 129
- Inverse, 62
 - mapping, 33
- Inverter, 281
- Irreducible polynomial, 177
- Isomorphism
 - of Boolean algebras, 277
 - of graphs, 307
 - of groups, 89
 - of monoids, 89
 - of rings, 154
 - of semigroups, 89
- Isomorphism theorem
 - first, 93, 155
 - second, 96
 - third, 97
- Join (in Boolean algebra), 266
- Join (in graph theory), 306, 314
- Karnaugh maps, 295
- Kernel, 92
- Kirkman schoolgirl problem, 205
- Klein four-group, 67, 116, 137
- Knapsack problem, 170
- Königsberg, bridges of, 305
- Lagrange's theorem, 78
- Latin square, 200
 - orthogonal, 201, 250
- Lattice, 279
- Leading coefficient, 172
- Leaf, 323
- Least common multiple, 17
- Least upper bound, 278
- Left coset, 76
- Left discriminative semigroup, 102
- Letters, 2
- Linear code, 212, 215
- Link, 306
- Literal, 282
- Logarithm in a finite field, 199
- Logical expression, 269
- Logic
 - gate, 280
 - network, 280
- Loop, 306, 314
- Lower bound, 278
 - greatest, 278
- Machine
 - finite-state, 20
 - Mealy, 20
 - minimal, 46
 - monoid of, 83
 - Moore, 20
 - parity check, 22
 - semigroup of, 83
 - unit delay, 26
- Machine of a semigroup, 102
- Machines, equivalent, 46
- Mapping, 19, 30
 - incidence, 306, 314
 - injective, 30
 - next-state, 20
 - one-to-one, 30
 - onto, 31
 - surjective, 31
- Mathematical induction, 8, 9
- Matrix
 - adjacency, 307, 315
 - generator, 216
 - of a relation, 34
 - parity check, 220
- Max-flow min-cut theorem, 338
- Maximal ideal, 157
- Maximum flow, 335
 - algorithm, 338, 339
- Maximum likelihood decoding, 228
- Maxterm, 288

- Mealy machine, 20
- Meet, 266
- Minimal machine, 46
- Minimal polynomial, 191
- Minimal sum of products, 297
- Minimum distance of a code, 226
- Minimum spanning tree, 325
 - algorithm, 325
- Minterm, 285
- Mod 3 binary counter, 23
- Monic polynomial, 172
- Monoid, 62
 - of a machine, 83
- Monomorphism, 89
- Moore machine, 20
- Multigraph, 306
- Multiplication table, 57

- Nearest neighbor decoding, 228
- Negation, 269
- Network, 332
 - gating, 280
 - logic, 280
- Next-state mapping, 20
- Nilpotent, 145
- Node, 314
- Normal form
 - conjunctive, 296
 - disjunctive, 290
- Normal subgroup, 84
- Normalizer, 75
- NOT gate, 281

- Octahedral group, 131
- Odd
 - permutation, 111
 - string, 22
- One-time pad, 165
- One-to-one mapping, 30
- Onto mapping, 31

- Operation
 - associative, 31
 - binary, 31
 - commutative, 31
 - unary, 31
- Orbit, 119
- Order, 73
- Ordering, partial, 30, 274
- OR gate, 280
- Outdegree, 315
- Output
 - alphabet, 20
 - function, 20
- Overall parity check, 210, 225

- Parallel circuit, 270
- Parent, 328
- Parity (of strings), 22
- Parity check
 - code, 210, 225
 - equations, 222
 - machine, 22
 - matrix, 220
- Partial order, 30, 274
- Partition
 - of an integer, 137
 - of a set, 41
- Path, 309
 - directed, 315
- Pattern inventory, 129
- Pendant vertex, 323
- Perfect code, 248
- Permutation, 4, 55
 - even, 111
 - odd, 111
- Pigeonhole principle, 5
- Plaintext, 165
- Planar graph, 313
- Pólya enumeration theory, 125
- Polynomial, 172
 - check, 256

- generator, 254
- irreducible, 177
- minimal, 191
- monic, 172
- primitive, 182, 199
- reducible, 177
- Polynomial ring, 173
- Poset, 30
- Power set, 3, 265
- Prime, 13
 - ideal, 156
 - implicant, 292
- Primitive
 - element, 198
 - polynomial, 182, 199
- Principal ideal, 154
- Principal ideal domain, 154
- Principle of duality, 266
- Principle of inclusion-exclusion, 18
- Product
 - of groups, 68
 - of rings, 158
 - of sums, 303
- Product term, 293
- Pseudograph, 306
- Public-key cryptosystem, 165
- Quine-McCluskey algorithm, 293, 298, 300
- Quotient
 - field, 146
 - group, 86
 - ring, 153
 - semigroup, 81
- Range, 19
- Rate of a code, 215
- Reachable, 315
- Redfield-Pólya Theorem, 129
- Reducible polynomial, 177
- Redundancy digits, 212
- Relation, 28
 - antisymmetric, 30
 - congruence, 81
 - digraph of, 37
 - equivalence, 30, 39
 - matrix of, 34
 - on a finite set, 34
 - partial order, 30, 274
 - reflexive, 29
 - symmetric, 29
 - transitive, 30
- Relatively prime
 - integers, 13
 - polynomials, 179
- Ring, 140
 - Boolean, 145, 273
 - commutative, 141
 - of polynomials, 173
 - quotient, 153
 - with identity, 141
- Ring homomorphism, 154
- Root, 174
 - of a tree, 327
 - multiple, 193
 - simple, 193
- Rooted tree, 327
- RSA
 - cryptosystem, 165
 - decryption, 167
 - encryption, 166
- Rule of nine, 151
- Search
 - breadth-first, 328
 - depth-first, 328
- Semiautomaton, 20
- Semigroup, 62
 - cyclic, 69
 - factor, 81
 - quotient, 81

- machine of, 102
- Semigroup of a machine, 83
- Semipath, 336
 - unsaturated, 337
 - augmenting, 337
- Sequence recognizer, 24
- Series circuit, 270
- Set builder notation, 2
- Shannon, C. E., 209, 265
- Shortest path algorithm, 318
- Signature, 169
- Simple
 - digraph, 314
 - graph, 306
- Simulate, 45
- Singleton bound, 235
- Sink, 332
- Source, 332
- Sphere-packing bound, 248
- Spanning
 - subgraph, 325
 - tree, 325
- Special linear group, 94
- Splitting field, 187
- Square, symmetries of, 58
- Stabilizer, 120
- Standard array, 236
- Standard form (of a generator matrix), 218
- Starting state, 104
- State, 19
- State set, 20
- Stater equivalence of, 46
- State triple system, 205, 251
- String
 - even, 22
 - odd, 22
- Subfield, 185
- Subfields of a finite field, 196
- Subgraph, 306
 - spanning, 325
- Subgroup, 68
 - cyclic, 72
 - normal, 84
- Submonoid, 68
- Subring, 144
- Subsemigroup, 68
 - cyclic, 69
- Subset, 2
- Sum of products, minimal, 297
- Surjective, 31
- Switch, 270
- Switching function, 284
- Symmetric difference, 142
- Symmetric group, 57, 108
- Symmetric relation, 29
- Syndrome, 239
 - table, 239
- Tail, 314
- Target, 19
- Tautology, 270
- Tetrahedral group, 133
- Tournament, 316
- Trail, 309
 - directed, 315
 - Eulerian, 309
- Transcendental element, 191
- Transition mapping, 20
- Transitive action, 119
- Transitive relation, 30
- Transposition, 111
- Trapdoor one-way function, 165
- Tree, 322
 - binary, 328
 - directed, 327
 - rooted, 327
- Truth table, 270
- Underlying graph, 315
- Union, 2
- Unique factorization

- of integers, 14
 - of polynomials, 178
- Unit delay machine, 26
- Unit (in a ring), 141
- Unsaturated semipath, 337
- Upper bound, 278
 - least, 278
- Value of a flow, 334
- Vandermonde determinant, 260, 263
- Vertex, 306, 314
 - cut, 312
 - pendant, 323
- Vertices, adjacent, 306, 314
- Walk, 309
 - closed, 309
 - directed, 315
- Weight (in enumeration theory), 128
- Weight
 - of a codeword, 226
 - of an arc, 316
 - of a subgraph, 325
- Weighted
 - digraph, 316
 - graph, 325
 - length, 317
- Well-defined, 80
- Well-formed formula, 269
- Well-ordering principle, 9
- Zero divisor, 141

ISBN 0-06-043878-9

