Information and Coding

Karl Petersen

January 17, 2018

Contents

Part 1

Introduction

This is a collection of materials from a first-year seminar at the University of North Carolina, Chapel Hill, which ran fairly regularly from 2000 to 2014. Prerequisites included high-school mathematics, curiosity, and willingness to deal with unfamiliar ideas. The aim was to review the many facets of information, coding, and cryptography, including their uses throughout history, their implications for modern life, and their mathematical underpinnings. Readings from the books listed in the syllabus were supplemented by these notes and current news articles. Classes consisted of discussions and demonstrations. Students conducted group or individual research projects which resulted in written (and critically edited and revised) papers and presentations to the seminar at the end of the term.

The following syllabus shows how the seminar progressed.

Mathematics 56H, T-Th 11:00-12:15, Phillips 228
Karl Petersen, Mathematics

Fall 2014

Satisfies QI Math Requirement. Students in the Honors Program have priority in registration for this seminar.

Texts:
Simon Singh, *The Code Book,* Doubleday, 1999
Hans Christian von Baeyer, *Information, The New Language of Science,* Phoenix, 2004
James Gleick, *The Information,* Vintage, 2011

Also of interest: Hal Abelson, Ken Leeden, and Harry Lewis, *Blown to Bits: Your Life, Liberty, and Happiness after the Digital Explosion,* Addison-Wesley, 2008

E-Reserve:

1. *For All Practical Purposes*, COMAP, W.H. Freeman, 2000: Chs. 9 & 10; 331-381
2. *Masked Dispatches*, US Government (NSA), 1992: Chs. 1,2,9,10; 11-24, 71-82
3. *Invitation to Cryptology*, Thomas H. Barr, Prentice Hall, 2002, Sections 2.7 and 2.8, 134-158.
4. *Modern Cryptology: A Tutorial*, Gilles Brassard, Springer-Verlag, Lecture Notes in Computer Science 325, 1988: parts of Ch. 5, 40-53, 70-78, Bibliography (91-107)
5. *Privacy on the Line*, W. Diffie and S. Landau, MIT Press, 1998: Ch. 6, 125-150; Ch. 8, 183-203; Ch. 10, 225-245
6. *Symbols, Signals and Noise: The Nature and Process of Communication*, J. R. Pierce, Harper Torchbooks, Harper & Row, New York, 1961: Ch. III, A Mathematical Model, pp. 45-63; Ch. IV, Encoding and Binary Digits, 64-77; Ch. V, Entropy, 78-106; Ch. VIII, The Noisy Channel, 145-165

Online Notes by KEP at http://www.math.unc.edu/Faculty/petersen/:

1. *Counting*
2. *Number Theory and Cryptography*
3. *Elementary Probability*
4. *Shannon's Information Theory*

Office Hours: T, Th 12:30-2 and by appointment, Phillips 300A. E-mail address: petersen@math.unc.edu,

It is common to say that we are now living in the information age. What are the ways in which information is stored, transmitted, presented, and protected? What is information anyway? Topics for this seminar will be drawn from cryptography (secret writing throughout history, including Thomas Jefferson's cipher machine, the German Enigma machine, public-key systems, and security and privacy on the internet) and information theory (entropy, information compression, and error correction). Further topics may include symbolic dynamics (study of symbol streams and associated dynamical systems and formal languages); applications like image compression and processing (compact disks, MP3 and JPEG, transforms, error correction, noise removal); the manipulation and analysis of the huge reams of data now being collected by science, industry, and government (genomes, consumer research, intelligence data); and visualization (how can different kinds of information be vividly and usefully presented, combined, and compared?). These topics are mathematically accessible to anyone with a high-school background and offer many possibilities for experimentation and theoretical exploration.

We will begin by reading, discussing, and working on the texts. There will be some mathematical and computer exercises. (We will use software such as Matlab and Mathematica, but no previous knowledge or experience of software or of programming is assumed.) After developing this background, students will select individual or group projects that could involve encoding and decoding messages, enhancing and compressing images, transforming and filtering signals, measuring properties of information sources (including analysis of artistic and literary objects), investigating current information-related discoveries and issues, and so on. Each project should involve some independent research, experimentation, and exploration and should contain a significant mathematical component. For group projects, the contributions of each individual should be identifiable. Students will present their proposals and results to the seminar orally and in writing.

In this research-exposure course, you will be working with a Graduate Research Consultant, Colin Thomson, who will assist you with the research project. [The GRC Program is sponsored by the Office for Undergraduate Research (www.unc.edu/depts/our), and you may be able to use this research-exposure course to meet a requirement of the Carolina Research Scholars Program (http://www.unc.edu/depts/our/students/students_crsp.html).] I encourage you to visit the OUR website to learn about how you can engage in research, scholarship and creative performance while you are at Carolina.

There will be a Quiz Tues. Sept. 16 and an Exam Tues. Nov. 18. The Final Exam is scheduled for Thurs. Dec. 11 at 12 PM. However, in view of the special seminar nature of this course, where the students are intensively involved throughout the semester and prepare and present substantial individual research projects, the scheduled final exam will be replaced by an alternative final assessment. The nature of this assessment will be specified later in the semester. Part of the final exam time may be used for final project presentations.

We will agree on an extra regular weekly meeting time (such as late afternoon, say 5-7, Wednesdays) for out-of-class experiences, such as installing Matlab and Mathematica, working on assignments together, viewing films and videos, and so on.

Week	Topics	Readings	Comment, Assignment
Aug. 19	Information, coding, history, mathematics. Mary, Queen of Scots. Overview of ciphers.	Singh 1; Counting 1-3; Gleick Prologue	Classes start Tues. Aug. 19. Counting 2.1-2.4.
Aug. 26	Vigenère (polyalphabetic) ciphers, cipher machines.	Singh 2-3; Counting 4-5	Counting 3.1-3.3, 4.1-4.4.
Sept. 2	Modular arithmetic, Enigma machine	Singh 4; Number Theory 1	Counting 5.1-5.4. Number Theory 1.1. No classes Mon. Sept. 1.
Sept. 9	Friedman and Kasiski tests, probability, languages	Singh 5 ; Barr 2.7; Prob Notes 1-2; Gleick 2-4	Prob 1.1-1.8, 2.1-2.5. Matlab Assgt. 1 (basics). Barr 2.7: 2.
Sept. 16	Cryptanalysis of Vigenère	Barr 2.8: pp. 143-top of 149. vB Prologue, 1-6,8	Barr 2.8: 1-3 (use M-files and Caesar spreadsheet). Quiz Tues. Sept. 16. Movie Wed. Sept. 17.
Sept. 23	Euclidean algorithm, primes, modular inverses. Information, physics, philosophy.	Number Theory Notes 2, 3; Gleick 8-11	Number Theory, Ex. 2-8. Project proposals.
Sept. 30	Public keys, RSA	Singh 6; FAPP 370-376; Number Theory Notes 4, 5, 6	Number Theory Notes Ex. 10, 11, 13, 14. FAPP p. 379, 15-18.
Oct. 7	Applications of one-way functions, policy issues	Singh 7; Number Theory Notes 7; Diffie-Landau 6, 8; Abelson et al. Ch. 2; Gleick 14, 15	Matlab Assgt. 1 Challenges. Univ. Day Sun. Oct. 12.
Oct. 14	Quantum computing	Singh 8, vB 19-23, Gleick 13	Fall Break Oct. 16-17.
Oct. 21	Probability and	vB 9-14; Prob Notes	Prob. Notes 3.1, 4.1, 4.2,5.1,

	information	3-7. Gleick 12.	5.2, 6.1-6.3. Project progress reports. Due Tues. Oct. 21.
Oct. 28	Stationary sources, data compression	Pierce 3, 4; Info Notes 1,2,7; Gleick 1,5; FAPP 358-370, 331-350.	FAPP p. 379, 20-24. Info Notes 1.1. Prob. Notes 7.1.
Nov. 4	Entropy and error correction	Pierce 5; Info Notes 3,4; Gleick 6	FAPP p. 353: 7, 11, 12; p. 378, 7-11.
Nov. 11	Shannon's theorems	Info Notes 5; Gleick 7	Info Notes 3.1, 3.2, 4.1, 5.1
Nov. 18	Exam and first presentation?		Exam Tues. Nov. 18
Nov. 25	Presentations start Thurs. Nov. 20 or Tues. Nov. 25		Project papers due Tues. Nov. 25. TG Nov. 27 and 28.
Dec. 2	Presentations		Classes end Wed. Dec. 3. Revised project papers due Thurs. Dec. 4 (Reading Day)
Dec. 9			Final exam time: Thurs. Dec. 11, 12 PM.

Honor System: Students in this course are bound by the UNC Honor System. You may (and probably should) work together on class preparation, homework, projects, and exam preparation, but papers should clearly indicate the contributions of each individual and should properly credit any sources used. Exams will be closed-book individual efforts. Students are asked to sign the Pledge at the end of each exam to attest that they followed the Honor System while taking it.

Homework Problems: Due Fridays (with exceptions for holidays, etc.). Papers are to be left in the wooden mailbox marked K. Petersen opposite Ph348 (not the metal mailbox in Ph327) before 1:00. Late papers will be given some credit at the discretion of the grader--just leave them in the box whenever they are ready. Please turn in papers that are neat and written so as to be coherent and easily readable, not first drafts with scribbles and scratch-outs. We want correct, clear, and efficient write-ups. Even for problems (or parts of problems) that require just direct calculations, include explanations, in grammatical English, of what you are doing, citing supporting formulas or theorems for the most significant steps. To achieve an acceptably high level of presentation, you will usually have to revise your paper after an initial draft, which already may be following several

attempts at solving a problem. See the notes “Writing up Mathematics” on the instructor’s website for an example and more details.

Discussion Leading: We will establish a rotation of teams of two (maybe sometimes one or three) students to animate our discussions of the reading and work that we do outside of class. The leaders will be prepared to raise questions, which may be open-ended or about details. They will also seek to go beyond the assigned reading to other sources and will try to come up with examples, activities, or commentaries related to the ideas involved. Each week's leaders should *meet with the instructor in advance on Friday or Monday* to discuss ideas for questions and activities.

About the Instructor: Karl Petersen was born in Tallinn, Estonia, and grew up in East Orange, New Jersey. His degrees are from Princeton and Yale, and he has held visiting positions at universities in Austria, Chile, France, and India. Petersen's research area is ergodic theory, a fairly new branch of mathematics which applies probability and analysis to study the long-term average behavior of complicated systems, with applications ranging from celestial dynamics through interactions of biological populations to the efficient transmission and recording of information. Favorite activities include tennis and hiking.

Part 2

A few questions

1. It's a dark and stormy night in the North Atlantic in 1942. As admiral at base, you have just read decoded intercepted Enigma-encoded messages which detail the time and place of a wolfpack attack on a huge and important convoy. Do you divert it?

2. How can you flip a coin (fairly) via a long-distance phone call?

3. Can there be a cryptosystem in which everyone can encode messages to you, and see each other's encoded messages to you, but only you can decode and read them? What special advantages would any such system have?

4. Can every imaginable cryptosystem be broken? (One-time pad, quantum systems, RSA, etc.)

5. Should the government be able to monitor all e-mail (or phone conversations) in the interest of national security? Should use (or export) of cryptosystems be limited? Should there be a key escrow system?

6. Do you (should you?) have the right to decrypt (and copy) DVD's that you buy?

7. Can we measure the amount of information contained in a message?

8. How can information be compressed, so that it can be transmitted or stored more quickly, efficiently, or cheaply?

9. How can information be protected against errors in transmission?

10. How can useful information be extracted from huge piles of data (for example, from the reams of intercepted e-mails that the government might pile up)?

Part 3

Writing

When writing up mathematics, make sure to embed the mathematics in English-language explanations. Use the proofs and explanations in the Notes on Number Theory and Cryptography as a guide to style.

Example 1: Prove that the sum of any two odd numbers is even.

Preliminary work (not of hand-in quality):

m odd $\implies m = 2j + 1$ $(2j + 1) + (2k + 1) = 2j + 2k + 2 = 2(j + k + 1)$, even!

Final write-up:

Proposition: The sum of any two odd numbers is even.

Proof: Let m and n be odd numbers. Then there are integers j and k such that $m = 2j + 1$ and $n = 2k + 1$. Thus

$$m + n = (2j + 1) + (2k + 1) = 2j + 2k + 2 = 2(j + k + 1),$$

which is even, since it is twice an integer.

[Remark: We assume that the definition of an odd number is one that is twice an integer plus 1.]

Example 2: Find the value of Euler's phi function at 21.

Preliminary work (not of hand-in quality):

$$\phi(21) =?$$

prime factorization: $21 = 3 * 7$

$\phi(21) = (3 - 1) * (7 - 1)$

$= 2 * 6$

$= 12$

Final write-up:

What is the value of the Euler phi-function at 21?

Solution: Since 21 is the product of the two primes 3 and 7, and we know from Exercise 3 of the "Notes on Number Theory and Cryptography" that for distinct primes p and q we have $\phi(pq) = (p - 1)(q - 1)$, we find that

$$\phi(21) = (3 - 1) * (7 - 1) = 2 * 6 = 12.$$

Comments:

1. Notice that the write-ups include brief statements of the problems.

2. What is on the page can be read aloud and understood as correct English prose.

3. Relevant reasons are supplied to support the most important steps. (What is important is often related to what the "point" of the problem is–what has recently been learned that the problem is using or illustrating.)

4. Special attention is paid to the key logical constructs "if...then", "if and only if", "there exists", and "for every", as well as the important words "and", "or", and "not". (How many of these came up in these examples?)

5. Why all this fuss?

(a) We are not just trying to get an answer, but to understand the process that produces the answer.

(b) We have to convince others (for example, the grader) that our reasoning is correct.

(c) It is nice to have a clear, complete, readable solution in hand to consult later, for example when working on other problems or reviewing for exams.

(d) The discipline of writing something down clearly and completely forces clear and complete understanding. First of all, it is a test of understanding (if you can't get it down clearly in writing, you don't really understand it). Second, the process of writing, criticizing, and revising is actually one of the best tools for PRODUCING better understanding.

Part 4

Counting

It is important to be able to count exactly the number of elements in any finite set. We will see many applications of counting as we proceed (number of Enigma plugboard arrangements, computer passwords of a certain form, telephone area codes, automobile license plates, binary strings of a certain length needed to encode uniquely each possible English message of a certain length, etc.). In this section we list some of the basic techniques that are helpful for counting even in complicated situations.

Notation. If A is any set, we denote by $|A|$ the number of elements in A. (If A is infinite, we write $|A| = \infty$.) $\mathbb{N} = \{1, 2, \dots\}$ denotes the set of *natural numbers*.

Remark. Thus $|A| = n \in \mathbb{N}$ if and only if A can be put in one-to-one correspondence with $\{1, \dots, n\}$, i.e., "counted". In the foundations of mathematics, following Georg Cantor, one defines the concept of cardinal number as follows. Two sets A and B are said to have the *same cardinal number* if they can be put in one-to-one correspondence, i.e., if there is a one-to-one onto function $f : A \to B$. Then even some infinite sets can be seen to have the same or different cardinal numbers. For example, the set $\mathbb{N} = \{1, 2, \dots\}$ of natural numbers, the set $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, 3, \dots\}$ of integers, and the set $\mathbb{Q}$ of rational numbers all have the same cardinal number—they are all *countable*—while the set $\mathbb{R}$ has a different (in fact larger) cardinal number than $\mathbb{N}$ and so is *uncountable*. A cardinal number is then an equivalence class under this concept of sameness. For example, 2 is the set of all sets that can be put in one-to-one correspondence with the set $\{\emptyset, \{\emptyset\}\}$.

4.1 Disjoint sets

If A and B are disjoint sets, i.e., $A \cap B = \emptyset$, then $|A \cup B| = |A| + |B|$.

Example 4.1.1. If $A = \{2, 4, 8\}$ and $B = \{1, 5, 11, 12\}$, then $|A \cup B| = 3 + 4 = 7$.

Example 4.1.2. Here's a useful everyday example: If you have 8 shirts in the laundry and 2 in the drawer, then you have at least 10 shirts.

4.2 Cartesian products

Recall that the *Cartesian product* $A \times B$ of two sets A and B is the set of all ordered pairs with the first element from A and the second element from B:

$$A \times B = \{(a, b) : a \in A, b \in B\}. \tag{4.1}$$

By repeating this construction we can build up the Cartesian product of n sets for any $n \geq 2$:

$$\begin{gathered} A_1 \times A_2 \times \cdots \times A_n = \\ \{(a_1, a_2, \dots . a_n) : a_1 \in A_1, a_2 \in A_2, \dots a_n \in A_N\}. \end{gathered} \tag{4.2}$$

Proposition 4.2.1.

$$|A \times B| = |A|\,|B| \tag{4.3}$$
$$|A_1 \times A_2 \times \cdots \times A_n| = |A_1|\,|A_2| \ldots |A_n|.$$

Example 4.2.1. How many "words" (arbitrary strings) are there on the 26 letters of the English alphabet consisting of a vowel followed by a consonant? Answer: $5 \cdot 21 = 105$.

Example 4.2.2. A (cheap) combination lock has three independently turning wheels, each of which can be set in any one of 8 different positions. How many possible combinations are there? Answer: $8 \cdot 8 \cdot 8 = 512$.

Example 4.2.3. How many license plate "numbers" can be made of three English letters followed by three digits? Answer: $26^3 \cdot 10^3 = 17,576,000$.

It is easy to see why Formula (4.2) is correct. Imagine n wheels on a spindle, like a combination lock, with n_i possible settings for wheel number $i, i = 1, \ldots, n$. For each of the n_1 settings of the first wheel, there are n_2 settings of the second, thus $n_1 n_2$ ways to set the first pair of wheels. For each of these settings of the first two wheels, there are n_3 ways to set the third wheel, hence $n_1 n_2 n_3$ ways to set the third, and so on.

Exercise 4.2.1. Let L denote the number of license plate "numbers" that can be made of three English letters followed by four digits, and let N denote the number that can be made of four English letters followed by three digits. Find L/N—*without* first computing L or N.

Exercise 4.2.2. How many *binary strings* (words on the alphabet $\{0, 1\}$) are there of length 7? How many are there of length less than or equal to 7 (counting the empty word, which has length 0)?

Exercise 4.2.3. A student shows up unprepared for a multiple-choice exam that has 10 questions with 4 possible answers for each one. The student takes random guesses on all the questions but gets a perfect score. How surprised should the student be?

Exercise 4.2.4. How many functions are there from a set with 4 elements to a set with 7 elements?

4.3 Permutations

A *permutation* of a set $A = \{a_1, \ldots, a_n\}$ is an ordering of that set.

Remark 4.3.1. Note that by definition the elements of any set are distinct. Thus there is no such set as $\{1, 1, 2\}$. Note also that the the set $\{1, 2, 3\}$ is the *same* as the set $\{2, 3, 1\}$.

Example 4.3.1. If $A = \{a, b, c\}$, then *abc* and *bac* are two different permutations of A.

Example 4.3.2. Permutations of $\{1, \ldots, n\}$ correspond to one-to-one onto functions $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$. To any such function π let correspond the ordering $\pi(1) \ldots \pi(n)$. And given a permutation $i_1 \ldots i_n$ of $\{1, \ldots, n\}$, for each $j = 1, \ldots, n$ define $\pi(j) = i_j$.

Proposition 4.3.1. *The number of permutations of any set with n elements is $n! = n(n-1) \cdots 2 \cdot 1$. ($1! = 1, 0! = 1$.)*

Proof. There are n choices for the first element in the ordering, and for each choice of the first element there are $n - 1$ choices for the second, hence there are $n(n-1)$ ways to choose the first two elements. Then for each of the choices of the first two elements, there are $n - 2$ ways to choose the third, and so on. □

Example 4.3.3. The three (different) scramblers of an Enigma machine can be placed into their three slots in $3! = 3 \cdot 2 \cdot 1 = 6$ ways. The 6 arrangements are:

123 132 213 231 312 321.

An *n-set* is any set with n elements.

Definition 4.3.1. An *r-permutation of an n-set* ($n, r \in \mathbb{N}, 0 \leq r \leq n$) is an ordering of an r-element subset of the n-set.

Example 4.3.4. Let $A = \{1, 2, 3\}$ be a set with $n = 3$ elements, and let $r = 2$. Then the 2-permutations of A are

$$12 \quad 21 \quad 13 \quad 31 \quad 23 \quad 31.$$

Proposition 4.3.2. *The number of r-permutations of an n-set is*

$$P(n, r) = n(n-1)\cdots(n-r+1). \tag{4.4}$$

Proof. The proof is the same as for Proposition 4.3.1. □

Example 4.3.5. The senior class wants to choose a president, vice president, and secretary from among its 4000 members. How many ways are there to do this (assuming no person can serve simultaneously in two positions)? Answer: $4000 \cdot 3999 \cdot 3998$ ways.

Exercise 4.3.1. A weekly bridge group decides it should get organized by having each member serve either as scheduler, food provider, treasurer, or scorekeeper, and that in order to even out the workloads they should have a new arrangement every month. How long can this four-member group keep this up without repeating an arrangement?

Exercise 4.3.2. How many 4-letter words (arbitrary strings) are there on the 26 letters of the English alphabet? How many are there which use 4 different letters?

Exercise 4.3.3. A baseball team has 17 players. How many ways are there to choose a starting lineup of 9 players plus determine the order in which they will bat? How many ways are there to do this and in addition assign to each player one of the 9 defensive positions (catcher, pitcher, first base, etc.)?

4.4 Subsets (Combinations)

By an *r-subset* of a set A we mean any set $B \subset A$ such that $|B| = r$, i.e., B has r elements. Notice that the elements of any r-subset are *unordered*, as they are for any set.

Proposition 4.4.1. *Let $n, r \in \mathbb{N}$ with $0 \leq r \leq n$. The number of r-subsets of a set with n elements is*

$$C(n, r) = \binom{n}{r} = \frac{n!}{r!(n-r)!}. \tag{4.5}$$

Proof. We sneak up on the result—counting the r-subsets of a set A with n elements—by counting instead, in a different way than before, the number $P(n, r)$ of *r-permutations* of A. We know already that

$$P(n, r) = n(n-1)\cdots(n-r+1).$$

But each r-permutation of A is formed by first selecting an r-subset of A (there are $C(n, r)$ ways to make this choice) and then ordering that particular r-element subset (there are $r!$ ways to do this). Therefore

$$P(n, r) = C(n, r) \cdot r!, \tag{4.6}$$

and hence

$$C(n, r) = \frac{n!}{r!(n-r)!}. \tag{4.7}$$

□

Example 4.4.1. The number of 2-subsets of a 4-set is $4!/(2!2!) = 6$. If the 4-set is $A = \{1, 2, 3, 4\}$, then these 2-subsets are

$$\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}.$$

Note that each of them can be ordered in $r! = 2$ ways, leading to $6 \cdot 2 = P(4, 2) = 4 \cdot 3$ 2-permutations of A.

$C(n, r) = \binom{n}{r}$ is sometimes read "n choose r", since it gives the number of ways to choose an unordered set of r objects from a pool of n. These numbers are also known as *binomial coefficients*, because of the following Proposition.

Proposition 4.4.2. *For any $n \in \mathbb{N}$ and any $a, b \in \mathbb{R}$,*

$$(a + b)^n = \sum_{r=0}^{n} C(n, r) a^r b^{n-r}. \tag{4.8}$$

The binomial coefficients have many fascinating properties, the most basic of which is the *Pascal Identity*:

Proposition 4.4.3. *Let $n, k \in \mathbb{N}$ with $k \leq n$. Then*

$$C(n + 1, k) = C(n, k - 1) + C(n, k). \tag{4.9}$$

Proof. This equation can be checked easily by calculation, using the definition of $C(n, k)$ as a quotient of factorials. But the following *combinatorial proof* can give more insight into the idea behind the identity.

Let A be a set with $n + 1$ elements, fix a particular $a \in A$, and let $B = A \setminus \{a\}$, so that B consists of all elements of A except a, and $|B| = n$.

We want to count the k-element subsets of A. These fall into two disjoint classes, the ones that contain a and the ones that don't contain a. The ones that contain a are obtained by choosing a $(k-1)$-element subset of B—there are $C(n, k - 1)$ ways to do this—and then adding a to it. The ones that do not contain a are obtained by choosing a k-element subset of B—and there are $C(n, k)$ ways to do this. □

This relation leads to the familiar and fascinating *Pascal Triangle*, in which each entry is the sum of the two immediately above it and the n'th row lists the coefficients in the expansion of $(a + b)^n$:

$$\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
 & & & & & \vdots & & & & &
\end{array}$$

Exercise 4.4.1. From a basketball team consisting of 12 players, how many ways are there to select a starting lineup consisting of five players? How many ways are there to select a starting lineup of five, if each player is assigned a different position—center, power forward, small forward, shooting guard, or point guard?

Exercise 4.4.2. How many binary strings of length seven have exactly three 0's?

Exercise 4.4.3. An intramural basketball team consists of four women and six men. The games are four on four. How many ways are there to pick a starting lineup that includes at least one man and at least one woman?

Exercise 4.4.4. An Enigma machine plugboard joins up 6 different pairs of the 26 letters. (For example a is joined to x, k to r, m to t, o to v, q to y, and s to z.) How many different plugboard settings are there?

Exercise 4.4.5. How many license plates that have three letters followed by three digits use no letter or digit twice?

Exercise 4.4.6. From a Congressional committee consisting of 13 Republicans and 12 Democrats it is necessary to choose a subcommittee of 4 Republicans and 3 Democrats. How many possibilities are there?

Exercise 4.4.7. Construct a combinatorial argument to prove that if $k, r, n \in \mathbb{N}$ with $k \leq r \leq n$, then $C(n, r)C(r, k) = C(n, k)C(n - k, r - k)$.

4.5 A Few More Counting Principles

4.5.1 Inclusion-Exclusion

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|. \tag{4.10}$$

This is because when we add $|A_1|$ and $|A_2|$, the elements that are in both sets, i.e. in $A_1 \cap A_2$, are counted in twice.

$$\begin{aligned} |A_1 \cup A_2 \cup A_3| &= |A_1| + |A_2| + |A_3| \\ -(|A_1 \cap A_2| &+ |A_1 \cap A_3| + |A_2 \cap A_3|) \\ &+ |A_1 \cap A_2 \cap A_3|. \end{aligned} \tag{4.11}$$

This is because in forming $|A_1| + |A_2| + |A_3|$ the elements that are in more than one of the sets A_i are counted in either twice, if they are in exactly two of these sets, or three times if they are in all three sets. Subtracting off $|A_1 \cap A_2| + |A_1 \cap A_3| + |A_2 \cap A_3|)$ accomplishes the correction for elements that are in exactly two of the sets, since it subtracts off the extra 1 for each of them. But this term subtracts off 3 for each element that is in all 3 sets, so the cardinality of $A_1 \cap A_2 \cap A_3$ has to be added back in.

A similar formula applies to the cardinality of the union of n sets for any $n \in \mathbb{N}$.

Example 4.5.1. Let's determine how many words (arbitrary strings) of length 6 on the 26 English letters either start with *a* or end with *ed*. The number starting with *a* is the number of arbitrary 5-letter words which can be appended to the initial *a*, namely 26^5. The number that end with *ed* is 26^4. And the number that start with *a and* end with *ed* is 26^3. So the answer is $26^5 + 26^4 - 26^3$.

Exercise 4.5.1. How many functions are there from $\{1, 2, \ldots, n\}$ to $\{0, 1\}$ which assign 0 to either 1 or n?

4.5.2 Counting the complement

Sometimes it is easier to count the number of elements that are *not* in a set A and then subtract from the total number to find out how many are in A.

Example 4.5.2. How many binary strings of length 10 contain at least one 1? Well, there are 2^{10} binary strings of length 10 and only one (all 0's) that does not contain any 1's, so the answer is $2^{10} - 1$.

Exercise 4.5.2. How many strings of 3 decimal digits do not contain the same digit 3 times?

Exercise 4.5.3. How many strings of 8 decimal digits contain a repeated digit?

4.5.3 The Pigeonhole Principle

This principle is so obvious that it hardly seems worth talking about:

Pigeonhole Principle. *If $n+1$ objects are placed into n boxes (or "pigeonholes"–see the array next to the elevator on the third floor of Phillips Hall), then at least one of the boxes has to contain at least two of the objects.*

Example 4.5.3. How many strings of 12 decimal digits contain a repeated digit? Answer: All 10^{12} of them. Think of each of the 12 places in the string as an "object" which is placed into one of the "boxes" 0,1,2,3,4,5,6,7,8,9 according to the entry in the string at that place. Since $12 > 10$, at least two of the 12 objects have to end up in the same box.

Exercise 4.5.4. Early one morning you are groping in the dark in a drawer full of unpaired blue and black socks. How many do you have to pull out in order to be certain that you have two that match?

Exercise 4.5.5. Six people are stranded on an island. Each pair of the six consists either of two allies or two enemies. Show that there must exist either a clique of three mutual allies or a set of three mutual enemies.

Exercise 4.5.6. For each real number x, denote by $\lfloor x \rfloor$ the greatest integer less than or equal to x, by $(x) = x - \lfloor x \rfloor$ the fractional part of x, and by $d(x, \mathbb{Z})$ the distance from x to the nearest integer.

Let $N \in \mathbb{N}$, and let α be any irrational number. Show that there is $q \in \mathbb{N}$ with $q \leq N$ such that $d(q\alpha, \mathbb{Z}) < 1/N$.

Hint: Consider the "objects" $0, (\alpha), (2\alpha), \ldots (N\alpha)$ and the "boxes" $\{x : i/N \leq x < (i+1)/N\}, i = 0, 1, \ldots, N-1$.

Part 5

Some elementary probability

5.1 Probability spaces

Probability theory is an attempt to work mathematically with the relative uncertainties of random events. In order to get started, we do not attempt to estimate the probability of occurrence of any event but instead assume that somehow these have already been arrived at and so are given to us in advance. These data are assembled in the form of a *probability space* $(X, \mathcal{B}, P)$, which consists of

1. a set X, sometimes called the *sample space*, which is thought of as the set of all possible *states* of some system, or as the set of all possible *outcomes* of some experiment;
2. a family $\mathcal{B}$ of subsets of X, which is thought of as the family of *observable events*; and
3. a function $P : \mathcal{B} \to [0, 1]$, which for each observable event $E \in \mathcal{B}$ gives the probability $P(E)$ of occurrence of that event.

While the set X of all possible outcomes is an arbitrary set, for several reasons, which we will not discuss at this moment, the set $\mathcal{B}$ of observable events is not automatically assumed to consist of *all* subsets of X. (But if X is a *finite* set, then usually we do take $\mathcal{B}$ to be the family of all subsets of X.)

We also assume that the family $\mathcal{B}$ of observable events and the probability measure P satisfy a minimal list of properties which permit calculations of probabilities of combinations of events:

1. $P(X) = 1$
2. $\mathcal{B}$ contains X and is is closed under the set-theoretic operations of union, intersection, and complementation: if $E, F \in \mathcal{B}$, then $E \cup F \in \mathcal{B}, E \cap F \in \mathcal{B}$, and $E^c = X \setminus E \in \mathcal{B}$. (Recall that $E \cup F$ is the set of all elements of X that are either in E or in F, $E \cap F$ is the set of all elements of X that are in both E and F, and E^c is the set of all elements of X that are not in E.)

   In fact, in order to permit even more calculations (but not *too* many) we suppose that also the union and intersection of *countably many* members of $\mathcal{B}$ are still in $\mathcal{B}$.
3. If $E, F \in \mathcal{B}$ are disjoint, so that $E \cap F = \emptyset$, then $P(E \cup F) = P(E) + P(F)$. In fact, we assume that P is *countably additive*: if $E_1, E_2, \dots$ are pairwise disjoint (so that $E_i \cap E_j = \emptyset$ if $i \neq j$), then

$$P(\cup_{i=1}^{\infty} E_i) = P(E_1 \cup E_2 \cup \dots) = P(E_1) + P(E_2) + \dots = \sum_{i=1}^{\infty} P(E_i). \tag{5.1}$$

Example 5.1.1. In some simple but still interesting and useful cases, X is a finite set such as $\{0, \dots, d-1\}$ and $\mathcal{B}$ consists of all subsets of X. Then P is determined by specifying the value $p_i = P(i)$ of each individual point i of X. For example, the single flip of a fair coin is modeled by letting $X = \{0, 1\}$, with 0 representing the outcome heads and 1 the outcome tails, and defining $P(0) = P(1) = 1/2$. Note that

the probabilities of all subsets of X are then determined (in the case of the single coin flip, $P(X) = 1$ and $P(\emptyset) = 0$).

Exercise 5.1.1. Set up the natural probability space that describes the roll of a single fair die and find the probability that the outcome of any roll is a number greater than 2.

Exercise 5.1.2. When a pair of fair dice is rolled, what is the probability that the sum of the two numbers shown (on the upward faces) is even?

Exercise 5.1.3. In a certain lottery one gets to try to match (after paying an entry fee) a set of 6 different numbers that have been previously chosen from $\{1, \ldots, 30\}$. What is the probability of winning?

Exercise 5.1.4. What is the probability that a number selected at random from $\{1, \ldots, 100\}$ is divisible by both 3 and 7?

Exercise 5.1.5. A fair coin is flipped 10 times. What is the probability that heads comes up twice in a row?

Exercise 5.1.6. Ten fair coins are dropped on the floor. What is the probability that at least two of them show heads?

Exercise 5.1.7. A fair coin is flipped ten times. What is the probability that heads comes up at least twice?

Exercise 5.1.8. Show that if E and F are observable events in any probability space, then

$$P(E \cup F) = P(E) + P(F) - P(E \cap F). \tag{5.2}$$

5.2 Conditional probability

Let $(X, \mathcal{B}, P)$ be a probability space and let $Y \in \mathcal{B}$ with $P(Y) > 0$. We can restrict our attention to Y, making it the set of possible states or outcomes for a probability space as follows:

1. The set of states is $Y \subset X$ with $P(Y) > 0$;
2. The family of observable events is defined to be
$$\mathcal{B}_Y = \{E \cap Y : E \in \mathcal{B}\}; \tag{5.3}$$
3. The probability measure P_Y is defined on $\mathcal{B}_Y$ by
$$P_Y(A) = \frac{P(A)}{P(Y)} \quad \text{for all } A \in \mathcal{B}_Y. \tag{5.4}$$

Forming the probability space $(Y, \mathcal{B}_Y, P_Y)$ is called "conditioning on Y". It models the revision of probability assignments when the event Y is known to have occurred: we think of $P_Y(A)$ as the probability that A occurred, given that we already know that Y occurred.

Example 5.2.1. When a fair die is rolled, the probability of an even number coming up is $1/2$. What is the probability that an even number came up if we are told that the number showing is greater than 3? Then out of the three possible outcomes in $Y = \{4, 5, 6\}$, two are even, so the answer is $2/3$.

Definition 5.2.1. For any (observable) $Y \subset X$ with $P(Y) > 0$ and any (observable) $E \subset X$ we define the *conditional probability of E given Y* to be

$$P(E|Y) = \frac{P(E \cap Y)}{P(Y)} = P_Y(E). \tag{5.5}$$

Exercise 5.2.1. A fair coin is flipped three times. What is the probability of at least one head? Given that the first flip was tails, what is the probability of at least one head?

Exercise 5.2.2. From a group of two men and three women a set of three representatives is to be chosen. Each member is equally likely to be selected. Given that the set includes at least one member of each sex, what is the probability that there are more men than women in it?

Definition 5.2.2. The observable events A and B in a probability space $(X, \mathcal{B}, P)$ are said to be *independent* in case

$$P(A \cap B) = P(A)P(B). \tag{5.6}$$

Notice that in case one of the events has positive probability, say $P(B) > 0$, then A and B are independent if and only if

$$P(A|B) = P(A); \tag{5.7}$$

that is, knowing that B has occurred does not change the probability that A has occurred.

Example 5.2.2. A fair coin is flipped twice. What is the probability that heads occurs on the second flip, given that it occurs on the first flip?

We model the two flips of the coin by bit strings of length two, writing 0 for heads and 1 for tails on each of the two flips. If Y is the set of outcomes which have heads on the first flip, and A is the set that has heads on the second flip, then

$$\begin{aligned} X &= \{00, 01, 10, 11\}, \\ Y &= \{00, 01\}, \quad \text{and} \\ A &= \{10, 00\}, \end{aligned}$$

so that $A \cap Y = \{00\}$ includes exactly one of the two elements of Y. Since each of the four outcomes in X is equally likely,

$$P(A|Y) = \frac{P(A \cap Y)}{P(Y)} = \frac{|A \cap Y|}{|Y|} = \frac{1}{2} = P(A).$$

Thus we see that A and Y are independent.

This example indicates that the definition of independence in probability theory reflects our intuitive notion of events whose occurrences do not influence one another. If repeated flips of a fair coin are modeled by a probability space consisting of bit strings of length n, all being equally likely, then an event whose occurrence is determined by a certain range of coordinates is independent of any other event that is determined by a disjoint range of coordinates.

Example 5.2.3. A fair coin is flipped four times. Let A be the event that we obtain a head on the second flip and B be the event that among the first, third, and fourth flips we obtain at least two heads. Then A and B are independent.

Exercise 5.2.3. Show that the events A and B described in the preceding example will be independent whether or not the coin being flipped is fair.

Exercise 5.2.4. Show that events A and B described in the preceding example will be independent even if the probability of heads could be different on each flip.

Exercise 5.2.5. When a pair of fair dice is rolled, is the probability of the sum of the numbers shown being even independent of it being greater than six?

5.3 Bayes' Theorem

Looking at the definition of conditional probability kind of backwards leads very easily to a simple formula that is highly useful in practice and has profound implications for the foundations of probability theory (frequentists, subjectivists, etc.). We use the notation from [**?**], in which C is an event, thought of as a *cause*, such as the presence of a disease, and I is another event, thought of as the existence of certain *information*. The formula can be interpreted as telling us how to revise our original estimate $P(C)$ that the cause C is present if we are given the information I.

Theorem 5.3.1 (Bayes' Theorem)**.** *Let $(X, \mathcal{B}, P)$ be a probability space and let $C, I \in \mathcal{B}$ with $P(I) > 0$. Then*

$$P(C|I) = P(C)\frac{P(I|C)}{P(I)}. \tag{5.8}$$

Proof. We just use the definitions of the conditional probabilities:

$$P(C|I) = \frac{P(C \cap I)}{P(I)}, \qquad P(I|C) = \frac{P(I \cap C)}{P(C)} \tag{5.9}$$

and the fact that $C \cap I = I \cap C$. □

Example 5.3.1. We discuss the example in [**?**, p. 77] in this notation. C is the event that a patient has cancer, and $P(C)$ is taken to be .01, the incidence of cancer in the general population for this example taken to be 1 in 100. I is the event that the patient tests positive on a certain test for this disease. The test is said to be 99% accurate, which we take to mean that the probability of error is less than .01, in the sense that $P(I|C^c) < .01$ and $P(I^c|C) < .01$.

Then $P(I|C) \approx 1$, and

$$P(I) = P(I|C)P(C) + P(I|C^c)P(C^c) \approx .01 + (.01)(.99) \approx .02. \tag{5.10}$$

Applying Bayes' Theorem,

$$P(C|I) = P(C)\frac{P(I|C)}{P(I)} \approx (.01)\frac{1}{.02} \approx .5. \tag{5.11}$$

The surprising conclusion is that even with such an apparently accurate test, if someone tests positive for this cancer there is only a 50% chance that he actually has the disease.

Often Bayes' Theorem is stated in a form in which there are several possible causes $C_1, C_2, \ldots$ which might lead to a result I with $P(I) > 0$. If we assume that the observable events $C_1, C_2, \ldots$ form a *partition* of the probability space X, so that they are pairwise disjoint and their union is all of X, then

$$P(I) = P(I|C_1)P(C_1) + P(I|C_2)P(C_2) + ..., \tag{5.12}$$

and Equation (5.8) says that for each i,

$$P(C_i|I) = P(C_i)\frac{P(I|C_i)}{P(I|C_1)P(C_1) + P(I|C_2)P(C_2) + ...}. \tag{5.13}$$

This formula applies for any finite number of observable events C_i as well as for a *countably infinite* number of them.

Exercise 5.3.1. Suppose we want to use a set of medical tests to look for the presence of one of two diseases. Denote by S the event that the test gives a positive result and by D_i the event that a patient has disease $i = 1, 2$. Suppose we know the incidences of the two diseases in the population:

$$P(D_1) = .07, \quad P(D_2) = .05, \quad P(D_1 \cap D_2) = .01. \tag{5.14}$$

From studies of many patients over the years it has also been learned that

$$P(S|D_1) = .9, \quad P(S|D_2) = .8, \\ P(S|(D_1 \cup D_2)^c) = .05, \quad P(S|D_1 \cap D_2) = .99. \tag{5.15}$$

(a) Form a partition of the underlying probability space X that will help to analyze this situation.

(b) Find the probability that a patient has disease 1 if the battery of tests turns up positive.

(c) Find the probability that a patient has disease 1 but not disease 2 if the battery of tests turns up positive.

5.4 Bernoulli trials

In Section 5.2 we came across independent repeated trials of an experiment, such as flipping a coin or rolling a die. Such a sequence is conveniently represented by a probability space whose elements are strings on a finite alphabet. Equivalently, if a single run of the experiment is modeled by a probability space $(D, \mathcal{B}, P)$, then n independent repetitions of the experiment are modeled by the Cartesian product of D with itself n times, with the probability measure formed by a product of P with itself n times.

We now state this more precisely. Let $(D, \mathcal{B}, P)$ be a probability space with $D = \{0, \dots, d-1\}$, $\mathcal{B} =$ the family of all subsets of D, $P(i) = p_i > 0$ for $i = 0, \dots, d-1$.

Denote by $D^{(n)}$ the Cartesian product of D with itself n times. Thus $D^{(n)}$ consists of all ordered n-tuples $(x_1, \dots, x_n)$ with each $x_i \in D, i = 1, \dots, n$. If we omit the commas and parentheses, we can think of each element of $D^{(n)}$ as a *string of length n on the alphabet D*.

Example 5.4.1. If $D = \{0, 1\}$ and $n = 3$, then

$$D^{(n)} = \{000, 001, 010, 011, 100, 101, 110, 111\},$$

the set of all bit strings of length 3.

We now define the set of observables in $D^{(n)}$ to be $\mathcal{B}^{(n)} =$ the family of all subsets of $D^{(n)}$. The probability measure $P^{(n)}$ on $D^{(n)}$ is determined by

$$P^{(n)}(x_1 x_2 \dots x_n) = P(x_1)P(x_2) \cdots P(x_n) \tag{5.16}$$

for each $x_1 x_2 \dots x_n \in D^{(n)}$.

This definition of $P^{(n)}$ in terms of products of probabilities seen in the different coordinates (or entries) of a string guarantees the independence of two events that are determined by disjoint ranges of coordinates. Note that this holds true even if the strings of length n are not all equally likely.

Exercise 5.4.1. A coin whose probability of heads is p, with $0 < p < 1/2$, is flipped three times. Write out the probabilities of all the possible outcomes. If A is the event that the second flip produces heads, and B is the event that either the first or third flip produces tails, find $P^{(3)}(A \cap B)$ and $P^{(3)}(A)P^{(3)}(B)$.

Let $D = \{0, 1\}$ and $P(0) = p \in (0, 1), P(1) = 1 - p$. Construct as above the probability space $(D^{(n)}, \mathcal{B}^{(n)}, P^{(n)})$ representing n independent repetitions of the experiment $(D, \mathcal{B}, P)$. The *binomial distribution* gives the probability for each $k = 0, 1, \dots, n$ of the set of strings of length n that contain exactly k 0's. recall that $C(n, k)$ denotes the binomial coefficient $n!/(k!(n-k)!)$, the number of k-element subsets of a set with n elements.

Proposition 5.4.1. *Let $(D^{(n)}, \mathcal{B}^{(n)}, P^{(n)})$ be as described above. Then for each $k = 0, 1, \dots, n$,*

$$P^{(n)}\{x_1 \dots x_n \in D^{(n)} : x_i = 0 \text{ for } k \text{ choices of } i = 1, \dots, n\} = \\ C(n, k)p^k(1-p)^{n-k}. \tag{5.17}$$

Proof. For each subset S of $\{1,\dots,n\}$, let

$$E(S) = \{x \in D^{(n)} : x_i = 0 \text{ if and only if } i \in S\}.$$

Note that if S_1 and S_2 are different subsets of $\{1,\dots,n\}$, then $E(S_1)$ and $E(S_2)$ are disjoint.

Fix $k = 0, 1, \dots, n$. There are $C(n,k)$ subsets of $\{1,\dots,n\}$ which have exactly k elements, and for each such subset S we have

$$P^{(n)}(E(S)) = p^k(1-p)^{n-k}.$$

Adding up the probabilities of these disjoint sets gives the result. □

Exercise 5.4.2. For the situation in Exercise 5.4.1 and each $k = 0, 1, 2, 3$, list the elements of $A_k =$ the event that exactly k heads occur. Also calculate the probability of each A_k.

Representing repetitions of an experiment with finitely many possible outcomes by strings on a finite alphabet draws an obvious connection with the modeling of information transfer or acquisition. A single experiment can be viewed as reading a single symbol, which is thought of as the outcome of the experiment. We can imagine strings (or experimental runs) of arbitrary lengths, and in fact even of infinite length. For example, we can consider the space of one-sided infinite bit strings

$$\Omega^+ = \{x_0x_1x_2\cdots : \text{ each } x_i = 0 \text{ or } 1\}, \tag{5.18}$$

as well as the space of two-sided infinite bit strings

$$\Omega = \{\dots x_{-1}x_0x_1\cdots : \text{ each } x_i = 0 \text{ or } 1\}. \tag{5.19}$$

Given p with $0 < p < 1$, we can again define a probability measure for many events in either of these spaces: for example,

$$P_p^{(\infty)}\{x : x_2 = 0, x_6 = 1, x_7 = 1\} = p(1-p)(1-p). \tag{5.20}$$

A set such as the one above, determined by specifying the entries in a finite number of places in a string, is called a *cylinder set*. Let us define the probability of each cylinder set in accord with the idea that 0's and 1's are coming independently, with probabilities p and $1-p$, respectively. Thus, if $0 \le i_1 < i_2 < \cdots < i_r$, each $a_1, \dots, a_r = 0$ or 1, and s of the a_j's are 0, let

$$P_p^{(\infty)}\{x \in \Omega^+ : x_{i_1} = a_1, \dots, x_{i_r} = a_r\} = p^s(1-p)^{r-s}. \tag{5.21}$$

It takes some effort (which we will not expend at this moment) to see that this definition does not lead to any contradictions, and that there is a unique extension of $P_p^{(\infty)}$ so as to be defined on a family $\mathcal{B}^{(\infty)}$ which contains all the cylinder sets and is closed under complementation, countable unions, and countable intersections.

Definition 5.4.1. If D is an arbitrary finite set, we denote by $\Omega^+(D)$ the set of one-sided infinite strings $x_0x_1x_2\dots$ with entries from the alphabet D, and we denote by $\Omega(D)$ the set of two-sided infinite strings with entries from D. We abbreviate $\Omega^+ = \Omega^+(\{0,1\})$ and $\Omega = \Omega(\{0,1\})$.

With each of these sequence spaces we deal always with a fixed family $\mathcal{B}$ of observable events which contains the cylinder sets and is closed under countable unions, countable intersections, and complementation.

The spaces $\Omega^+(D)$ and $\Omega(D)$ are useful models of *information sources*, especially when combined with a family of observables $\mathcal{B}$ which contains all cylinder sets and with a probability measure P defined on $\mathcal{B}$. (We are dropping the extra superscripts on $\mathcal{B}$ and P in order to simplify the notation.) Given a string $a = a_0 \dots a_{r-1}$ on the symbols of the alphabet D and a time $n \ge 0$, the probability that the

source emits the string at time n is given by the probability of the cylinder set $\{x : x_n = a_0, x_1 = a_1, \ldots, x_{n+r-1} = a_{r-1}\}$.

Requiring that countable unions and intersections of observable events be observable allows us to consider quite interesting and complicated events, including various combinations of infinite sequences of events.

Example 5.4.2. In the space Ω^+ constructed above, with the probability measure $P_p^{(\infty)}$, let us see that the set of (one-sided) infinite strings which contain infinitely many 0's has probability 1. For this purpose we assume (as can be proved rigorously) that the probability space $(\Omega^+, \mathcal{B}^{(\infty)}, P_p^{(\infty)})$ does indeed satisfy the properties set out axiomatically at the beginning of these notes. Let

$$A = \{x \in \Omega^+ : x_i = 0 \text{ for infinitely many } i\}.$$

We aim to show that $P^{(\infty)}(A^c) = 0$ ($A^c = \Omega^+ \setminus A$ = the complement of A), and hence $P^{(\infty)}(A) = 1$.

For each $n = 0, 1, 2, \ldots$ let

$$B_n = \{x \in \Omega^+ : x_n = 0 \text{ but } x_i = 1 \text{ for all } i > n\},$$

and let B_{-1} consist of the single string $1111\ldots$. Then the sets B_n are pairwise disjoint and their union is A^c.

By countable additivity,

$$P_p^{(\infty)}(\bigcup_{n=-1}^{\infty} B_n) = \sum_{n=-1}^{\infty} P_p^{(\infty)}(B_n),$$

so it is enough to show that

$$P_p^{(\infty)}(B_n) = 0 \quad \text{for all } n.$$

Fix any $n = -1, 0, 1, 2, \ldots$. For each $r = 1, 2, \ldots,$

$$B_n \subset Z_{n+1,n+r} = \{x \in \Omega^+ : x_{n+1} = x_{n+2} = \cdots = x_{n+r} = 1\},$$

and

$$P_p^{(\infty)}(Z_{n+1,n+r}) = (1-p)^r.$$

Since $0 < 1 - p < 1$, we have $(1-p)^r \to 0$ as $r \to \infty$, so $P_p^{(\infty)}(B_n) = 0$ for each n.

If A is an observable event in any probability space which has probability 1, then we say that A occurs *almost surely*, or *with probability* 1. If some property holds for all points $x \in D$ in a set of probability 1, then we say that the property holds *almost everywhere*.

Exercise 5.4.3. In the probability space $(\Omega^+, \mathcal{B}^{(\infty)}, P_p^{(\infty)})$ constructed above, find the probability of the set of infinite strings of 0's and 1's which never have two 1's in a row. (*Hint:* For each $n = 0, 1, 2, \ldots$ consider $B_n = \{x \in \Omega^+ : x_{2n}x_{2n+1} \neq 11\}$.)

5.5 Markov chains

Symbols in strings or outcomes of repeated experiments are not always completely independent of one another—frequently there are relations, interactions, or dependencies among the entries in various coordinates. In English text, the probabilities of letters depend heavily on letters near them: h is much more likely to follow t than to follow f. Some phenomena can show very long-range order, even infinite memory. Markov chains model processes with only short-range memory, in which the probability of what symbol comes next depends only on a fixed number of the immediately preceding symbols. In the

simplest case, 1-step Markov chains, the probability of what comes next depends only on the immediately preceding symbol. The outcome of any repetition of the experiment depends only on the outcome of the immediately preceding one and not on any before that.

The precise definition of a Markov chain on a finite state space, or alphabet, $D = \{0, 1, \dots, d-1\}$ is as follows. The sample space is the set Σ^+ of all one-sided (could be also two-sided) infinite sequences $x = x_0 x_1 \dots$ with entries from the alphabet D. The family of observable events again contains all the cylinder sets. The probability measure M is determined by two pieces of data:

1. a *probability vector* $p = (p_0, \dots, p_{d-1})$, with each $p_i \geq 0$ and $p_0 + \cdots + p_{d-1} = 1$, giving the *initial distribution* for the chain;

2. a matrix $P = (P_{ij})$ giving the *transition probabilities* between each pair of states $i, j \in D$. It is assumed that each $P_{ij} \geq 0$ and that for each i we have $P_{i1} + P_{i2} + \cdots + P_{i,d-1} = 1$. Such a P is called a *stochastic matrix*.

Now the probability of each basic cylinder set determined by fixing the first n entries at values $a_0, \dots, a_{n-1} \in D$ is defined to be

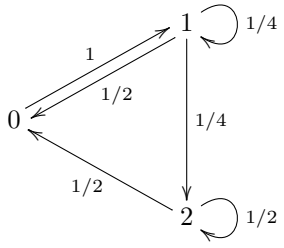
$$M\{x \in \Sigma^+ : x_0 = a_0, \dots, x_{n-1} = a_{n-1}\} = p_{a_0} P_{a_0 a_1} P_{a_1 a_2} \dots P_{a_{n-2,n-1}}. \tag{5.22}$$

The idea here is simple. The initial symbol of a string, at coordinate 0, is selected with probability determined by the initial distribution p: symbol i has probability p_i of appearing, for each $i = 0, 1, \dots, d-1$. Then given that symbol, a_0, the probability of transitioning to any other symbol is determined by the entries in the matrix P, specifically the entries in row a_0: the probability that a_1 comes next, given that we just saw a_0 is $P_{a_0 a_1}$. And so on. The condition that the matrix P have row sums 1 tells us that we are sure to be able to add *some* symbol each time.

The 1-step memory property can be expressed as follows. For any choice of symbols $a_0, \dots, a_n$,

$$M\{x \in \Sigma^+ : x_n = a_n | x_0 = a_0, \dots, x_{n-1} = a_{n-1}\} = M\{x \in \Sigma^+ : x_n = a_n | x_{n-1} = a_{n-1}\}.$$

Finite-state Markov chains are conveniently visualized in terms of random paths on directed graphs.



Here the states are $0, 1, 2$ and the transition probabilities between states are the labels on the arrows. Thus the stochastic transition matrix is

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \end{pmatrix}.$$

If we specified an initial distribution $p = (1/6, 1/2, 1/3)$ listing the initial probabilities of the states $0, 1, 2$, respectively, then the probabilities of strings starting at the initial coordinate would be calculated as in this example:

$$M\{x \in \Sigma^+ : x_0 = 1, x_2 = 1, x_3 = 0\} = p_1 P_{11} P_{10} = \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{16}.$$

Exercise 5.5.1. For the example above, with p and P as given, find the probabilities of all the positive-probability strings of length 3.

Recall that the vector $p = (p_0, \ldots, p_{d-1})$ gives the initial distribution: the probability that at time 0 the system is in state $j \in \{0, \ldots, d-1\}$ is p_j. So what is the probability that the system is in state j at time 1?

Well, the event that the system is in state j at time 1, namely $\{x \in \Sigma^+ : x_1 = j\}$, is the union of d disjoint sets defined by the different possible values of x_0:

$$\{x \in \Sigma^+ : x_1 = j\} = \bigcup_{i=0}^{d-1} \{x \in \Sigma^+ : x_0 = i, x_1 = j\}. \tag{5.23}$$

Since the i'th one of these sets has probability $p_i P_{ij}$, we have

$$M\{x \in \Sigma^+ : x_1 = j\} = \sum_{i=0}^{d-1} p_i P_{ij}. \tag{5.24}$$

So we have determined the distribution $p^{(1)}$ of the chain at time 1. The equations

$$p_j^{(1)} = \sum_{i=0}^{d-1} p_i P_{ij} \quad \text{for } j = 0, \ldots, d-1 \tag{5.25}$$

are abbreviated, using multiplication of vectors by matrices, by

$$p^{(1)} = pP. \tag{5.26}$$

Similarly, the distribution at time 2 is given by

$$p^{(2)} = p^{(1)} P = pP^2, \tag{5.27}$$

where P^2 is the square of the matrix P according to matrix multiplication. And so on: the probability that at any time $n = 0, 1, 2, \ldots$ the chain is in state $j = 0, \ldots, d-1$ is $(pP^n)_j$, namely, the j'th entry of the vector obtained by multiplying the initial distribution vector p on the right n times by the stochastic transition matrix P.

Here's a *quick definition of matrix multiplication.* Suppose that A is a matrix with m rows and n columns ($m, n \geq 1$; if either equals 1, A is a (row or column) vector). Suppose that B is a matrix with n rows and p columns. Then AB is defined as a matrix with m rows and p columns. The entry in the i'th row and j'th column of the product AB is formed by using the i'th row of A and the j'th column of B: take the sum of the products of the entries in the i'th row of A (there are n of them) with the entries in the j'th column of B (there are also n of these)—this is the "dot product" or "scalar product" of the i'th row of A with the j'th column of B:

$$(AB)_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}, \quad \text{for } i = 1, \ldots, m; j = 1, \ldots, p. \tag{5.28}$$

Note that here we have numbered entries starting with 1 rather than with 0. (This is how Matlab usually does it).

Markov chains have many applications in physics, biology, psychology (learning theory), and even sociology. Here is a nonrealistic indication of possible applications.

Exercise 5.5.2. Suppose that a certain study divides women into three groups according to their level of education: completed college, completed high school but not college, or did not complete high school. Suppose that data are accumulated showing that the daughter of a college-educated mother has a probability .7 of also completing college, probability .2 of only making it through high school, and probability .1 of not finishing high school; the daughter of a mother who only finished high school has probabilities .5, .3, and .2, respectively, of finishing college, high school only, or neither; and the daughter of a mother who did not finish high school has corresponding probabilities .3, .4, and .3.

(a) We start with a population in which 30% of women finished college, 50% finished high school but not college, and 20% did not finish high school. What is the probability that a granddaughter of one of these women who never finished high school will make it through college?

(b) Suppose that the initial distribution among the different groups is $(.5857, .2571, .1571)$. What will be the distribution in the next generation? The one after that? The one after that?

Remark 5.5.1. Under some not too stringent hypotheses, the powers P^k of the stochastic transition matrix P of a Markov chain will converge to a matrix Q all of whose rows are equal to the same vector q, which then satisfies $qQ = q$ and is called the *stable distribution* for the Markov chain. You can try this out easily on Matlab by starting with various stochastic matrices P and squaring repeatedly.

5.6 Space mean and time mean

Definition 5.6.1. A *random variable* on a probability space $(X, \mathcal{B}, P)$ is a function $f : X \to \mathbb{R}$ such that for each interval (a, b) of real numbers, the event $\{x \in X : f(x) \in (a, b)\}$ is an observable event. More briefly,

$$f^{-1}(a, b) \in \mathcal{B} \quad \text{for all } a, b \in \mathbb{R}. \tag{5.29}$$

This definition seeks to capture the idea of making measurements on a random system, without getting tangled in talk about numbers fluctuating in unpredictable ways.

Example 5.6.1. In an experiment of rolling two dice, a natural sample space is $X = \{(i, j) : i, j = 1, \ldots, 6\}$. We take $\mathcal{B} =$ the family of all subsets of X and assume that all 36 outcomes are equally likely. One important random variable on this probability space is the sum of the numbers rolled:

$$s(i, j) = i + j \quad \text{for all } (i, j) \in X.$$

Example 5.6.2. If X is the set of bit strings of length 7, $\mathcal{B} =$ all subsets of X, and all strings are equally likely, we could consider the random variable

$$s(x) = x_0 + \cdots + x_6 = \text{ number of 1's in } x.$$

In the following definitions let $(X, \mathcal{B}, P)$ be a probability space.

Definition 5.6.2. A *partition* of X is a family $\{A_1, \ldots, A_n\}$ of observable subsets of X (each $A_i \in \mathcal{B}$) which are pairwise disjoint and whose union is X. The sets A_i are called the *cells* of the partition.

Definition 5.6.3. A *simple random variable* on X is a random variable $f : X \to \mathbb{R}$ for which there is a partition $\{A_1, \ldots, A_n\}$ of X such that f is constant on each cell A_i of the partition: there are $c_1, \ldots, c_n \in \mathbb{R}$ such that $f(x) = c_i$ for all $x \in A_i, i = 1, \ldots, n$.

Definition 5.6.4. Let f be a simple random variable as in Definition 5.6.3. We define the *space mean*, or *expected value*, or *expectation* of f to be

$$\mathbb{E}(f) = \sum_{i=1}^{n} c_i P(A_i). \tag{5.30}$$

Example 5.6.3. Let the probability space and random variable f be as in Example 5.6.1—the sum of the numbers showing. To compute the expected value of $f = s$, we partition the set of outcomes according to the value of the sum: let $A_j = s^{-1}(j), j = 2, \ldots, 12$. Then we figure out the probability of each cell of the partition. Since all outcomes are assumed to be equally likely, the probability that $s(x) = i$ is the number of outcomes x that produce sum i, times the probability (1/36) of each outcome. Now the numbers of ways to roll $2, 3, \ldots, 12$, respectively, are seen by inspection to be $1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1$. Multiply each value (2 through 12) of the random variable s by the probability that it takes that value $(1/36, 2/36, \ldots, 1/36)$ and add these up to get $\mathbb{E}(s) = 7$.

Thus 7 is the expected sum on a roll of a pair of dice. This is the *mean* or *average* sum. The expected value is not always the same as the most probable value (if there is one)—called the *mode*—as the next example shows.

Exercise 5.6.1. Find the expected value of the random variable in Example 5.6.2.

Exercise 5.6.2. Suppose that the bit strings of length 7 in Example 5.6.2 are no longer equally likely but instead are given by the probability measure $P^{(7)}$ on $\{0,1\}^{(7)}$ determined by $P(0) = 1/3, P(1) = 2/3$. Now what is the expected value of the number of 1's in a string chosen at random?

The expectation of a random variable f is its average value over the probability space X, taking into account that f may take values in some intervals with greater probability than in others. If the probability space modeled a game in which an entrant received a payoff of $f(x)$ dollars in case the random outcome were $x \in X$, the expectation $\mathbb{E}(f)$ would be considered a fair price to pay in order to play the game. (Gambling establishments charge a bit more than this, so that they will probably make a profit.)

We consider now a situation in which we make not just a single measurement f on a probability space $(X, \mathcal{B}, P)$ but a *sequence* of measurements $f_1, f_2, f_3, \ldots$. A sequence of random variables is called a *stochastic process*. If the system is in state $x \in X$, then we obtain a sequence of numbers $f_1(x), f_2(x), f_3(x), \ldots$, and we think of $f_i(x)$ as the result of the observation that we make on the system at time $i = 1, 2, 3, \ldots$.

It is natural to form the averages of these measurements:

$$A_n\{f_i\}(x) = \frac{1}{n}\sum_{k=1}^{n} f_k(x) \tag{5.31}$$

is the average of the first n measurements. If we have an *infinite* sequence $f_1, f_2, f_3, \ldots$ of measurements, we can try to see whether these averages settle down around a limiting value

$$A_\infty\{f_i\}(x) = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} f_k(x). \tag{5.32}$$

Such a limiting value may or may not exist—quite possibly the sequence of measurements will be wild and the averages will not converge to any limit.

We may look at the sequence of measurements and time averages in a different way: rather than imagining that we make a sequence of measurements on the system, we may imagine that we make the *same* measurement f on the system each time, but the system changes with time. This is the viewpoint of dynamical systems theory; in a sense the two viewpoints are equivalent.

Example 5.6.4. Consider the system of Bernoulli trials $(\Omega^+, \mathcal{B}^{(\infty)}, P_p^{(\infty)})$ described above: the space consists of one-sided infinite sequences of 0's and 1's, the bits arriving independently with $P(0) = p$ and $P(1) = 1 - p$. We can "read" a sequence in two ways.

(1) For each $i = 0, 1, 2, \ldots$, let $f_i(x) = x_i$. We make a different measurement at each instant, always reading off the bit that is one place more to the right than the previously viewed one.

(2) Define the *shift transformation* $\sigma : \Omega^+ \to \Omega^+$ by $\sigma(x_0x_1x_2\dots) = x_1x_2\dots$. This transformation lops off the first entry in each infinite bit string and shifts the remaining ones one place to the left. For each $i = 1, 2, \dots$, σ^i denotes the composition of σ with itself i times; thus σ^2 lops off the first two places while shifting the sequence two places to the left. On the set Ω of two-sided infinite sequences we can shift in both directions, so we can consider σ^i for $i \in \mathbb{Z}$.

Now let $f(x) = x_0$ for each $x \in \Omega^+$. Then the previous $f_i(x) = f(\sigma^i x)$ for all $i = 0, 1, 2, \dots$. In this realization, we just sit in one place, always observing the first entry in the bit string x as the string streams by toward the left.

This seems to be maybe a more relaxed way to make measurements. Besides that, the dynamical viewpoint has many other advantages. For example, many properties of the stochastic processes $\{f(\sigma^i x)\}$, can be deduced from study of the action of σ on $(\Omega^+, \mathcal{B}^{(\infty)}, P_p^{(\infty)})$ alone, independently of the particular choice of f.

Exercise 5.6.3. In the example $(\Omega^+, \mathcal{B}^{(\infty)}, P_p^{(\infty)})$ just discussed, with $f(x) = x_0$ as above, do you think that the time average

$$A_\infty f(x) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f(\sigma^k x)$$

will exist? (For all x? For most x?) If it were to exist usually, what should it be?

Exercise 5.6.4. Same as the preceding exercise, but with f replaced by

$$f(x) = \begin{cases} 1 & \text{if } x_0x_1 = 01 \\ 0 & \text{otherwise.} \end{cases}$$

5.7 Stationary and ergodic information sources

We have already defined an information source. It consists of the set of one- or two-sided infinite strings $\Omega^+(D)$ or $\Omega(D)$ with entries from a finite alphabet D; a family $\mathcal{B}$ of subsets of the set of strings which contains all the cylinder sets and is closed under complementation and countable unions and intersections; and a probability measure P defined for all sets in $\mathcal{B}$. (For simplicity we continue to delete superscripts on $\mathcal{B}$ and P.) We also have the shift transformation, defined on each of $\Omega^+(D)$ and $\Omega(D)$ by $(\sigma x)_i = x_{i+1}$ for all indices i. If $f(x) = x_0$, then observing $f(\sigma^k x)$ for $k = 0, 1, 2, \dots$ "reads" the sequence $x = x_0x_1x_2\dots$ as σ makes time go by.

Definition 5.7.1. An information source as above is called *stationary* if the probability measure P is shift-invariant: given any word $a = a_0a_2\dots a_{r-1}$ and any two indices n and m in the allowable range of indices ($\mathbb{Z}$ for $\Omega(D)$, $\{0, 1, 2, \dots\}$ for $\Omega^+(D)$),

$$\begin{aligned} &P\{x : x_n = a_0, x_{n+1} = a_1, \dots, x_{n+r-1} = a_{r-1}\} = \\ &P\{x : x_m = a_0, x_{m+1} = a_1, \dots, x_{m+r-1} = a_{r-1}\}. \end{aligned} \tag{5.33}$$

The idea is that a stationary source emits its symbols, and in fact consecutive strings of symbols, according to a probability measure that does not change with time. The probability of seeing a string such as 001 is the same at time 3 as it is at time 3003. Such a source can be thought of as being in an equilibrium state—whatever mechanisms are driving it (which are probably random in some way) are not having their basic principles changing with time.

Example 5.7.1. The Bernoulli sources discussed above are stationary. This is clear from the definition of the probability of the cylinder set determined by any word as the product of the probabilities of the individual symbols in the word.

Example 5.7.2. Consider a Markov source as above determined by an initial distribution p and a stochastic transition matrix P. If p is in fact a stable distribution for P (see Remark 5.5.1),

$$pP = p,$$

then the Markov process, considered as an information source, is stationary.

Definition 5.7.2. A stationary information source as above is called *ergodic* if for every simple random variable f on the set of sequences, the time mean of f almost surely equals the space mean of f. More precisely, the set of sequences x for which

$$A_\infty f(x) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f(\sigma^k x) = \mathbb{E}(f) \tag{5.34}$$

(in the sense that the limit exists and equals $\mathbb{E}(f)$) has probability 1.

In fact, it can be shown that in order to check whether or not a source is ergodic, it is enough to check the definition for random variables f which are the *characteristic functions* of cylinder sets. Given a word $a = a_0a_1a_2\dots a_{r-1}$, define

$$f_a(x) = \begin{cases} 1 & \text{if } x_0x_1\dots x_{r-1} = a \\ 0 & \text{otherwise.} \end{cases} \tag{5.35}$$

Ergodicity is then seen to be equivalent to requiring that *in almost every sequence, every word appears with limiting frequency equal to the probability of any cylinder set defined by that word.* Here "almost every" sequence means a set of sequences which has probability one.

Example 5.7.3. The Bernoulli systems defined above are all ergodic. This is a strong version of Jakob Bernoulli's Law of Large Numbers (1713).

What kinds of sources are *not* ergodic, you ask? It's easiest to give examples if one knows that ergodicity is equivalent to a kind of indecomposability of the probability space of sequences.

Example 5.7.4. Let us consider an information source which puts out one-sided sequences on the alphabet $D = \{0, 1\}$. Let us suppose that the probability measure P governing the outputs is such that with probability $1/2$ we get a constant string of 0's, otherwise we get a string of 0's and 1's coming independently with equal probabilities. If we consider the simple random variable f_0, which gives a value of 1 if $x_0 = 0$ and otherwise gives a value 0, we see that on a set of probability $1/2$ the time mean of f_0 is 1, while on another set of probability $1/2$ it is $1/2$ (assuming the result stated in Example 5.7.3). Thus, no matter the value of $\mathbb{E}(f_0)$, we cannot possibly have $A_\infty f_0 = \mathbb{E}(f_0)$ almost surely.

Exercise 5.7.1. Calculate the space mean of the random variable f_0 in the preceding example.

Exercise 5.7.2. Calculate the space mean and time mean of the random variable f_1 in the preceding example (see Formula (5.35)).

References

Hans Christian von Baeyer, *Information: The New Language of Science*, Phoenix, London, 2004.

Part 6

Number theory and cryptography

6.1 Modular arithmetic

Many interesting and useful properties of the set of integers $\mathbb{Z}$ (the whole numbers $\ldots, -3, -2, -1, 0, 1, 2, 3, 4, 5, \ldots$) can be studied by thinking in terms of divisibility by a particular base or *modulus*, a positive integer m. For example, it is frequently important to know whether a given number k is even or odd, that is to say, whether or not k is (evenly) divisible by 2. (An even number of young swimmers can be paired off in a buddy system, but if there are an odd number, a problem arises.) The entire set $\mathbb{Z}$ of integers breaks into two disjoint subsets, the evens and the odds. Note that two numbers x and y are in the same class if and only if their difference is *even*, that is, 2 divides $x - y$. Further, the sum of any two even numbers is even, the sum of any two odd numbers is even, and the sum of an even number and an odd number is odd. This allows us to do arithmetic with the two classes of even and odd numbers. Similar statements hold for divisibility by any positive integer modulus m, and we now give those statements formally.

Definition 6.1.1. Let m be a positive integer. We say that two integers x and y are *congruent modulo* m, and write $x \equiv y \mod m$, alternatively $x \equiv_m y$, in case m divides $x - y$, which is written as $m|(x-y)$.

Thus $x \equiv y \mod m$ if and only if there is $k \in \mathbb{Z}$ such that $x - y = km$.

The relation $x \equiv_m y$ divides the set of integers $\mathbb{Z}$ into m disjoint *congruence classes*:
$[0]_m$ = all those $x \in \mathbb{Z}$ which are congruent to $0 \mod m$, i.e., all those x which are divisible by m;
$[1]_m$ = all those $x \in \mathbb{Z}$ which are congruent to $1 \mod m$, i.e., all those x which leave a remainder of 1 when divided by m;
.
.
.
$[m-1]_m$ = all those $x \in \mathbb{Z}$ which are congruent to $m-1 \mod m$, i.e., all those x which leave a remainder of $m - 1$ when divided by m.

This set of congruence classes is denoted by $\mathbb{Z}_m$. It has exactly m elements. We will see in a moment that it is possible to do some arithmetic with these elements.

Think of the integers as being written on a long ribbon, which is then wrapped around a circle of circumference m. The numbers that fall on top of (or underneath) 0 constitute the congruence class $[0]_m$, and similarly for the congruence classes of $1, 2, \ldots, m - 1$. Each of the m congruence classes contains infinitely many elements. Any two elements in the same congruence class differ by a multiple of m; and elements of different congruence classes always differ by an amount that is not a multiple of m. The congruence classes are also called *residue classes*, since they concern the remainders, or residues, left after division by m.

To work with $\mathbb{Z}_m$, we can deal either with entire congruence classes or else just pick convenient representatives from each congruence class. A set of integers that contains exactly one member of each congruence class in $\mathbb{Z}_m$ is called a *complete set of representatives modulo* m. A particularly natural

way to do this is to pick the smallest nonnegative member of each class; this gives the complete set of representatives $\{0, 1, \ldots, m-1\}$. If x is any integer, its representative in this set is denoted by $x \mod m$:

$$x \mod m = \text{ that } j \in \{0, 1, \ldots, m-1\} \text{ such that } x \equiv_m j.$$

This is not always the most convenient set of representatives. For example, when $m = 3$ the complete set of representatives $\{-1, 0, 1\}$ is handy because multiplying is really easy.

This brings us up to the topic of arithmetic modulo m.

Proposition 6.1.1. *If $a \equiv_m c$ and $b \equiv_m d$, then $a + b \equiv_m c + d$, $a - b \equiv_m c - d$, and $ab \equiv_m cd$.*

Exercise 6.1.1. Prove this Proposition by using the definition of congruence modulo m.

This Proposition allows us to do addition, subtraction, and multiplication with the elements of $\mathbb{Z}_m$. For example, to compute

$$j + k \mod m,$$

choose any $a \in [j]_m$ (that is, any integer a that is congruent to j modulo m) and any $b \in [k]_m$, and compute $a + b$. The answer is the congruence class modulo m of this result, namely $[a + b]_m$. If we had made different choices of a and b (say we had chosen $c \equiv_m j$ and $d \equiv_m k$), we might have gotten a different result for $j + k$, but the *congruence class* $[j + k]_m$ of the result would have been the same.

These arithmetic operations in $\mathbb{Z}_m$ obey the familiar laws of arithmetic. Addition and multiplication are associative ($(x(yz) = (xy)z$) and commutative ($xy = yx$). There is an identity element, $[0]_m$ for addition modulo m. Every element of $\mathbb{Z}_m$ has an additive inverse: $[a]_m + [-a]_m = [0]_m$. And multiplication distributes over addition: $x(y + z) = xy + yz$. Any set with operations satisfying these conditions is called a *ring*.

$\mathbb{Z}_m$ also has an identity element, $[1]_m$, for multiplication. What about division? Can we find multiplicative inverses of nonzero elements?

There is a bit of a problem here. Let's consider the case $m = 6$. If $a = [5]_6$, then we can indeed find a multiplicative inverse for a: $a \cdot [5]_6 = [5]_6[5]_6 = [5 \cdot 5]_6 = [25]_6 = [1]_6$, so that a is its own multiplicative inverse! But looking at $b = [2]_6$, when we multiply in turn by all the possible candidates, namely the classes of 0,1,2,3,4,5, we get $[0]_6, [2]_6, [4]_6, [0]_6, [2]_6, [4]_6$, and none of these equals $[1]_6$!

We can tell that there is trouble because $[2]_6[3]_6 = [0]_6$, so that neither $[2]_6$ nor $[3]_6$ can have a multiplicative inverse modulo 6. If $a, b \in \mathbb{Z}_m$ satisfy $ab = [0]_m$, and if, say, a has a multiplicative inverse x modulo m, then $[0]_m = x[0]_m = x(ab) = (xa)b = 1 \cdot b = b$. Thus if two nonzero elements of $\mathbb{Z}_m$ have product equal to $[0]_m$, then neither of them can have a multiplicative inverse in $\mathbb{Z}_m$.

6.2 Common divisors and the Euclidean Algorithm

The previous discussion showed that it can be important to know whether or not two positive integers have a common divisor. The *greatest common divisor* of two positive integers m and n is defined already by its name; it is denoted by $\gcd(m, n)$ or occasionally and more briefly, when no confusion is likely because of the context, simply by (m, n). We say that m and n are *relatively prime* in case $\gcd(m, n) = 1$. Two integers have no common prime divisors if and only if they are relatively prime. Isn't it clear that if n_1 and m are relatively prime, and $n_1 \equiv_m n_2$, then n_2 and m are relatively prime? Thus we can talk about *congruence classes relatively prime to m*. We will see below that the set of congruence classes relatively prime to a fixed modulus m forms an interesting and important set; already the number of elements that it contains deserves attention and study.

Definition 6.2.1. For each positive integer $m > 1$, the number of integers $n \in \{0, 1, \ldots, m-1\}$ such that $\gcd(n, m) = 1$ is denoted by $\phi(m)$. We also define $\phi(1) = 1$. The function ϕ is called the *Euler ϕ-(or totient) function.*

Exercise 6.2.1. Show that if p is prime, then $\phi(p) = p - 1$.

Exercise 6.2.2. Show that if p and q are distinct primes, then $\phi(pq) = (p-1)(q-1)$.

Recall that every positive integer m has a unique factorization

$$m = p_1^{e_1} \cdots p_r^{e_r}, \tag{6.1}$$

where $r \geq 1$, $p_1, \ldots, p_r$ are prime numbers with $p_i < p_j$ whenever $1 \leq i < j \leq r$, and $e_1, \ldots, e_r$ are positive integers. In fact, to prove this statement one usually uses the Euclidean Algorithm, which we are about to discuss (so our presentation is not in strict logical order).

The greatest common divisor of m and n can be read off from looking at their prime factorizations: for each prime that appears in both factorizations, take the lowest of the two powers to which it appears, then multiply together all these prime powers to arrive at $d = \gcd(m, n)$. But usually we do not know the prime factorizations of numbers that we come across, and it is a hard problem to determine the prime factorizations of numbers. Indeed, it can be very hard to tell whether a number is prime. (The apparent difficulty of primality testing and prime factorization is at the base of modern asymmetric, public-key cryptographic systems.) Therefore it is important to have a straightforward method for calculating greatest common divisors, and this is what the Euclidean Algorithm provides. We will see that the information accumulated while running the algorithm is also useful for calculating multiplicative inverses in $\mathbb{Z}_m$.

The Euclidean Algorithm for finding $d = \gcd(m, n)$ just consists of repeated division, keeping track of the remainders. The last nonzero remainder is the sought-for d. Let us suppose that $m < n$. We put $r_0 = n$ and $r_1 = m$. Now divide r_0 by r_1, getting a remainder r_2 with $0 \leq r_2 < r_1$:

$$r_0 = k_1 r_1 + r_2. \tag{6.2}$$

Now divide r_1 by r_2, getting a remainder r_3 with $0 \leq r_3 < r_2$, and continue this process until arriving at a remainder of 0:

$$\begin{aligned} r_0 &= k_1 r_1 + r_2 \\ r_1 &= k_2 r_2 + r_3 \\ &\vdots \\ r_{i-1} &= k_i r_i + r_{i+1} \\ r_i &= k_{i+1} r_{i+1} + 0. \end{aligned} \tag{6.3}$$

Because the remainders are all nonnegative and they are decreasing, each smaller than the next, eventually one has to equal 0. The one just before this, the last nonzero remainder, is $r_{i+1} = d = \gcd(m, n)$.

Example 6.2.1. Let us perform this algorithm to find the greatest common divisor of $m = 24$ and $n = 128$:

$$\begin{aligned} 128 &= 5 \cdot 24 + 8 \\ 24 &= 3 \cdot 8 + 0 \\ \gcd(24, 128) &= 8. \end{aligned} \tag{6.4}$$

Check by looking at the prime factorizations: $24 = 2^3 \cdot 3, 128 = 2^7$.

How do we see that the Euclidean Algorithm does, as claimed, produce the greatest common divisor of r_0 and r_1? The key is to notice that any integer k that divides both r_0 and r_1 also divides r_2: this is because $k|a$ and $k|b$ implies $k|(a - b)$. Then by the same reasoning any such k must also divide r_3, and so on, down to $d = r_{i+1}$. So any common divisor of r_0 and r_1 also divides r_{i+1}. Similarly, the equations

show that r_{i+1} divides r_i, hence also r_{i-1}, etc., until we see that r_{i+1} divides both r_0 and r_1. This shows that indeed $r_{i+1} = \gcd(m, n)$.

The following fact may seem strange at first, but it turns out to be very useful for computing multiplicative inverses modulo m. It says that if m and n are positive integers, and we lay off all of their integer multiples on the number line, then it is possible to find two of them at distance $\gcd(m, n)$ from one another.

Proposition 6.2.1. *If m and n are positive integers and $d = \gcd(m, n)$, then there are integers x and y such that*

$$d = xm - yn. \tag{6.5}$$

Remark 6.2.1. Replacing y by $-y$, this is the same as being able to find integers x and y such that

$$d = xm + yn. \tag{6.6}$$

Proof. This is accomplished by reading backwards the sequence of steps in the Euclidean Algorithm for finding $\gcd(m, n) = r_{i+1}$. We have

$$d = r_{i+1} = r_{i-1} - k_i r_i. \tag{6.7}$$

Plugging in

$$r_i = r_{i-2} - k_{i-1} r_{i-1}, \tag{6.8}$$

we find

$$\begin{aligned} r_{i+1} &= r_{i-1} - k_i(r_{i-2} - k_{i-1} r_{i-1}) \\ &= r_{i-1}(1 + k_i k_{i-1} - k_1 r_{i-2}). \end{aligned} \tag{6.9}$$

Continuing in this way, we arrive eventually at

$$r_{i+1} = x r_0 + y r_1, \tag{6.10}$$

as required. □

Example 6.2.2. Looking at the preceding example with $m = 24$ and $n = 128$,

$$8 = -5 \cdot 24 + 128. \tag{6.11}$$

Of course this was a bit easy, involving only one step.

Exercise 6.2.3. (a) Use the Euclidean Algorithm to find $\gcd(792, 2145)$.
(b) Find the prime factorizations of 792 and 2145 and use this information to check your answer to part (a).
(c) Use the reverse of the Euclidean Algorithm to find integers x and y such that $\gcd(792, 2145) = 792x + 2145y$.

6.3 Finding modular multiplicative inverses

Continue to denote by m a fixed positive integer. First let us note that if n is a positive integer which is not relatively prime to m, so that $d = \gcd(m, n) > 1$, then it is *not* possible to find a multiplicative inverse for n modulo m. For if we can find some integer x with

$$xn \equiv_m 1, \tag{6.12}$$

this means that there is an integer k such that

$$xn - 1 = km, \tag{6.13}$$

so that

$$xn - km = 1. \tag{6.14}$$

Now d divides both terms on the left side of this equation, so d divides 1, hence $d = 1$.

On the other hand, if n is relatively prime to m, then n *does* have a multiplicative inverse modulo m. Use the Euclidean Algorithmn reversed to find x and y such that

$$xm + yn = 1. \tag{6.15}$$

Then

$$yn - 1 = -xm, \tag{6.16}$$

so that $yn \equiv_m 1$. Thus $[y]_m$ is the multiplicative inverse of $[n]_m$ in $\mathbb{Z}_m$.

Here is another way to find modular multiplicative inverses—the *power method*. Let us assume that $\gcd(n, m) = 1$, so that we know in advance that n has a multiplicative inverse modulo m, and all we have to do is identify which congruence class of $\mathbb{Z}_m$ it is. Let us look at the powers of n modulo m, namely

$$[n]_m, [n^2]_m, \ldots, [n^j]_m, \ldots. \tag{6.17}$$

Since there are only m congruence classes in $\mathbb{Z}_m$ and this list is infinite, there have to be repeats. In fact, I claim that before there is a repeat there is a first power $i \geq 1$ such that $n^i \equiv_m 1$. For consider the first repeat in this list, say $[n^{j+i}]_m = [n^j]_m$ with $i, j \geq 1$. Since n has a multiplicative inverse $[y]_m$ modulo m, we can multiply by it j times and obtain $n^i \equiv_m 1$. If $i = 2$, then $n^2 \equiv_m 1$, and n is its own multiplicative inverse modulo m. This can happen! For example, 5 is its own multiplicative inverse modulo 4. If $i > 1$, then $[n^{i-1}]_m$ is the multiplicative inverse of n modulo m. If $i = 1$, then $n \equiv_m 1$ and n is its own multiplicative inverse modulo m.

Exercise 6.3.1. Use both the reverse Euclidean Algorithm and the power method to find the modular inverse of 9 modulo 38.

You can use a spreadsheet (or a calculator or other software, such as Matlab or Mathematica) to perform the Euclidean Algorithm and to calculate even very long lists of powers modulo m.

Exercise 6.3.2. Set up a spreadsheet for the Euclidean Algorithm as follows. (a) in the first row, in cells C1 and D1, place the labels $r0$ and $r1$. In cell E1, place the label int(r0/r1). This will be the integer part of $r0/r1$, that is, the number of times that $r0$ "goes evenly"' into $r1$.In cell F1, place the label Rem= $c - de$. Here we will put $r0 - r1\text{int}(r0/r1)$, which is the remainder after dividing $r0$ by $r1$.

(b) This was just setting up the labels. Now in C3 put 38, in D3 put 9, in E3 put =int(C3/D3), and in F3 put =C3-D3*E3:

C	D	E	F
r0	r1	int(r0/r1)	Rem=c-de
38	9	4	2

(c) Now we want to get this to repeat. In C4 put =D3, and in D4 put =F3. This is the important recursive step in which we replace $r0, r1$ by $r1, r2$. Put the mouse pointer on the small black square in the lower right corner of cell C4 and drag it down several rows. Do the same with each of D4–F4. The last nonzero (and non-error) entry in column F should be $\gcd(r0, r1)$.

(d) Change the choice of $r0$ and $r1$ and see whether you still get the right gcd.

Exercise 6.3.3. Set up a spreadsheet for powers modulo m as follows. In cells C1–F1 put the labels n, m, n^j, Mod(n^j, m), respectively. In C3 put 9, in D3 put 38, in E3 put =C3, In C4 put =C3, in D4 put =D3, in E4 put =C3*E3, and in F3 put =Mod(E3,D3). Think about what this means! Then drag the black square in the lower right of C4 downwards a ways, similarly for D4, E4, and F3. The entry in column F above 1 should be the inverse of n modulo m. (You can keep track of the power involved if you like by putting label j in G1, entering 1 in G3, = 1+G3 in G4, and then dragging down the black square in G4.)

You may experiment around with the spreadsheet in the preceding example (and the one before). Notice that if the numbers involved get too big, the capacity of the software is exceeded and the computer refuses to give a numerical answer. Mathematica and Matlab do much better at handling large numbers than does the average spreadsheet, but eventually the capacity of the computer or our patience will be exceeded. However, there is a very nice aspect of arithmetic modulo m: *we never have to deal with numbers larger than* $m - 1$: just keep reducing modulo m at every stage. The size of numbers involved can be reduced even more by using a complete set of representatives modulo m that includes negative numbers.

Exercise 6.3.4. Modify the spreadsheet in the preceding exercise so that all calculations are done modulo m. Then use it to find the multiplicative inverse of 33 modulo 23.

Later on we will explore a bit more the algebraic nature of $\mathbb{Z}_m$. While thinking about the set of powers of n modulo m, in the case that $\gcd(n, m) = 1$, we may note the following properties of this set:

Closure: The product of two elements of the set is also in the set;

Identity: The (multiplicative) identity element $[1]_m$ is in the set;

Inverses: each element of the set has a multiplicative inverse in the set.

Together with the fact that multiplication in $\mathbb{Z}_m$ is associative ($a(bc) = (ab)c$ for all $a, b, c \in \mathbb{Z}_m$), these properties say that if $\gcd(n, m) = 1$, then the set of powers of n modulo m form a *group* with the operation of multiplication modulo m. Since multiplication is also commutative ($ab = ba$ for all $a, b \in \mathbb{Z}_m$), we say that they form a *commutative* or *abelian* group.

Exercise 6.3.5. Use the modular powers spreadsheet developed previously to study the set of powers of n modulo m in several cases where n and m are not relatively prime. What structure, if any, do these sets have? What structure, if any, does $\mathbb{Z}_m$ have in terms of these sets?

6.4 The RSA public-key cryptosystem

This brilliant and important cryptographic system was invented in 1977 by Ronald Rivest, Adi Shamir, and Leonard Adelman and discovered independently in 1973, but not announced, by Clifford Cocks and James Ellis of the British Government Communications Headquarters, the successor of Bletchley Park [**?**, p. 279 ff.]. The system has a strange but extremely useful asymmetry property: there are *two* keys involved: each recipient of messages has both a *public key* and *private key*. The recipient announces his public key to everyone but keeps the private key secret. Anyone can use the public key to encode messages for a particular recipient, but, curiously, only someone with knowledge of the private key can decode them. How is this possible? Why does the system work? How is it used in practice? What other applications and implications are there of the existence of such asymmetric systems?

Asymmetric cryptographic systems are based on *one-way* functions—functions whose outputs can be computed in a reasonable amount of time but whose inverses are inordinately difficult and time-consuming to compute (given the output, or result of applying the function, it is essentially impossible to determine what the input was). It is believed that prime multiplication and modular exponentiation—$f(x) = b^x \mod m$—have this property: factorization into primes and modular logarithms are difficult. A *trapdoor function* is a one-way function whose inverse *can* be feasibly computed if one is given an extra

piece of information. Cryptographic systems based on trapdoor functions are now used in real-world communication by governments, businesses, and individuals, notably for secure transactions over the internet.

Even the application of the RSA system takes a lot of computing power and time, because in order for the one-way property to take real effect, very large numbers must be used as keys. Thus this system is typically used to agree on a key for a standard, symmetric, single-key system, such as the Advanced Encryption Standard, which replaced the Data Encryption Standard.

Here is the algorithm for RSA encoding and decoding. We will follow up with examples, plus an explanation of why the system works.

1. The recipient's *public key* consists of the product n of two distinct large primes p and q and an *encoding exponent* e which is *relatively prime* to $m = \phi(n) = (p-1)(q-1)$.

2. The recipient's *private key* consists of the two primes p and q and of the *decoding exponent* d, which is the multiplicative inverse of e modulo $m = \phi(n)$: $de \equiv_m 1$. Someone who knows n and e but not m will have a great deal of trouble finding d. To know m, one should know the factors p and q of n. The factorization of n is the extra piece of information that provides the secret "trapdoor" entrance to the inverse of the presumed one-way function that accomplishes the encoding.

3. Each message to be sent is an integer $M \in \{0, 1, \ldots, n-1\}$. In practice, any text to be sent is first converted to a string of numbers, for example by assigning the numerical ASCII codes that correspond to ordinary keyboard characters. These are then written in binary (base 2) notation, so that the text becomes a string of 0's and 1's. Then a preliminary encoding may be applied, followed by encoding for error correction, for example by adding check digits. Then the text can be cut into blocks of length no more than $\log_2 n$, so that each 0, 1-block represents a unique integer between 0 and $n-1$.

4. The message is converted to $R = M^e \mod n$. This is the message transmitted to the recipient. Note that all senders use the same encoding method for a particular recipient. They can see each other's encoded messages, but no one can decode them except the intended recipient.

5. The recipient receives R and applies to it his secret decoding exponent d, computing, as we will later see,

$$R^d \equiv_n (M^e)^d = M^{ed} \equiv_n M. \tag{6.18}$$

A *modification of the RSA algorithm* allows one to replace $m = \phi(n) = (p-1)(q-1)$ in the preceding steps by $m = \operatorname{lcm}(p-1, q-1)$ (the *least common multiple* of $p-1$ and $q-1$). We will verify this later.

It is believed that, knowing the public key n and e, it would take too much computing power and time to find the decoding exponent d, without knowledge of $m = \phi(n) = (p-1)(q-1)$, for which knowledge of the factorization $n = pq$ would be necessary. It is on this belief that the one-way property of the coding algorithm is based.

Example 6.4.1. Let's see how this works with two small primes, $p = 5$ and $q = 11$. We have $n = pq = 55$ and $m = \phi(n) = (p-1)(q-1) = 4 \cdot 10 = 40$. We need an encoding exponent e relatively prime to $40 = 2^3 \cdot 5$; let's say we choose $e = 7$. Now for our message $M \in \{0, 1, \cdots, n-1\}$, say $M = 13$.

To encode, we compute $R = M^e \mod n = (13)^7 \mod 55$. This can be done by spreadsheet, by calculator, or even pencil and paper—remembering that we need only deal with numbers in $\{0, 1, \cdots, 54\}$ and that the power 7 can be approached rapidly by repeated squaring:

$$\begin{aligned} (13)^2 &= 169 \equiv_{55} 4 \\ (13)^4 &\equiv_{55} 4^2 = 16 \\ (13)^6 &\equiv_{55} 4 \cdot 16 = 64 \equiv_{55} 9 \\ (13)^7 &\equiv_{55} 9 \cdot 13 = 117 \equiv_{55} 7. \end{aligned} \tag{6.19}$$

So the encoded message is $R = 7$.

The recipient needs the decoding exponent d, which is the multiplicative inverse modulo $m = 40$ of $e = 7$. This can be found by the power method, the Euclidean algorithm reversed, or guessing. Taking powers of 7 modulo 40, perhaps with the help of the spreadsheet in Exercise 6.3.4, we find successively

$$7^1 = 7, 7^2 = 49 \equiv_{40} 9, 7^3 \equiv_{40} 63 \equiv_{40} 23, 7^4 \equiv_{40} 7 \cdot 23 \equiv_{40} 161 \equiv_{40} 1, \tag{6.20}$$

so that $d \equiv_{40} 7^3 \equiv_{40} 23$.

The recipient decodes by calculating $R^d = 7^{23}$. Again the spreadsheet in Exercise 6.3.4 will do this very quickly and easily. With paper and pencil we might calculate this way:

$$\begin{gathered} 7^2 = 49 \equiv_{55} -6 \\ 7^4 \equiv_{55} 36 \\ 7^8 \equiv_{55} (36)^2 = 1296 \equiv_{55} -24 \\ 7^{16} \equiv_{55} (-24)^2 = 576 \equiv_{55} 26 \\ 7^{23} \equiv_{55} 7^{16+4+2+1} \equiv_{55} (26)(36)(-6)(7) \equiv_{55} 1 \cdot (-42) \equiv_{55} 13 = M. \end{gathered} \tag{6.21}$$

Example 6.4.2. What if the message M happens not to be relatively prime to the modulus n? Let's try it in the same system as above, but with, for example, $M = 15$ instead of 13.

We find $R = (15)^7 \mod 55 = 5$, and $R^d = 5^{23} \equiv_{55} 15$ (by any of the methods suggested above), so the method seems to work in this case too. We will investigate all of this below.

Exercise 6.4.1. Set up a spreadsheet to perform RSA encoding and decoding as follows. The first column shows what to type in column C, the second what to type in column E, starting with row 2:

*p	5
*q	11
n=pq	=E2*E3
phi(n)=(p-1)(q-1)	=(E2-1)*(E3-1)
*e rel prime to phi(n)	7
*d with de=1 mod phi(n)	23
*message M in Z_ n	13
M^e mod n = R	=mod(E9^E6,E4)
R^d mod n	=mod(E10^E7,E4)

The entries marked by * indicate where the user will have to supply input in those rows in column E; the spreadsheet is supposed to calculate the rest. The computer may fizzle when numbers get too large, like perhaps 7^{23}. The decoding exponent d can be found by means of the spreadsheet in Exercise 6.3.4.

Exercise 6.4.2. Let's extend the spreadsheet of the preceding exercise so that it performs the modified RSA algorithm, using $m = \text{lcm}(p-1, q-1)$ in place of $m = \phi(n) = (p-1)(q-1)$. Starting in row 14, enter

*gcd(p-1,q-1)	2
m=lcm(p-1,q-1)	=E5/E14
*f rel prime to m	7
*g with gf=1 mod m	3
message M	13
M^f mod n = R	=mod(E19^E16,E4)
R^g mod n	=mod(E20^E17,E4)

Note that g is easier to find than d was (because m is now smaller), and the computer also has an easier time accomplishing the coding.

Exercise 6.4.3. Explain why $\text{lcm}(a,b) = (ab)/\gcd(a,b)$. (This fact is used in the preceding spreadsheet.)

Exercise 6.4.4. Try to modify the preceding spreadsheet so that the computer will not be stymied by excessively large numbers. One idea could be to do the arithmetic modulo n when computing powers. For this purpose, see the spreadsheet in Exercise 6.3.4, and consider using the spreadsheet functions *index* or *lookup*, applied to vectors or arrays.

6.5 The mathematics behind the RSA cryptographic system

The key to the functioning of the RSA algorithm is the important but eay-to-understand Euler-Fermat Theorem. We proceed to develop the small amount of number theory needed to prove this theorem.

Fix a positive integer m. We want to see that the set of congruence classes relatively prime to m forms a *commutative group* G_m with respect to multiplication. Recall that if $\gcd(n_1, m) = 1$, and $n_2 \equiv_m n_1$, then also $\gcd(n_2, m) = 1$. Thus we are justified in using the terminology "congruence classes relatively prime to m". We need to check that products of elements of G_m are again in G_m, that G_m contains an identity element for multiplication, that every element of G_m has a multiplicative inverse which is in G_m, and that multiplication is commutative (taking the product of two elements in either order gives the same result).

First, recall that we know that the identity element $[1]_m$ is in G_m and that each element of $x \in G_m$ has a multiplicative inverse $y \in \mathbb{Z}_m$. We must have $y \in G_m$, because we know that y has a multiplicative inverse (x), and we have seen above that an element of $\mathbb{Z}_m$ has a multiplicative inverse if and only if it is relative prime to m, i.e., if and only if it is in G_m.

We have not yet verified that G_m is *closed under multiplication*: $x, y \in G_m$ implies $xy \in G_m$. This can be seen with the help of the following very basic and important fact about prime numbers and divisibility.

Proposition 6.5.1. *If p is prime and m, n are positive integers such that $p|(mn)$, then either $p|m$ or $p|n$.*

Proof. This well-known fact seems to be clearly true, and it is an immediate consequence of the prime factorization property of integers—but the most logical among us would wonder how the prime factorization property is proved. Maybe with the help of this Proposition? The Proposition can be proved directly with the help of the very handy application of the Euclidean Algorithm, Proposition 6.2.1.

So suppose that $p|(mn)$ and p does not divide m. Since p has no divisors besides itself and 1, we have $\gcd(p, m) = 1$. Now using Proposition 6.2.1, find x and y such that $xp + ym = 1$. Multiplying through by n gives $n = nxp + ymn$. Now p divides both terms on the right side of the equation, so $p|n$. □

Corollary 6.5.1. *If $m|(n_1n_2)$ and $\gcd(n_1, m) = 1$, then $m|n_2$.*

Proof. We just pick up the last part of the preceding proof. Since $\gcd(n_1, m) = 1$, we can find $x, y \in \mathbb{Z}$ with $xn_1 + ym = 1$. Then $n_2 = xn_1n_2 + ymn_2$ is seen to be divisible by m. □

Corollary 6.5.2. *If $\gcd(n_1, m) = \gcd(n_2, m) = 1$, then $\gcd(n_1n_2, m) = 1$. The converse is also true ($\gcd(n_1n_2, m) = 1$ implies $\gcd(n_1, m) = \gcd(n_2, m) = 1$.*

Proof. We can argue that if $\gcd(n_1n_2, m) \neq 1$, then n_1n_2 and m have in common an integer divisor $d > 1$ and hence have in common a prime divisor. This requires the extra observation that every positive integer greater than 1 is either prime or else has a prime divisor. How to prove this? We could

consider the *smallest* integer $d > 1$ which is not prime and has no prime divisor. But then d must have some divisor d_0 with $1 < d_0 < d$, and d_0 can't have any prime divisor, because if it did, so would d.

Okay, if we don't like that argument, we can fall back on Propostition 6.2.1. Find x_1 and y_1 with $1 = x_1n_1 + y_1m$ and x_2 and y_2 with $1 = x_2n_2 + y_2m$. Multiply out and regroup, to see that any d that divides both m and n_1n_2 also divides 1 and hence must equal 1.

For the converse, note that if, for example, n_1 and m have a common divisor $d > 1$, then n_1n_2 and m share the same common divisor. □

Recall that the number of elements in G_m is given by Euler's ϕ-function, $\phi(m)$. Recall also that we already know (from preceding exercises) that if p is prime, then $\phi(p) = p - 1$, and if q is a different prime, then $\phi(pq) = (p-1)(q-1)$. Because of its importance for the RSA algorithm, and indeed in the theory of numbers in general, we want to note the following two facts, which will allow us to calculate $\phi(m)$ for all positive integers m.

Proposition 6.5.2. *If p is prime and r is a positive integer, then $\phi(p^r) = p^r - p^{r-1}$.*

Proof. Among the complete set of representatives $\{1, 2, \ldots, p^r\}$ modulo p^r, the only numbers which are not relatively prime to p are the ones that are divisible by p (because p is prime). We can list these as

$$p, 2p, 3p, \ldots, p^{r-1}p, \tag{6.22}$$

which shows that there are exactly p^{r-1} of them. □

Theorem 6.5.1. *The Euler ϕ-function is* multiplicative *in the following sense: if m and n are positive integers such that $\gcd(m, n) = 1$, then $\phi(mn) = \phi(m)\phi(n)$.*

Proof. We give two proofs, or, perhaps, one proof with a visualization. Consider all congruence classes $[xm + yn]_{mn}$ for $x \in G_n$ and $y \in G_m$. There are $\phi(m)\phi(n)$ choices of the pair (x, y). We want to show that they all give different congruence classes in $\mathbb{Z}_{mn}$, and that every congruence class relatively prime to mn comes up in this way.

First we check that all such $xm + yn$ are in fact relatively prime to mn. If p is a prime that divides mn, then p must also divide m or n. Let's suppose that $p|m$. If also $p|(xm + yn)$, then (taking the difference) $p|(yn)$. Now p cannot divide n, since m and n are relatively prime, and p is known already to divide m. Therefore $p|y$. But this is also impossible since y and m are relatively prime. A similar argument applies, of course, in case it is n and not m that p divides.

For the second point, suppose that $x_1m + y_1n \equiv_{mn} x_2m + y_2n$, with $1 = \gcd(x_1, n) = \gcd(x_2, n) = \gcd(y_1, m) = \gcd(y_2, m)$. Then

$$\begin{aligned} &x_1m + y_1n \equiv x_2m + y_2n \mod mn \text{ implies that} \\ &(x_1 - x_2)m + (y_1 - y_2)n \equiv 0 \mod mn, \text{ and hence} \\ &(x_1 - x_2)m = -(y_1 - y_2)n + kmn \text{ for some } k \in \mathbb{Z}. \end{aligned} \tag{6.23}$$

Thus $n|(x_1 - x_2)m$, and, since $\gcd(m, n) = 1$, by Corollary 6.5.1 we have $n|(x_1 - x_2)$. By symmetry of the notation and situation, $m|(y_1 - y_2)$.

These observations imply that $\phi(mn) \geq \phi(m)\phi(n)$. To prove the reverse inequality, we show now that every residue class $\mod mn$ that is relatively prime to mn arises as the class of some $xm + yn$ with x relatively prime to n and y relatively prime to m. For suppose that $k \in \mathbb{Z}$ and $\gcd(k, mn) = 1$. Since $\gcd(m, n) = 1$, by the Euclidean Algorithm (read backwards) there are integers r and s such that $1 = rm + sn$, and hence there are integers x and y such that

$$k = xm + yn.$$

Now y must be relatively prime to m, since otherwise this equation shows that k and m have a common prime factor, contradicting the fact that k and mn are relatively prime. Similarly, x is relatively prime to n. This concludes the first proof.

We are using here Corollary 6.5.2, according to which a number is relatively prime to a product mn if and only if it is relatively prime to each of the factors, m and n. This is also the basis of the following more visual proof of the fact that $\gcd(m, n) = 1$ implies $\phi(mn) = \phi(m)\phi(n)$.

Let us lay out the standard complete set of representatives modulo mn in n rows and m columns as follows:

$$\begin{array}{rrrcr} 0 & 1 & 2 & \dots & m-1 \\ m & m+1 & m+2 & \dots & 2m-1 \\ 2m & 2m+1 & 2m+2 & \dots & 3m-1 \\ \vdots & & & & \\ (n-1)m & (n-1)m+1 & (n-1)m+2 & \dots & nm-1 \end{array}$$

Because of the foregoing observation, we are interested in the number of entries i in this array that are relatively prime to both m and n.

In each column, all the entries are congruent to one another modulo m. The columns correspond to the the elements of $\mathbb{Z}_m$, the congruence classes modulo m. There are $\phi(m)$ columns whose entries are all relatively prime to m. We now restrict our attention to those $\phi(m)$ columns.

In any one of these columns relatively prime to m, how many entries are there that are relatively prime to n? Let us note that each column contains n entries, no two of which are congruent modulo n: If $0 \leq j, k \leq n-1$ and $jm + r \equiv_n km + r$ for some r, then $n|(j-k)m$, and, since $\gcd(n, m) = 1$, $n|(j-k)$, so that $j = k$. Thus each column consists of a complete set of representatives modulo n and hence contains $\phi(n)$ elements relatively prime to n.

We conclude that in the array there are exactly $\phi(n)\phi(m)$ elements relatively prime to both m and n, and hence to mn. □

Exercise 6.5.1. (a) Find $\phi(68)$.
(b) Find $\phi(7911)$.

We now advance our understanding of the $\phi(m)$-element abelian group G_m of congruence classes relatively prime to m in preparation for proving the Euler-Fermat Theorem, which is the secret of success of the RSA algorithm.

Proposition 6.5.3. *If $\{r_1, r_2, \dots, r_{\phi(m)}\}$ is a complete set of representatives relatively prime to m modulo m, so that*

$$G_m = \{[r_1]_m, [r_2]_m, \dots, [r_{\phi(m)}]_m\},$$

and $\gcd(a, m) = 1$, then $\{ar_1, ar_2, \dots, ar_{\phi(m)}\}$ is also a complete set of representatives relatively prime to m modulo m.

Proof. We know that all the ar_i are relatively prime to m, because we have shown that G_m is closed under multiplication. Moreover, because G_m, being a group, has multiplicative inverses, all of these congruence classes are distinct:

$$ar_i \equiv_m ar_j \text{ implies } r_i \equiv_m r_j \tag{6.24}$$

(upon multiplying by the multiplicative inverse a^{-1} of a in G_m). Since we have here $\phi(m)$ distinct congruence classes modulo m relatively prime to m, we must be looking at all of G_m, each element represented exactly once. □

Euler-Fermat Theorem. *If m is a positive integer and a is an integer with* $\gcd(a, m) = 1$*, then*

$$a^{\phi(m)} \equiv 1 \mod m. \tag{6.25}$$

Proof. Let $\{r_1, r_2, \ldots, r_{\phi(m)}\}$ be a complete set of representatives for the group G_m of congruence classes modulo m relatively prime to m. Because of the preceding Proposition,

$$(ar_1)(ar_2)\cdots(ar_{\phi(m)}) \equiv_m r_1 r_2 \cdots r_{\phi(m)}, \tag{6.26}$$

since both products contain exactly the same factors, and their order does not affect the value of the product, the group G_m being commutative. If we denote the righthand side by R (an element of the group G_m), this equation says

$$a^{\phi(m)} R \equiv_m R, \tag{6.27}$$

and muliplying by $R^{-1} \in G_m$ gives

$$a^{\phi(m)} \equiv_m 1. \tag{6.28}$$

□

Fermat's Little Theorem. *If p is prime and a is an integer not divisible by p, then*

$$a^{p-1} \equiv 1 \mod p. \tag{6.29}$$

Corollary 6.5.3. *If p is prime and $a \in \mathbb{Z}$, then*

$$a^p \equiv a \mod p. \tag{6.30}$$

Proof. If $p|a$, then both sides are 0 modulo p. □

Remark 6.5.1. If G is any finite commutative ($ab = ba$ for all $a, b \in G$) group, let I denote the identity element of G and $|G|$ the number of elements in G. Then

$$a^{|G|} = I \quad \text{for all } a \in G. \tag{6.31}$$

The proof is the same as for Proposition 6.5.3 and the Euler-Fermat Theorem.

6.6 Verification of the functioning of the RSA algorithm

1. The recipient's public key consists of the product n of two distinct primes p and q (which are kept secret) and an encoding exponent e which is relatively prime to $m = \phi(n) = (p-1)(q-1)$. A message (or piece of a message) is a congruence class $M \in \mathbb{Z}_n$. We deal first with the case when M is relatively prime to n. (This covers $(p-1)(q-1)$ of the pq congruence classes modulo n.) The encoded message is

$$R = M^e \mod n. \tag{6.32}$$

The recipient knows p and q, and hence also m, and so can find the secret decoding exponent d, which is the multiplicative inverse of e modulo m. There is $k \in \mathbb{Z}$ such that $de = km+1$, and so, if $\gcd(M, n) = 1$,

$$R^d \equiv_n M^{ed} = M^{km+1} = M(M^m)^k = M(M^{\phi(n)})^k \equiv_n M(1^k) \equiv_n M, \tag{6.33}$$

by the Euler-Fermat Theorem. The decoding is accomplished correctly!

2. What if $\gcd(M, n) \neq 1$? For this, as well as the improved RSA algorithm, we need an ancient and useful number theory result.

Chinese Remainder Theorem. *Suppose that $m_1, m_2, \ldots, m_r$ are pairwise relatively prime integers, all at least 2.*
(1) Given $d_1, \ldots, d_r \in \mathbb{Z}$, the system of congruences

$$x \equiv d_i \mod m_i, i = 1, \ldots, r \tag{6.34}$$

has an integer solution x with $0 \leq x < m = m_1 m_2 \cdots m_r$.
(2) The solution is unique modulo $m = m_1 m_2 \cdots m_r$: if $x \equiv y \mod m_i$ for each $i = 1, \ldots, r$. Then $x \equiv y \mod m = m_1 m_2 \cdots m_r$.

Proof. The proof of part (1) is Exercise 6.6.1. Statement (2) is credible in view of prime factorization, but we can indicate also a proof based on tools we have been using before. We have

$$x - y = k_1 m_1 \text{ for some } k_1 \in \mathbb{Z}. \tag{6.35}$$

Since $m_2 | (x - y)$ and $\gcd(m_2, m_1) = 1$, necessarily $m_2 | k_1$, and so we may write

$$x - y = k_2 m_2 m_1 \text{ for some } k_2 \in \mathbb{Z}. \tag{6.36}$$

Now m_3 divides $x - y$ but is relatively prime to $m_2 m_1$ (recall Corollary 6.5.2, which was key to proving that ϕ is multiplicative), so $m_3 | k_2$, and so

$$x - y = k_3 m_3 m_2 m_1 \text{ for some } k_2 \in \mathbb{Z}. \tag{6.37}$$

Continuing in this way (or, in an explicitly more rigorous proof, applying the Axiom of Mathematical Induction), we arrive at the asserted conclusion. □

Exercise 6.6.1. Prove part (1) of the Chinese Remainder Theorem. (*Hint:* For each $i = 1, \ldots, r$ let $u_i = m/m_i$ and let v_i be the multiplicative inverse of u_i modulo m_i (explain why this exists). Then try $x = d_1 u_1 v_1 + \cdots + d_r u_r v_r$.)

Now let's see that RSA decoding works even if $\gcd(M, n) \neq 1$. First, if $\gcd(M, p) = 1$, then $M^{p-1} \equiv_p 1$, by Fermat's Little Theorem, so that

$$\begin{aligned} M^m \equiv_p (M^{p-1})^{q-1} \equiv_p 1^{q-1} = 1, \text{ and hence} \\ M^{ed} = M^{km+1} = (M^m)^k M \equiv_p 1^k M \equiv_p M. \end{aligned} \tag{6.38}$$

But this latter congruence holds even if $\gcd(M, p) \neq 1$, since then both sides are 0 modulo p. Similarly,

$$M^{ed} \equiv_q M. \tag{6.39}$$

Since p and q are distinct primes, the Chinese Remainder Theorem applies, and we conclude that

$$M^{ed} \equiv_n M. \tag{6.40}$$

Exercise 6.6.2. In Example 6.4.2, the message $M = 15 \equiv 0 \mod p$, since $p = 5$. Yet the decoding gives us back 15, not 0. Sort through the steps of this calculation in light of the Chinese Remainder Theorem to explain exactly how the decoding works.

3. Finally, we will check that decoding works in the modified RSA algorithm, in which $m = \phi(n) = (p-1)(q-1)$ is replaced by $m = \text{lcm}(p-1, q-1)$. The decoding exponent d is still the multiplicative inverse of the encoding exponent e modulo this new (and probably greatly reduced) m, so we still have $ed = km + 1$ for some $k \in \mathbb{Z}$. Because m is a multiple of $p - 1$, there is $j_1 \in \mathbb{Z}$ such that $m = j_1(p-1)$. If $\gcd(M, p) = 1$, then

$$M^{ed} = M^{km+1} = M(M^{p-1})^{j_1 k} \equiv_p M(1^{j_1 k}) \equiv_p M, \tag{6.41}$$

by Fermat's Little Theorem. As before, the congruence

$$M^{ed} \equiv_p M \tag{6.42}$$

holds true whether or not $\gcd(M, p) = 1$. Similarly,

$$M^{ed} \equiv_q M, \tag{6.43}$$

and then

$$M^{ed} \equiv_n M \tag{6.44}$$

follows from the Chinese Remainder Theorem.

6.7 A few applications

6.7.1 Digital signature

In electronic communication there is inherent anonymity: we cannot see or directly hear the people with whom we are exchanging information. When sensitive information is involved, or commands are sent, or money is moved around, it is important to know that the person on the other end of the communication line has the necessary authority. This is the problem of *authentication*. For 2002 tax returns submitted electronically, the IRS used an electronic signature consisting of birthdate and a self-selected 5-digit identification number. An asymmetric cryptographic system, such as the RSA system, allows for a clever and much surer form of authentication.

The sender who wishes to adduce proof of his identity prepares a message S consisting of his name, plus the date and time of transmission. (This will prevent illicit reuse of the signature by anyone who might intercept it.) He then encodes this message using his personal secret *decoding exponent*, d:

$$R = S^d \mod n.$$

(The RSA encoding setup involving $p, q, n = pq, d$, and e is as before.) This signature R can be sent as an addendum to any other message, M. The combined message and signature can be encoded using a recipient's public key and then sent to that recipient.

When the recipient applies his private decoding key to what he has received, he obtains MR—a legible message M followed by a random-looking stream of bits, R. Now the receiver applies the sender's *public encoding key*, the exponent e, to R, obtaining

$$R^e \equiv_n S^{de} \equiv_n S^{k\phi(m)+1} \equiv_n S,$$

the original, legible, signature S.

The recipient is sure that only someone with knowledge of the sender's secret decoding exponent d could have encoded S to R so as to produce this result and is therefore convinced that the sender is indeed who he says he is.

6.7.2 Diffie-Hellman key exchange

The problem of key exchange has long been one of the basic issues in cryptology. How can two parties wishing to communicate agree securely and conveniently on a key that will be used to encode (and decode) messages between them? In practice, keys must be changed often in order to foil cryptanalysis. Weakness in key selection played a major role in the Polish and British attacks on the German Enigma cipher in World War II.

For our purposes, let us say that the problem is for two people A and B to agree on an element k of $\mathbb{Z}_m$. A clever way to use one-way functions to accomplish this task, even while communicating over an

insecure channel, was discovered by Whitfield Diffie and Martin Hellman, predating the discovery of the RSA system. It is based on the presumed one-way function, for a fixed base $r \in \mathbb{Z}_m$,

$$f(x) = r^x \mod m. \tag{6.45}$$

The two partners A and B each pick elements of $\mathbb{Z}_m$, say A picks a and B picks b. Then

A sends $f(a) = \alpha$ to B, and

B sends $f(b) = \beta$ to A.

The partners do not care if these communications are intercepted. Upon receipt,

A calculates $k = \beta^a \mod m = r^{ba} \mod m$, and

B calculates $k = \alpha^b \mod m = r^{ab} \mod m$.

Because $ab = ba$, A and B arrive at the same key, k. Any interceptor of the communications would have trouble determining a or b and hence, it is presumed, k, from $\alpha = f(a)$ and $\beta = f(b)$.

6.7.3 Digital coin flipping

How can two people on opposite coasts accomplish a fair coin flip over the telephone? A classic example involves a divorcing couple trying to decide who gets the car. The spouse in California flips a coin, the spouse in New York calls "Tails!", and the spouse in California says, "Sorry, you lose!". The New York spouse might not be convinced that the process was fair.

Maybe each of the two people, A and B, could choose either 0 or 1, with the understanding that A wins if their choices agree, while B wins if the choices disagree. The choices could be transmitted to a third party, who would then announce the result. But the involvement of a third party would lead to complication and delay and would also raise the problem of trustworthiness of that arbiter, who could perhaps be bribed or otherwise influenced.

Maybe A and B could simultaneously send their choices to each other. But achieving true simultaneity and preventing cheating present their own problems.

A long-distance fair coin flip can be accomplished by using a one-way function f (say from $\mathbb{Z}$ to $\mathbb{Z}$ or from $\mathbb{Z}_m$ to $\mathbb{Z}_m$ for some m) which has the property that given $f(x)$ it is not only essentially impossible to determine x but also even the *parity* of x, that is, $x \mod 2$. Particular such functions can be constructed using number theory—see [**?**, p. 52]. Here is one procedure to accomplish a fair long-distance coin flip:

A chooses $a \in \{0, 1, \ldots, m-1\}$, B chooses $b \in \{0, 1, \ldots, m-1\}$.

A sends $f(a)$ to B, B sends b to A.

Having received b, A can compute $a + b \mod 2$ and determine the winner.

A sends a to B. Now B can also compute $a + b \mod 2$ and determine the winner.

B can plug the received a into f and verify that it does indeed produce the value $f(a)$ previously transmitted by A, showing that A did not change the value of a after receiving b, and thus the process was fair.

References

1. Gilles Brassard, *Modern Cryptology: A Tutorial*, Lecture Notes in Computer Science, vol. 325, Springer-Verlag, 1988.

2. Simon Singh, *The Codebook*, Anchor Books, 1999.

Part 7

Shannon's information theory

We will learn here (1) that *entropy* is a measure of the fundamental information content of messages, giving the minimal number of binary bits per symbol needed to encode the source; (2) that information sources can be *compressed* via block coding, so that the number of binary digits in the recoded message per symbol in the original message is very near the entropy; and (3) that even if messages are going to be altered by random noise, they can first be encoded in such a way that the original message can be recovered with an arbitrarily low probability of error. These startling statements were discovered in the late 1940's by Claude Shannon, and they enable many of the technological marvels we use today: computers, CD's and DVD's, telephones, and so on. This is pretty much a retelling of the account in [**?**].

7.1 Information sources

The mathematical study of information processing begins with the concept of an *information source*. In our everyday experience one of our most familiar sources of information is the natural (human) language which we may be hearing or reading at any particular time. The sources of mathematical information theory do include idealized models of natural languages, but they are not the only starting point for analyzing such languages rigorously. The field of *formal languages* seeks to understand the basic principles governing human languages, computer languages, and even structures in chemistry, biology, and the physics of materials, by carefully analyzing the properties of abstract languages.

Let D be a finite alphabet. The elements of D will be called *symbols* or *letters*. Frequently $D = \{0, 1, \ldots, d-1\}$ for some $d \in \mathbb{N}$. The symbol D^* denotes the set of all *finite strings* on the symbols in D, including the empty string, ϵ.

Definition 7.1.1. A *formal language* is any set of finite strings on a finite alphabet, that is, any $\mathcal{L} \subset D^*$ for any finite alphabet D.

This definition is very very broad, It's useful, because it is so inclusive. But it's not too useful, since it includes so many languages of so many different kinds. Analysis can begin when we consider particular examples of languages and languages of particular kinds.

Example 7.1.1. Let $D = \{0, 1\}$ and let $\mathcal{L}_e$ denote the set of all strings on the alphabet D which contain an *even* number of 1's.

Example 7.1.2. Again let $D = \{0, 1\}$ and let $\mathcal{L}_p$ denote the set of *palindromes* on D, that is, all strings on the alphabet D which read the same forwards as backwards.

Example 7.1.3. Let $\mathcal{L}_=$ denote the set of all strings on $D = \{0, 1\}$ which contain the same number of 0's as 1's.

Example 7.1.4. Let $\mathcal{L}_{11}$ denote the set of all strings on $D = \{0, 1\}$ which do not contain two consecutive 1's.

Exercise 7.1.1. For each of the four languages in the preceding examples, list all the words in the language of length less than or equal to four.

In the *Notes on Elementary Probability* we defined an *information source* as a system $\mathcal{X} = (\Omega(D), \mathcal{B}, \mathcal{P}, \sigma)$, or $(\Omega^+(D), \mathcal{B}, \mathcal{P}, \sigma)$. Here D is a finite alphabet; $\Omega(D)$ and $\Omega^+(D)$ are the sets of one- or two-sided infinite strings on the alphabet D; $\mathcal{B}$ is a family of subsets of the set of strings on D which contains all the cylinder sets and is closed under complementation and countable unions and intersections; P is a probability measure defined for all sets in $\mathcal{B}$; and σ is the shift transformation, which makes time go by or allows us to read one incoming symbol from the source at a time. The set of all finite strings found in all infinite sequences emitted by the source is *the language of the source*. Usually we deal with sources that are stationary and ergodic. By a major theorem of the twentieth century, every word in the language of a stationary ergodic source appears in almost every sequence emitted by the source with limiting frequency equal to the probability of the cylinder set to which it corresponds.

7.2 The possibility of information compression

One if by land, two if by sea;
And I on the opposite shore will be,
Ready to ride and spread the alarm
Through every Middlesex village and farm,
For the country folk to be up and to arm.

Henry Wadsworth Longfellow

Suppose that someone has a choice of only two messages to send. Each message could consist of even a lengthy string of symbols, but the information boils down to just the choice between two alternatives: whether it is message A that is sent or message B. The messages could be numbered 0 and 1, or represented by one lantern or two in the steeple of Christ Church, Boston. (Interestingly, 0 lanterns or 1 lantern might not work: often one needs also to signal that a message is present.) This situation shows us in action the fundamental unit of information: the designation of one of two possibilities constitutes the *bit*. Information is measured by the minimum number of bits—elementary yes-no decisions—required to convey it.

In the Paul Revere story, we see that the message "The British are coming by land" has been *compressed* to the message "1", and the message "The British are coming by sea" has been compressed to the message "2". Any two symbols could have been used instead of 1 and 2, for example 0 and 1 would be a basic choice. But it takes a minimum of two symbols to distinguish the two distinct messages.

If we have 3 or 4 messages to choose among, we can distinguish them by assigning to each a different string of length 2 on the symbols 0 and 1. The original message strings could be quite long, but once the assignment of strings is made and agreed on, we have to send only 00, 01, 10, or 11. The messages have been compressed. If we have m messages to send, and we want to assign to each one a different binary string of length k, we must have

$$2^k \geq m, \qquad \text{that is,} \qquad k \geq \log_2 m : \tag{7.1}$$

It takes binary strings of length $\log_2 m$ to distinguish m different messages.

Suppose now that someone has a repertoire of messages from which to choose, sending one a day, but with varying probabilities. Suppose further that the messages have widely varying lengths. We can also assign binary strings of varying lengths to these messages, and it would be smart to use shorter strings for the more probable messages. Then on average we will be sending fairly short messages, achieving information compression.

Example 7.2.1. Dagwood goes to the same diner for lunch every day. Usually he orders either "the special", whatever it is, unless he doesn't like it, and then he orders chili, extra hot, with lettuce and tomato on the side, ranch dressing, and two croissants. In the latter case, to save time, he just says "the usual". Very rarely, he is tired of chili and the special is unattractive, so he orders "pea soup and meat loaf with French fries". Note the economical use of letters on the average.

Example 7.2.2. During the summer in North Carolina, the weather forecast is often for partly cloudy weather, temperature in the high 80's, chance of afternoon and evening thundershowers. Sometimes TV forecasters will even say, "Same as yesterday". When the weather is unusual (which happens not so often), longer descriptions take place.

Here is a fairly efficient way to encode English text for compression that is suggested in the book by Pierce. Let us take for our *source alphabet* D the 26 English letters plus space, ignoring upper and lower case, punctuation marks, and other special characters. (If really needed, they can be spelled out.) In order to encode these $m = 27$ symbols by binary strings, that is, by a *code alphabet* $C = \{0, 1\}$, we would need strings of length at least $k = 5$, since 5 is the smallest integer k for which $2^k \geq m$ ($2^5 = 32$). But we propose to encode text not letter by letter, but actually word by word, at least for the *most probable* words.

Note: In what follows, we will also consider a *channel*, which typically accepts one binary digit per second and outputs one binary digit per second (maybe different strings come out than are put in). Then the code alphabet $C = \{0, 1\}$ will also be the *channel alphabet*. Any source to be sent across the channel will have to be first encoded into strings on the channel alphabet. (This happens all the time, for example when data put into a computer is converted to binary strings.)

Suppose we use binary strings of length 14 to encode our source, standard English text on an alphabet of 32 characters. There are $2^{14} = 16,384$ such strings. Let's assign $16,384 - 32 = 16,352$ of these strings to the $16,352$ most common English words and assign the 32 remaining strings to our 26 letters, space, and the most common punctuation symbols, so as to be able to spell out any words not in our list of the most probable $16,352$.

If we go to encode an English text into binary strings of length 14 in this way (Plan A), most of the time we will be encountering the most frequent words, and occasionally we will have to spell out an unusual word that comes up. Is there any saving over character-by-character encoding? If we used binary strings of length 5 to encode each of our 32 characters (Plan B), an English string of length N would be encoded to a binary string of length $5N$. What length binary string would Plan A yield?

The answer depends on the lengths of the words involved, since in Plan A we encode word by word. Pierce states that on average English words have length 4.5 characters, hence really 5.5 since a space precedes each word. So in Plan A, our input English text of length N contains approximately $N/5.5$ words, and most of them are probable. Each of these probable words is assigned a binary string of length 14, and the result is a binary string that is not much longer than $(N/5.5)14 \approx 2.55N$. Thus Plan A gives us an encoding that is about *half the length* of Plan B's character-by-character encoding.

In Plan B, we are using 5 binary digits per symbol. In Plan A we use only 2.55 binary digits per symbol—some excess, redundant, or predictable chaff has been winnowed out. Somehow the essential information in the message is less than 2.55 *bits per symbol*. Note that Pierce uses *bit* for the basic unit of information—a fundamental yes-no answer—and *binary digit* for the symbols 0 and 1.

The minimum number of binary digits per symbol needed to encode a message is taken to be the information content per symbol of the message. It is called the entropy *and is measured in bits per symbol.*

So we could have a source, such as English text, that is presented as 5 binary digits per symbol, say if we look at it character by character. But in actual fact the entropy of English text is less than 5 bits per symbol–on average, English text can be encoded by binary strings of length 2.55 per symbol. In fact, it turns out that English text has entropy about *one bit per symbol*.

A typical English text of length N (a string on 32 characters, say) can be encoded to a *binary string of approximately the same length*!

7.3 The entropy of a source

The idea was announced at the end of the preceding section: the entropy of an information source is the minimum number of binary digits per symbol needed, on average, to encode it. The entropy is measured in bits per symbol. We will see how to try to achieve this compression by block coding. For now, we focus on three equivalent definitions of the entropy of a source, which do not require considering all possible encodings. Each definition gives its own insight into the idea.

1. The entropy of a source is the *average information per symbol* that it conveys. How to quantify this?

One key idea is that *the amount of information gained equals the amount of uncertainty removed.*

Another is that when we are told that a highly probable event has occurred we gain less information than when we are told that a highly improbable event has occurred. So when we are told that an event A of probability $P(A)$ has occurred, we say that *the amount of information gained is* $-\log_2 P(A)$ *bits.* A couple of points to help explain this:

- If $P(A) = 1$, so that A is *almost sure* to occur, then the information conveyed in telling us that A occurred is 0.
- If $P(A)$ is very small, then when we are told that $P(A)$ occurred we gain a *lot* of information—we are really really surprised.
- We use $\log_2$ (logarithm base 2) because we want to measure the information in *bits*—the number of binary digits required to convey it. Recall that each simple yes-no distinction can be conveyed by giving one binary digit, 0 or 1. If the two possibilities are equally likely, each yields one bit of information. This gibes with the fact that if $P(A) = P(A^c) = 1/2$, then $-\log_2 P(A) = \log_2(1/2) = \log_2(2) = 1$. If we use *natural logarithms*, that is $\ln = \log_e$ ($e = 2.718281828459045\dots$), the unit of information is sometimes called the *nat.*
- If A_1 and A_2 are *independent events*, so that $P(A_1 \cap A_2) = P(A_1)P(A_2)$, then the information gained in learning that *both* events occurred is the *sum* of the information gains from learning that *each* of them has occurred. This is reasonable, since for independent events the occurrence of either one does not affect the probability of occurrence of the other. This point is one of the main justifications for the use of the logarithm in the definition of information.

Exercise 7.3.1. Calculate $\log_2 x$ in case $x = 32, 1/16, 20$, and -4.

Suppose we have a source that is emitting symbols from a finite alphabet $D = \{0, 1, \dots, d-1\}$, symbol i occurring with probability p_i (but the symbols do not necessarily occur independently of one another). If we were to read just one symbol, the one that is arriving just now, our *average information gain* would be

$$H(p) = -\sum_{i=0}^{d-1} p_i \log_2 p_i. \tag{7.2}$$

Note that here our probability space is $(\Omega(D), \mathcal{B}, P)$, where $\Omega(D)$ is the set of two-sided infinite sequences on the alphabet D. If we define $f(x) = -\log_2 p_{x_0}$, the logarithm of the probability of occurrence of the symbol being read at time 0, then the above average information gain is just the *expected or average value* of the random variable f:

$$H(p) = \mathbb{E}(f). \tag{7.3}$$

If the symbols were actually coming independently (that is, if we had a Bernoulli source—see §3 of the notes on probability) Equation 7.2 would actually give the entropy of the source. In general, though, occurrence of certain symbols can influence the probabilities of occurrence of other ones at various times, thereby affecting the amount of new information gained as each symbol is read. The three definitions of entropy of a source which we are here working up deal with this possible interdependence in three different ways. The first one suggests that we *sliding block code* the word into long blocks, say of length n, considering these as new symbols; apply Formula (7.2) to this new source to get an idea of its entropy (actually an overestimate); divide by n to get an overestimate of the entropy of the original source, in bits per original symbol; and hope that these numbers converge to a limit as $n \to \infty$ (in fact they will).

Let's describe this more precisely, with mathematical notation. Denote by D^n the collection of all words of length n on the elements of the alphabet D. The sliding block code mentioned above associates to each infinite sequence $x \in \Omega(D)$ with entries from D the sequence $s_n(x) \in \Omega(D^n)$ with entries n-blocks on the alphabet D defined by

$$(s_n(x))_j = x_j x_{j+1} \dots x_{j+n-1} \quad \text{for all } j \in \mathbb{Z}. \tag{7.4}$$

As we shift along the sequence x *one* step at a time, we see a sequence of symbols from D^n, which is the new sequence on the new alphabet of n-blocks.

Carrying out the strategy outlined in the preceding paragraph leads to the definition of *the entropy of the source* as

$$h(\mathcal{X}) = \lim_{n\to\infty} \left[-\frac{1}{n} \sum_{B \in D^n} P(B) \log_2 P(B) \right]. \tag{7.5}$$

We repeat that this number is supposed to give the average information received per symbol (which equals the amount of uncertainty removed, or the degree of surprise) as we read the symbols emitted by the source one by one.

2. The second definition measures our average uncertainty about the next symbol to be read, given all the symbols that have been received so far. Naturally this involves the idea of conditional probability—see §2 of the *Notes on Elementary Probability*. Given a symbol $s \in D$ and a block $B \in D^n$, let us agree to denote by $P(s|B)$ the conditional probability that the next symbol received will be s, given that the string of symbols B has just been received. Since our source is stationary, the time when we are waiting to see whether s is sent does not matter, so we may as well take it to be the present, time 0. Thus

$$\begin{aligned} P(s|B) &= \frac{P\{x \in \Omega(D) : x_{-n} \dots x_{-1} = B, x_0 = s\}}{P\{x \in \Omega(D) : x_{-n} \dots x_{-1} = B\}} \\ &= \frac{P(Bs)}{P(B)}. \end{aligned} \tag{7.6}$$

Given that a block $B \in D^n$ has just been received, according to the preceding discussion (in connection with Definition 1 of entropy), our average information gain upon receiving the next symbol is

$$-\sum_{s \in D} P(s|B) \log_2 P(s|B). \tag{7.7}$$

We average this over all the possible n-blocks B, weighting each according to its probability, and then take the limit as $n \to \infty$, so that we are allowed to look farther and farther back into the past to decide how surprised we ought to be when a symbol s arrives at time 0. This gives us the second definition of the entropy of the source:

$$h(\mathcal{X}) = \lim_{n\to\infty} \left[-\sum_{B \in D^n} P(B) \sum_{s \in D} P(s|B) \log_2 P(s|B) \right]. \tag{7.8}$$

We repeat that this measures our average information gain when a symbol is received at time 0, given all of the symbols that have been received in the past.

3. The third description that we present of the entropy of a (stationary, ergodic) source is actually a theorem, due to Claude Shannon, Brockway McMillan, and Leo Breiman. It leads to methods for finding the entropy of a source by examining the typical sequences that it emits. With probability 1,

$$h(\mathcal{X}) = \lim_{n\to\infty} -\frac{1}{n}\log_2 P(x_0x_1\dots x_{n-1}). \tag{7.9}$$

This implies that for large n, there are approximately $2^{nh(\mathcal{X})}$ *probable n-blocks* B on the symbols of D, each with probability approximately $2^{-nh(\mathcal{X})}$. This is because for the initial n-block $B = x_0x_1\dots x_{n-1}$ of a typical sequence x emitted by the source, for large n we will have

$$h(\mathcal{X}) \approx -\frac{1}{n}\log_2 P(B), \tag{7.10}$$

so that

$$\log_2 P(B) \approx -nh(\mathcal{X}) \quad \text{and hence} \quad P(B) \approx 2^{-nh(\mathcal{X})}. \tag{7.11}$$

This third "definition" of the entropy $h(\mathcal{X})$ of a stationary ergodic source $\mathcal{X}$, as the exponential growth rate of the number of probable n-blocks, substantiates our initial effort to understand the entropy of a source as the minimal average number of binary digits per symbol needed to encode it. Recall Formula (7.1): In order to be able to assign a different binary string of length k to each of m messages, we need $k \geq \log_2 m$. Now choose n large enough so that there are approximately $m = 2^{nh(\mathcal{X})}$ probable n-blocks. Using the strategy outlined in the preceding section, with $k \geq \log_2 m = nh(\mathcal{X})$, we can assign a binary string of length k to each of these probable n-blocks (reserving a few binary strings to assign to basic characters, so as to be able to spell out improbable words). The result is that, on the average, strings of length n are coded one-to-one by binary strings of length $k \approx \log_2 m = nh(\mathcal{X})$, so that the source is encoded by

$$\frac{k}{n} \approx \frac{nh(\mathcal{X})}{n} = h(\mathcal{X}) \quad \text{binary digits per symbol.} \tag{7.12}$$

(It takes some further argument to see that it is not possible to improve on this rate.)

Example 7.3.1. The entropy of a *Bernoulli source* (see §3 of the *Notes on Elementary Probability*) which emits symbols from an alphabet $D = \{0, 1, \dots, d-1\}$ independently, symbol i appearing with probability p_i, is

$$H(p_0, \dots, p_{d-1}) = -\sum_{i=0}^{d-1} p_i \log_2 p_i. \tag{7.13}$$

This can be seen by easy calculation based on any of the three definitions of entropy given above.

Example 7.3.2. Consider a *Markov source* as in §4 of the *Notes on Elementary Probability* on the alphabet $D = \{0, 1, \dots, d-1\}$ determined by a probability vector $p = (p_0, \dots, p_{d-1})$, which gives the probabilities of the individual symbols, and a matrix $P = (P_{ij})$, which gives the probabilities of transitions from one symbol to the next, so that

$$P\{x \in \Omega(D) : x_n = j | x_{n-1} = i\} = P_{ij} \quad \text{for all } i, j, n. \tag{7.14}$$

A short calculation based on Definition 2 above shows that this source has entropy

$$H(p, P) = -\sum_{i=0}^{d-1} p_i \sum_{j=0}^{d-1} P_{ij} \log_2 P_{ij}. \tag{7.15}$$

Exercise 7.3.2. Suppose we have a source $\mathcal{X}$ on the alphabet $D = \{0, 1\}$ which emits 0's and 1's with equal probabilities, but subject to the requirement that no 0 can be emitted immediately after another 0, and no 1 can be emitted immediately after another 1. Use each of the three preceding definitions of entropy to calculate the entropy of this source.

Exercise 7.3.3. Given a source $\mathcal{X} = (\Omega(D), \mathcal{B}, \mathcal{P}, \sigma)$ as above and $n = 1, 2, \ldots$, we can associate to it the *n-blocks source*

$$\mathcal{X}^{(n)} = (\Omega(D^n), \mathcal{B}^{(n)}, \mathcal{P}^{(n)}, \sigma^n), \tag{7.16}$$

as in the beginning of §3 of the *Notes on Elementary Probability*. The symbols of the new source are n-blocks on the original alphabet D, the observable events and probability measure are defined in a natural way (note—not necessarily assuming that the n-blocks come independently), and we now shift n steps at a time instead of 1, in order always to shift to the next n-block. Use as many as possible of the definitions or explanations of entropy given above to verify that $h(\mathcal{X}^{(n)}) = nh(\mathcal{X})$.

7.4 The entropy of a language

We have just seen that a stationary ergodic source emits, for large n, about $2^{nh(\mathcal{X})}$ probable blocks of length n. Given any language $\mathcal{L}$ on a finite alphabet D, we can ask for a number $h(\mathcal{L})$ such that the *total number* of words of length n in the language is approximately $2^{nh(\mathcal{L})}$. If we define $\mathcal{L}_n$ to be the set of words in $\mathcal{L}$ that have length n, and $|\mathcal{L}_n|$ to be the size of this set, that is, the number of words in $\mathcal{L}$ that have length n, we seek

$$h(\mathcal{L}) = \lim_{n\to\infty} \frac{1}{n} \log_2 |\mathcal{L}_n|. \tag{7.17}$$

Sometimes this limit exists, and then we call it the *entropy* or *topological entropy* of the language. If $\mathcal{X}$ is a source that emits only words from a language $\mathcal{L}$, then it follows from above that

$$h(\mathcal{X}) \leq h(\mathcal{L}). \tag{7.18}$$

Exercise 7.4.1. Calculate or at least estimate the entropies of the languages in the examples in §1.

7.5 Source coding for compression

So far we have implicitly been assuming that we are dealing with a *noiseless channel*: a source $\mathcal{X}$ is emitting symbols from an alphabet D, and we receive exactly what is sent (or at least we can recover, with probability 1, what was sent from what was received). (Soon we will consider the possibility that there may be random errors, so that what is received may differ from what was sent.) We have also considered *recoding* the source, perhaps replacing strings on the alphabet D by binary strings, so as to improve (lower) the average number of new binary digits per old symbol, or to be able to input our messages to a channel which accepts and puts out only binary strings. For example, encoding English text character by character requires about 5 binary digits per character (allowing for 32 characters); but we saw above how to recode so that in a long typical text we use only about 2.55 binary digits per English character.

Shannon's Source Coding Theorem says that given a stationary ergodic source $\mathcal{X}$ of entropy $h(\mathcal{X})$, it is possible to encode the strings output by the source in a one-to-one (uniquely decodable) way so that the average number of binary digits used per source symbol is as close as we like to the entropy of the source. Moreover, we cannot recode to get the average number of binary digits used per source symbol to be less than the entropy of the source.

In Section 7.3 we saw the possibility of doing this sort of "information compression", but the method envisaged there depended on forming a list of the most probable blocks of a certain length that are emitted by the source that we are trying to code. There are more efficient and practical ways to accomplish

this goal, devised by Shannon and Fano, Huffman, Lempel and Ziv (used for example in computer data compression programs like gzip), and others. Pierce presents the method of *Huffman coding*:

List the symbols to be encoded (possibly the set D^n of words of some fixed length length n emitted by the source) in order of decreasing probabilities. We operate on this list of probabilities, converting it into a new list. Choose two of the lowest possible probabilities in the list and draw lines from each to the right that meet. Label the upper line with 1 and the lower with 0. At the joining point, enter the sum of the two probabilities; this replaces the pair of probabilities that we started with, thus giving us a new list of probabilities whose length is 1 less than that of the original.

Continue in this way, constructing a tree with edges labeled by 1 and 0 and vertices by probabilities, until we arrive at a vertex labeled by probability 1. Now starting from this final vertex, reading right to left, write down the labels on the edges encountered as one traverses the graph toward an original symbol s. The resulting binary string, $b(s)$, is the codeword assigned to s, and the set of codewords is called the *Huffman code* assigned to the set of source words to be encoded.

Here are a couple of important points about this Huffman coding procedure:

- The Huffman code is a *variable-length* code: each of m source symbols is assigned a binary string of length less than or equal to m.
- The set of codewords is *prefix free*: no code word is an initial segment of any other codeword. This guarantees that the code is *invertible*: each string of codewords can be parsed into a unique string of source words.
- When a string of symbols from a *Bernoulli* source $\mathcal{X}$ of entropy $h(\mathcal{X})$ is encoded by the Huffman procedure, the average number of binary digits used per symbol is less than $h(\mathcal{X}) + 1$.

This last observation requires some proof, which we do not attempt here. But we can see immediately how it could yield Shannon's Source Coding Theorem. Choose a large n and form the source $\mathcal{X}^{(n)}$ which reads n-blocks of the original source, always moving from one full n-block to the next (by applying the n'th power of the shift transformation). By Exercise 7.3.3, this new source has entropy $h(\mathcal{X}^{(n)}) = nh(\mathcal{X})$. Moreover, it is approximately Bernoulli, in the sense that

$$-\sum_{B \in D^n} P(B) \log_2 P(B) \approx h(\mathcal{X}^{(n)}) \tag{7.19}$$

(see Formula (7.5)). When the symbols of $\mathcal{X}^{(n)}$ are encoded to binary strings by the Huffman procedure, according to the above we use, approximately and on average, less than $h(\mathcal{X}^{(n)}) + 1 = nh(\mathcal{X}) + 1$ binary digits per symbol of $\mathcal{X}^{(n)}$, and therefore we use on average less than

$$\frac{h(\mathcal{X}^{(n)}) + 1}{n} = \frac{nh(\mathcal{X}) + 1}{n} = h(\mathcal{X}) + \frac{1}{n} \tag{7.20}$$

binary digits per symbol of the original source $\mathcal{X}$.

Finally, we attempt to explain the discussion in Pierce, pp. 97–98, about the relation of this theorem to a kind of channel capacity. Suppose that we have a noiseless channel which accepts binary strings as inputs and also puts out binary strings—maybe not exactly the same ones, but at least in an invertible manner, so that the input can be recovered from the output, at least with probability 1. Let us also introduce a time element for the channel, so that it can transmit τ binary digits per second. Now if n-blocks from a source $\mathcal{X}$ are coded to binary strings of length k to be fed into the channel, presumably with k/n on average not much larger than $h(\mathcal{X})$, then τ binary digits per second will correspond to approximately $\tau n/k$ symbols of the original source traversing the channel per second (in binary digit encoded form).

Example 7.5.1. Suppose we have a source whose 2-blocks are coded by binary strings of average length 4, and we have a channel of time-capacity $\tau = 5$ binary digits per second. Then binary strings move across the channel at 5 digits per second. This corresponds to source strings being input at one end of the channel and output at the other end, without taking into account any encoding or decoding time, at $5(2/4)$ symbols per second. Each binary string of length 1 makes it across the channel in $1/5$ seconds, so each binary string of length 4 makes it across the channel in $4/5$ seconds and corresponds to a source string of length 2, so source strings are being conveyed at $(4/5)/2$ seconds per symbol. Then take the reciprocal.

Thus we have an average transmission rate of source symbols across this channel of

$$\tau \frac{n}{k} = \frac{\tau}{k/n} \quad \text{source symbols per second.} \tag{7.21}$$

Recalling that we can make k/n, which is always greater than or equal to $h(\mathcal{X})$, as close as we like to $h(\mathcal{X})$, but not any smaller, we see that the rate in (7.21), which is always smaller than $\tau/h(\mathcal{X})$, can be made as close as we like to $\tau/h(\mathcal{X})$, but not any greater. This is supposed to explain the set-off statements at the top of p. 98 in Pierce.

Exercise 7.5.1. A source emits once a second one of 6 different symbols, with probabilities $1/2, 1/4, 1/8, 1/16, 1/32, 1/32$, respectively, independently of what has been emitted before or will be emitted later.

(a) Calculate the entropy of the source.

(b) Devise a Huffman code for this source.

(c) If a typical long message of length N emitted by the source is recoded with the code from part (b), what will likely be the length of the resulting binary string?

(d) If the source is Huffman encoded as in Part (b) and then is put into a channel that conveys 0.57 binary digits per second, what will be the transmission rate in source symbols per second?

7.6 The noisy channel: equivocation, transmission rate, capacity

In the preceding section we saw how to recode a source $\mathcal{X}$ of entropy $h(\mathcal{X})$ to *compress* its output, representing it by binary strings with a rate not much more than $h(\mathcal{X})$ binary digits per source character. Now we consider how to add some *check digits* to protect against errors in transmission. As before, we suppose that the output of our source is recoded into binary strings and fed into a channel which accepts and puts out binary strings (or more generally accepts strings on an alphabet A and puts out strings on an alphabet B), but we now admit the possibility of random alteration of the strings during transmission, so that the input may not be recoverable, with probability 1, from seeing the output. Can we protect against this? How well? At what cost?

One way to *detect* errors in transmission is just to repeat every digit twice: for example, to send a string 010010, put 001100001100 into the channel. This method will detect single isolated errors, but it entails the cost of doubling transmission time. To *correct* single isolated errors, one could triple each digit, and this would multiply the cost even more. Can we correct *all errors* (with probability 1)? If so, at what cost?

A *channel* is actually a family of probability measures, one for each infinite string that can be fed into the channel, specifying the probabilities of the observable events regarding the output string, namely that in the output we see a particular finite-length block at a particular time. Further technical conditions must be put on channels to ensure that they make sense and that the theorems we want to prove are true. This work is far from finished. In this section we deal with a very simple kind of noisy channel, the *discrete memoryless channel*, for which each incoming symbol produces the various possible output

symbols with fixed probabilities, independently. Thus we are given transition probabilities

$$\begin{aligned} p_x(y) &= \text{ probability that } y \text{ comes out when } x \text{ is put in,} \\ q_y(x) &= \text{ probability that when } y \text{ comes out } x \text{ was put in,} \end{aligned} \tag{7.22}$$

where x, y are symbols of the channel alphabet (for us usually 0 and 1). We will use the same notations for analogous probabilities when x and y are *strings* on the input and output alphabets.

Given such a channel, it turns out that we can associate to it an important number, C, called the *capacity* of the channel, which tells the complete story about the prospects for coding a source for error protection before trying to send it across that channel. We will also define a quantity E, called the *equivocation*, which is a measure of the amount of information lost on average when messages are sent across the channel. We also consider F, the *average frequency of error*, the proportion of digits in the limit (in long messages) which are changed in transmission across the channel. Once these quantities are understood, we can appreciate

Shannon's Channel Coding Theorem: Given a channel of capacity C, a source $\mathcal{X}$ of entropy $h(\mathcal{X}) < C$, and any number $\epsilon > 0$, we can recode the source so that when it is sent over the channel and the result is decoded both the frequency of error and the equivocation are less than ϵ. If $h(\mathcal{X}) > C$, we must always have the equivocation $E \geq h - C$, but by recoding the source we can make it as close to $h - C$ as we like: given $\epsilon > 0$, we can have $h - C \leq E < h - C + \epsilon$.

In this statement, entropy, capacity, etc. are thought of in *bits per second*. If we eliminate time as a consideration, then this remarkable result says that any (stationary ergodic) source can be sent across any (binary symmetric) channel virtually error free: first compress the source by source coding, then assign n-blocks of the source to long enough k-blocks of the channel. The theorem guarantees that we can recode to reduce the frequency of errors, and the equivocation, below any desired level. Similar results apply to more general channels, but satisfyingly general and clean results have not yet been achieved.

It is also remarkable that this theorem is proved by a *random coding argument*—basically counting. One selects k and n large and with n/k having the right ratio (close to $C/h(\mathcal{X})$ in case all alphabets involved are $\{0, 1\}$). Then a calculation is done to show that when each n-block of the source is assigned to a different randomly chosen k-block to be put into the channel, with high probability we get a recoding of the source that satisfies the theorem. Another way to say it refers to the approximately $2^{nh(\mathcal{X})}$ "good n-blocks" put out by the source (see the third definition given above of the entropy of the source). It turns out that there are also approximately 2^{kC} *distinguishable k-blocks* that can be put into the channel, blocks whose corresponding outputs will probably be very different from one another. (In fact when we choose this many k-blocks at random, if k is large enough we will with high probability get a set of distinguishable blocks.) So when we assign the good n-blocks of the source to the distinguishable k-blocks to be put into the channel, we code most of the output of the source in a way that the channel is unlikely to confuse.

To finish off our discussion of this amazing theorem, we have to define the key terms "capacity" and "equivocation", and before that we have to define and examine some other concepts as well. We assume that we have a stationary ergodic source $\mathcal{X}$, with an alphabet A, that has already been encoded, if necessary, so that its alphabet matches the input of our channel. When this source is put into the channel, the output, the composition of input source and channel, constitutes *another source*, $\mathcal{Y} = \mathcal{S}(\mathcal{X})$, on another alphabet B (usually $\{0, 1\}$). We assume that the channel has the property that this source $\mathcal{Y}$ will *also* always be stationary and ergodic. (It is difficult to describe exactly which channels have this property, but the binary symmetric one does.) We will now consider strings x of length n that are put into the channel and examine the strings y of length n that emerge (we assume zero time delay).

(1) The *joint input-output process* $\mathcal{X} \vee \mathcal{Y}$ is the source whose output consists of infinite strings of pairs of symbols (a, b), with a a symbol emitted by the source $\mathcal{X}$ at each time j and fed into the channel, and b the symbol output by the channel at the same time j. This corresponds to a know-it-all observer who can see both the input and the output of the channel at the same time. To avoid complicated

notation, we use the same symbol P to describe the probabilities of events related to this process as for the processes $\mathcal{X}$ and $\mathcal{Y}$. The entropy of the joint input-output process is

$$h(\mathcal{X} \vee \mathcal{Y}) = -\lim_{n\to\infty} \frac{1}{n} \sum_{\substack{x\in A^n \\ y\in B^n}} P(x,y) \log_2 P(x,y). \tag{7.23}$$

(2) We define the *uncertainty about the output, given the input*, to be

$$H_{\mathcal{X}}(\mathcal{Y}) = -\lim_{n\to\infty} \frac{1}{n} \sum_{\substack{x\in A^n \\ y\in B^n}} p(x) p_x(y) \log_2 p_x(y). \tag{7.24}$$

The sum gives the weighted average, over all input strings x of a given length n, of the uncertainty about the output (which equals the amount of information gained or the degree of surprise when the output is received) when the input string x is known.

(3) We define the *uncertainty about the input, given the output*, to be

$$H_{\mathcal{Y}}(\mathcal{X}) = -\lim_{n\to\infty} \frac{1}{n} \sum_{\substack{x\in A^n \\ y\in B^n}} q(y) q_y(x) \log_2 q_y(x). \tag{7.25}$$

The sum gives the weighted average, over all output strings y of a given length n, of the uncertainty about the input when the output string y is known. This quantity measures the loss of information when the source is sent across the channel, and it is called the *equivocation*.

(4) We have the key relation

$$h(\mathcal{X} \vee \mathcal{Y}) = h(\mathcal{X}) + H_{\mathcal{X}}(\mathcal{Y}) = h(\mathcal{Y}) + H_{\mathcal{Y}}(\mathcal{X}). \tag{7.26}$$

The first equality says that the uncertainty about what will come out of the joint source is the sum of the uncertainty about what is put into the channel plus the uncertainty about what will come out of the channel when what is put in is already known. Thus these equations seem to make some sense.

(5) When a source $\mathcal{X}$ is connected to the channel as above, producing an output process $\mathcal{Y}$ and a joint input-output process $\mathcal{X} \vee \mathcal{Y}$, we define the *information transmission rate* to be

$$\begin{aligned} R(\mathcal{X}) &= h(\mathcal{X}) - H_{\mathcal{Y}}(\mathcal{X}) \\ &= h(\mathcal{Y}) - H_{\mathcal{X}}(\mathcal{Y}) \\ &= h(\mathcal{X}) + h(\mathcal{Y}) - h(\mathcal{X} \vee \mathcal{Y}). \end{aligned} \tag{7.27}$$

This probably calls for some discussion. The first expression for R, namely $R(\mathcal{X}) = h(\mathcal{X}) - H_{\mathcal{Y}}(\mathcal{X})$, says that the rate at which information comes across the channel equals the rate at which the source puts information into the channel, minus the amount lost in the channel—the uncertainty remaining about what the input was once we see the output. The second expression says the rate is the amount of information received (uncertainty about the output) , minus the spurious extra uncertainty due to noise in the channel—the uncertainty about what will come out even when we know what was put in. The third expression says that information flow across the channel is given by the sum of the uncertainties about the input and output separately, minus the uncertainty about the process in which the input and output are connected.

(6) Finally we can define the *capacity* C of the channel to be the maximum (more precisely *supremum*, abbreviated sup—the supremum of the interval $[0,1)$ is 1, while this set has no maximum) of all the possible information transmission rates over all the possible stationary ergodic sources that can be fed into the channel:

$$C = \sup\{R(\mathcal{X}) : \mathcal{X} \text{ is a source input to the channel}\}. \tag{7.28}$$

Example 7.6.1. Let's consider the binary symmetric channel in which when a symbol 0 or 1 emerges, there is probability $\alpha \in (0,1)$ that it is the same as the symbol that was put in, independently of all other symbols. Thus

$$q_0(0) = q_1(1) = \alpha, \quad \text{while} \quad q_0(1) = q_1(0) = 1 - \alpha. \tag{7.29}$$

There are reasons to believe that the maximum information transmission rate for such a channel will be achieved by the Bernoulli source (see §6 of the *Notes on Elementary Probability*) which at each instant emits one of the the symbols 0 and 1, each with probability 1/2, independently of what is emitted at any other instant. (At least for this source we have maximum uncertainty about what it will emit, and the symmetry of the channel makes us think that the optimal source should also be symmetric). Then the symbol output probabilities are $q(0) = q(1) = 1/2$, and because of independence in Formula (7.25) we can look just at 1-blocks and do not have to take a limit. We find that

$$H_{\mathcal{Y}}(\mathcal{X}) = -\frac{1}{2}[\alpha \log_2 \alpha + (1-\alpha)\log_2(1-\alpha) + (1-\alpha)\log_2(1-\alpha) + \alpha \log_2 \alpha], \tag{7.30}$$

so that

$$C = h(\mathcal{X}) - H_{\mathcal{Y}}(\mathcal{X}) = 1 + \alpha \log_2 \alpha + (1-\alpha)\log_2(1-\alpha). \tag{7.31}$$

Note that

$$H(\alpha) = -[\alpha \log_2 \alpha + (1-\alpha)\log_2(1-\alpha)] \geq 0, \tag{7.32}$$

so that $C \leq 1$. Moreover, if $\alpha = 1/2$, so that we have no idea whether each symbol coming out equals the one put in or not, the channel capacity is 0. On the other hand, the capacity is the maximum of 1 *both* when $\alpha = 0$ and when $\alpha = 1$: when the symbol out is always different from the symbol in, we can still recover the input message perfectly well.

7.7 Coding for error protection

Shannon's Channel Coding Theorem establishes the existence of good error-correcting codes by a random coding argument and so does not tell us how to construct such good sets of codewords in practice. Finding methods to construct good codes, especially ones that are easy to implement, continues as an active area of research, some of it involving very fancy modern mathematics, with results which have immediate impact in modern society.

We look here at an early example of a clever error-correcting coding system, the *Hamming codes*. For each $n = 2, 3, \ldots$ there is a Hamming code consisting of binary strings of length $2^n - 1$. The first $2^n - n - 1$ binary digits are *data bits*, and the final n are *check bits*. (We relax here the distinction between "binary digits" and "bits".) For example, consider the case $n = 3$, which gives us strings of length 7 with 4 data bits, thus called the *Hamming* $(7,4)$ *code*. Each data string of length 4 (and there are $2^4 = 16$ of them) $a = a_1a_2a_3a_4$ is assigned a string $c = c(a) = c_1c_2c_3$ of 3 check bits according to the following rule, in which all the additions are modulo 2:

$$\begin{aligned} c_1 &= a_1 + a_2 + a_3 \\ c_2 &= a_1 + a_3 + a_4 \\ c_3 &= a_2 + a_3 + a_4. \end{aligned} \tag{7.33}$$

Let's think for a moment about the strategy behind making a set of codewords. We have some data that we want to send across a noisy channel. We will assign each of our data symbols (or n-blocks, possibly the result of a preliminary source coding for data compression) to a different codeword to be put into the channel. We want our codewords to be somehow stable when they go across the channel, so that when we look at what comes out of the channel we will have a pretty good idea of what went in. The channel may change some of the digits in each codeword, but we want the code to be such that it

will be very unlikely that two codewords will change to the same string. A natural desideratum, then, is that every codeword should differ from every other codeword in a large number of places. There is an associated concept of distance, called the *Hamming distance*: if u and v are binary strings of length n, we define the Hamming distance between them to be the number of places at which they disagree:

$$d_H(u,v) = \sum_{i=1}^{n} |u_i - v_i| = \sum_{i=1}^{n} [(u_i + v_i) \mod 2]. \tag{7.34}$$

(Note the convenience of arithmetic modulo 2: adding detects when two entries disagree.) Thus we look for codes with many codewords, all at large Hamming distance from one another. And of course we want them also to be short if possible. Visualizing the codewords as points in a space that are well spread out may be helpful.

Note that the Hamming code given above is *linear*: the modulo 2 sum of two codewords is again a codeword. This is because the check bits are just modulo 2 sums of certain data bits, so that if a data word a has a check word $c(a)$ added on its end, then $c(a+a') \equiv c(a) + c(a') \mod 2$; thus if $ac(a)$ and $a'c(a')$ are codewords, so is their (entrywise modulo 2) sum, since it equals $(a+a')c(a+a')$.

Of course the sum of a codeword with itself is the word of all 0's, which we denote by $\overline{0}$. Then, with addition modulo 2,

$$d_H(u,v) = \sum_{i=1}^{n} (u_i + v_i) = d_H(u+v, \overline{0}). \tag{7.35}$$

Thus the *minimum distance* of the code (the minimum distance between any two codewords) is the same as the minimum distance from any nonzero codeword to $\overline{0}$, and this is the same as the *weight* t of the code, which is the minimum number of 1's in any nonzero codeword.

If we want to be able to *correct* E errors in the digits of any codeword due to noise in the channel, we will want to use a code whose minimum distance is $2E+1$: then changing no more than E digits in codewords will still keep them distinguishable. When we look at a word put out by the channel, we will interpret is as the codeword that is closest to it in terms of the Hamming distance. For the Hamming $(7,4)$ code given above, the weight, and thus the minimum distance, is 3, so we can correct 1 error in transmission, but not 2. Note that this includes errors in transmission of either data or check bits.

With the $(7,4)$ code we can *detect* up to two errors in transmission of any codeword, since it requires changing at least 3 entries to convert any codeword to another codeword. In general, a code with minimum distance t will allow *detection* of up to $t-1$ errors, and *correction* of up to $\lfloor (t-1)/2 \rfloor$ errors. (Recall that $\lfloor x \rfloor$ means the greatest integer that is less than or equal to x.)

7.8 Comments on some related terms and concepts

Shannon's brilliant approach handled information in a clean, mathematical, and useful way by avoiding murky and complicated issues about meaning, usefulness, cost in terms of money or energy, and so on. In this section we comment on several associated terms or concepts in a brief and incomplete manner that is adapted to our current context.

7.8.1 Information

We have seen that information in Shannon's sense is measured by the minimal number of binary digits needed to convey it. Also, the amount of information gained equals the amount of uncertainty removed. But what *is* information really? Is there no hope of pinning it down?

We are at no more of a loss here than we are in trying to say exactly what is mass, or force. As explained in my freshman physics class by John Wheeler, a physical law, such as Newton's Second Law $F = ma$, can function first as a definition and then as a law for making predictions. One can measure

a force, such as the propulsion due to a jet engine, on an object of a certain mass by observing the acceleration produced. Then we can be sure that if the object is cut in half the acceleration will be doubled. In the same way, information content is defined by comparison with an agreed on standard, coding by means of binary digits.

7.8.2 Language

We can define a language as *a tool for recording or transmitting information by means of physically realizable symbols*, thereby including spoken languages, chemical or microbiological signaling systems, electronic or optical information systems, animal calls, and so on. In Section 7.1 we defined a (formal) language to be any set of finite strings (words) on a finite alphabet. Any language can be studied from a mathematical viewpoint if it can be encoded (maybe with some information loss) by means of strings on a finite alphabet to produce a formal language.

7.8.3 Knowledge

We can define knowledge to be *recorded information*. Note that knowledge can be unconscious, nonverbal, direct: human language is not needed in order to have knowledge. Infants, human or animal, can recognize their (supposed) parents, and salmon can (chemically) recognize the streams where they were born. We can know how to swim, ride a bicycle, or hit a topspin backhand without ever involving any human language, even in thought. But in all of these examples, the relevant information is recorded somewhere in the form of stored physical symbols of some kind.

7.8.4 Symbol

7.8.4.1

Thus far we have used the term "symbol" for a member of a finite alphabet which may be used to form finite strings (words) or infinite strings (messages).

7.8.4.2

The word "symbol" can also be used to mean anything that stands for or represents something else. Here are several examples:

a rose stands for love

a crown stands for the monarchy

the presidential seal stands for the office of president

a cross stands for the Christian faith

a crescent stands for the Muslim faith

in Stendahl's novel, red stands for the military and black for the clergy

3 stands for the family of all sets that can be put in one-to-one correspondence with $\{0, 1, 2\}$

in a Wagner opera, a musical phrase can be a "leitmotif" that stands for a curse that is driving the plot

7.8.5 Meaning

The meaning of a symbol (or collection of symbols) could be the (possibly mental) object or objects behind it, along with their interrelationships. For example, in the movie *Enigma* the phrase "Mary Jane Hawkins" stands for a sudden insight—as meaning suddenly emerges. Meanings can reside in the associations or resonances that a symbol has with other symbols as well as in its history. Thus the meaning of a word can be indicated by giving partial synonyms and antonyms as well as its etymology.

7.8.6 Ambiguity

Ambiguity occurs when a symbol or collection of symbols has several possible meanings. Ambiguity adds richness, depth, and interest to literature—also to music, when reference of a musical phrase to an underlying key is uncertain. In the movie *Enigma* there is ambiguity about the intentions and motivations of the main characters. An analogue occurs in science, when several theories compete to explain the same observed data.

7.8.7 Theory

A theory is a relatively short collection of general principles that explains many particular similar phenomena (data). Slightly paraphrased, a theory is a simple, universal explanation for a complicated collection of contingent data. Important examples are Newton's theory of gravity, Darwin's theory of evolution and natural selection, and Einstein's theory of special relativity and theory of general relativity.

7.8.8 Understanding

Understanding consists of the formation, verification, and application of an effective theory.

References

J. R. Pierce, *Symbols, Signals and Noise*, Harper Torchbooks, New York, 1961.