Preface to the First Edition

We are to admit no more causes of natural things (as we are told by Newton) than such as are both true and sufficient to explain their appearances. This central theme is basic to the pursuit of science, and goes back to the principle known as Occam's razor: "if presented with a choice between indifferent alternatives, then one ought to select the simplest one." Unconsciously or explicitly, informal applications of this principle in science and mathematics abound.

The conglomerate of different research threads drawing on an objective and absolute form of this approach appears to be part of a single emerging discipline, which will become a major applied science like information theory or probability theory. We aim at providing a unified and comprehensive introduction to the central ideas and applications of this discipline.

Intuitively, the amount of information in a finite string is the size (number of binary digits, or *bits*) of the shortest program that without additional data, computes the string and terminates. A similar definition can be given for infinite strings, but in this case the program produces element after element forever. Thus, a long sequence of 1's such as

$$\underbrace{11111\ldots1}_{10{,}000 \text{ times}}$$

contains little information because a program of size about $\log 10,000$ bits outputs it:

```
for i := 1 to 10,000
print 1
```

Likewise, the transcendental number $\pi = 3.1415\ldots$, an infinite sequence of seemingly "random" decimal digits, contains but a few bits of information. (There is a short program that produces the consecutive digits of π forever.) Such a definition would appear to make the amount of information in a string (or other object) depend on the particular programming language used.

Fortunately, it can be shown that all reasonable choices of programming languages lead to quantification of the amount of "absolute" information in individual objects that is invariant up to an additive constant. We call this quantity the "Kolmogorov complexity" of the object. If an object contains regularities, then it has a shorter description than itself. We call such an object "compressible."

The application of Kolmogorov complexity takes a variety of forms, for example, using the fact that some strings are extremely compressible; using the compressibility of strings as a selection criterion; using the fact that many strings are not compressible at all; and using the fact that

some strings may be compressed, but that it takes a lot of effort to do so.

The theory dealing with the quantity of information in individual objects goes by names such as "algorithmic information theory," "Kolmogorov complexity," "K-complexity," "Kolmogorov-Chaitin randomness," "algorithmic complexity," "stochastic complexity," "descriptional complexity," "minimum description length," "program-size complexity," and others. Each such name may represent a variation of the basic underlying idea or a different point of departure. The mathematical formulation in each case tends to reflect the particular traditions of the field that gave birth to it, be it probability theory, information theory, theory of computing, statistics, or artificial intelligence.

This raises the question about the proper name for the area. Although there is a good case to be made for each of the alternatives listed above, and a name like "Solomonoff-Kolmogorov-Chaitin complexity" would give proper credit to the inventors, we regard "Kolmogorov complexity" as well entrenched and commonly understood, and we shall use it hereafter.

The mathematical theory of Kolmogorov complexity contains deep and sophisticated mathematics. Yet one needs to know only a small amount of this mathematics to apply the notions fruitfully in widely divergent areas, from sorting algorithms to combinatorial theory, and from inductive reasoning and machine learning to dissipationless computing.

Formal knowledge of basic principles does not necessarily imply the wherewithal to apply it, perhaps especially so in the case of Kolmogorov complexity. It is our purpose to develop the theory in detail and outline a wide range of illustrative applications. In fact, while the pure theory of the subject will have its appeal to the select few, the surprisingly large field of its applications will, we hope, delight the multitude.

The mathematical theory of Kolmogorov complexity is treated in Chapters 2, 3, and 4; the applications are treated in Chapters 5 through 8. Chapter 1 can be skipped by the reader who wants to proceed immediately to the technicalities. Section 1.1 is meant as a leisurely, informal introduction and peek at the contents of the book. The remainder of Chapter 1 is a compilation of material on diverse notations and disciplines drawn upon.

We define mathematical notions and establish uniform notation to be used throughout. In some cases we choose nonstandard notation since the standard one is homonymous. For instance, the notions "absolute value," "cardinality of a set," and "length of a string," are commonly denoted in the same way as $|\cdot|$. We choose distinguishing notations $|\cdot|$, $d(\cdot)$, and $l(\cdot)$, respectively.

Briefly, we review the basic elements of computability theory and probability theory that are required. Finally, in order to place the subject in the appropriate historical and conceptual context we trace the main roots of Kolmogorov complexity.

This way the stage is set for Chapters 2 and 3, where we introduce the notion of optimal effective descriptions of objects. The length of such a description (or the number of bits of information in it) is its Kolmogorov complexity. We treat all aspects of the elementary mathematical theory of Kolmogorov complexity. This body of knowledge may be called *algorithmic complexity theory*. The theory of Martin-Löf tests for randomness of finite objects and infinite sequences is inextricably intertwined with the theory of Kolmogorov complexity and is completely treated. We also investigate the statistical properties of finite strings with high Kolmogorov complexity. Both of these topics are eminently useful in the applications part of the book. We also investigate the recursion-theoretic properties of Kolmogorov complexity (relations with Gödel's incompleteness result), and the Kolmogorov complexity version of information theory, which we may call "algorithmic information theory" or "absolute information theory."

The treatment of *algorithmic probability theory* in Chapter 4 presupposes Sections 1.6, 1.11.2, and Chapter 3 (at least Sections 3.1 through 3.4). Just as Chapters 2 and 3 deal with the optimal effective description length of objects, we now turn to optimal (greatest) effective probability of objects. We treat the elementary mathematical theory in detail. Subsequently, we develop the theory of effective randomness tests under arbitrary recursive distributions for both finite and infinite sequences. This leads to several classes of randomness tests, each of which has a universal randomness test. This is the basis for the treatment of a mathematical theory of inductive reasoning in Chapter 5 and the theory of algorithmic entropy in Chapter 8.

Chapter 5 develops a general theory of inductive reasoning and applies the developed notions to particular problems of inductive inference, prediction, mistake bounds, computational learning theory, and minimum description length induction in statistics. This development can be viewed both as a resolution of certain problems in philosophy about the concept and feasibility of induction (and the ambiguous notion of "Occam's razor"), as well as a mathematical theory underlying computational machine learning and statistical reasoning.

Chapter 6 introduces the incompressibility method. Its utility is demonstrated in a plethora of examples of proving mathematical and computational results. Examples include combinatorial properties, the time complexity of computations, the average-case analysis of algorithms such as Heapsort, language recognition, string matching, "pumping lemmas"

in formal language theory, lower bounds in parallel computation, and Turing machine complexity. Chapter 6 assumes only the most basic notions and facts of Sections 2.1, 2.2, 3.1, 3.3.

Some parts of the treatment of resource-bounded Kolmogorov complexity and its many applications in computational complexity theory in Chapter 7 presuppose familiarity with a first-year graduate theory course in computer science or basic understanding of the material in Section 1.7.4. Sections 7.5 and 7.7 on "universal optimal search" and "logical depth" only require material covered in this book. The section on "logical depth" is technical and can be viewed as a mathematical basis with which to study the emergence of life-like phenomena—thus forming a bridge to Chapter 8, which deals with applications of Kolmogorov complexity to relations between physics and computation.

Chapter 8 presupposes parts of Chapters 2, 3, 4, the basics of information theory as given in Section 1.11, and some familiarity with college physics. It treats physical theories like dissipationless reversible computing, information distance and picture similarity, thermodynamics of computation, statistical thermodynamics, entropy, and chaos from a Kolmogorov complexity point of view. At the end of the book there is a comprehensive listing of the literature on theory and applications of Kolmogorov complexity and a detailed index.

How to Use This Book

The technical content of this book consists of four layers. The main text is the first layer. The second layer consists of examples in the main text. These elaborate the theory developed from the main theorems. The third layer consists of nonindented, smaller-font paragraphs interspersed with the main text. The purpose of such paragraphs is to have an explanatory aside, to raise some technical issues that are important but would distract attention from the main narrative, or to point to alternative or related technical issues. Much of the technical content of the literature on Kolmogorov complexity and related issues appears in the fourth layer, the exercises. When the idea behind a nontrivial exercise is not our own, we have tried to give credit to the person who originated the idea. Corresponding references to the literature are usually given in comments to an exercise or in the historical section of that chapter.

Starred sections are not really required for the understanding of the sequel and should be omitted at first reading. The application sections are not starred. The exercises are grouped together at the end of main sections. Each group relates to the material in between it and the previous group. Each chapter is concluded by an extensive historical section with full references. For convenience, all references in the text to the Kolmogorov complexity literature and other relevant literature are given in full were they occur. The book concludes with a References section intended as a separate exhaustive listing of the literature restricted to the

theory and the direct applications of Kolmogorov complexity. There are reference items that do not occur in the text and text references that do not occur in the References. We added a very detailed index combining the index to notation, the name index, and the concept index. The page number where a notion is defined first is printed in boldface. The initial part of the Index is an index to notation. Names as "J. von Neumann" are indexed European style "Neumann, J. von."

The exercises are sometimes trivial, sometimes genuine exercises, but more often compilations of entire research papers or even well-known open problems. There are good arguments to include both: the easy and real exercises to let the student exercise his comprehension of the material in the main text; the contents of research papers to have a comprehensive coverage of the field in a small number of pages; and research problems to show where the field is (or could be) heading. To save the reader the problem of having to determine which is which: "I found this simple exercise in number theory that looked like Pythagoras's Theorem. Seems difficult." "Oh, that is Fermat's Last Theorem; it was unsolved for three hundred and fifty years...," we have adopted the system of *rating numbers* used by D.E. Knuth [*The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, Addison-Wesley, 1973 (2nd Edition), pp. xvii–xix]. The interpretation is as follows:

00 A very easy exercise that can be answered immediately, from the top of your head, if the material in the text is understood.

10 A simple problem to exercise understanding of the text. Use fifteen minutes to think, and possibly pencil and paper.

20 An average problem to test basic understanding of the text and may take one or two hours to answer completely.

30 A moderately difficult or complex problem taking perhaps several hours to a day to solve satisfactorily.

40 A quite difficult or lengthy problem, suitable for a term project, often a significant result in the research literature. We would expect a very bright student or researcher to be able to solve the problem in a reasonable amount of time, but the solution is not trivial.

50 A research problem that, to the authors' knowledge, is open at the time of writing. If the reader has found a solution, he is urged to write it up for publication; furthermore, the authors of this book would appreciate hearing about the solution as soon as possible (provided it is correct).

This scale is "logarithmic": a problem of rating 17 is a bit simpler than average. Problems with rating 50, subsequently solved, will appear in

a next edition of this book with rating 45. Rates are sometimes based on the use of solutions to earlier problems. The rating of an exercise is based on that of its most difficult item, but not on the number of items. Assigning accurate rating numbers is impossible—one man's meat is another man's poison—and our rating will differ from ratings by others.

An orthogonal rating "M" implies that the problem involves more mathematical concepts and motivation than is necessary for someone who is primarily interested in Kolmogorov complexity and applications. Exercises marked "HM" require the use of calculus or other higher mathematics not developed in this book. Some exercises are marked with "•"; and these are especially instructive or useful. Exercises marked "O" are problems that are, to our knowledge, unsolved at the time of writing. The rating of such exercises is based on our estimate of the difficulty of solving them. Obviously, such an estimate may be totally wrong.

Solutions to exercises, or references to the literature where such solutions can be found, appear in the "Comments" paragraph at the end of each exercise. Nobody is expected to be able to solve all exercises.

The material presented in this book draws on work that until now was available only in the form of advanced research publications, possibly not translated into English, or was unpublished. A large portion of the material is new. The book is appropriate for either a one- or a two-semester introductory course in departments of mathematics, computer science, physics, probability theory and statistics, artificial intelligence, cognitive science, and philosophy. Outlines of possible one-semester courses that can be taught using this book are presented below.

Fortunately, the field of descriptional complexity is fairly young and the basics can still be comprehensively covered. We have tried to the best of our abilities to read, digest, and verify the literature on the topics covered in this book. We have taken pains to establish correctly the history of the main ideas involved. We apologize to those who have been unintentionally slighted in the historical sections. Many people have generously and selflessly contributed to verify and correct drafts of this book. We thank them below and apologize to those we forgot. In a work of this scope and size there are bound to remain factual errors and incorrect attributions. We greatly appreciate notification of errors or any other comments the reader may have, preferably by email to kolmogorov@cwi.nl, in order that future editions may be corrected.

Acknowledgments

We thank Greg Chaitin, Péter Gács, Leonid Levin, and Ray Solomonoff for taking the time to tell us about the early history of our subject and for introducing us to many of its applications. Juris Hartmanis and Joel Seiferas initiated us into Kolmogorov complexity in various ways.

Many people gave substantial suggestions for examples and exercises, or pointed out errors in a draft version. Apart from the people already mentioned, these are, in alphabetical order, Eric Allender, Charles Bennett, Piotr Berman, Robert Black, Ron Book, Dany Breslauer, Harry Buhrman, Peter van Emde Boas, William Gasarch, Joe Halpern, Jan Heering, G. Hotz, Tao Jiang, Max Kanovich, Danny Krizanc, Evangelos Kranakis, Michiel van Lambalgen, Luc Longpré, Donald Loveland, Albert Meyer, Lambert Meertens, Ian Munro, Pekka Orponen, Ramamohan Paturi, Jorma Rissanen, Jeff Shallit, A.Kh. Shen', J. Laurie Snell, Th. Tsantilas, John Tromp, Vladimir Uspensky, N.K. Vereshchagin, Osamu Watanabe, and Yaacov Yesha. Apart from them, we thank the many students and colleagues who contributed to this book.

We especially thank Péter Gács for the extraordinary kindness of reading and commenting in detail on the entire manuscript, including the exercises. His expert advice and deep insight saved us from many pitfalls and misunderstandings. Piergiorgio Odifreddi carefully checked and commented on the first three chapters. Parts of the book have been tested in one-semester courses and seminars at the University of Amsterdam in 1988 and 1989, the University of Waterloo in 1989, Dartmouth College in 1990, the Universitat Polytècnica de Catalunya in Barcelona in 1991/1992, the University of California at Santa Barbara, Johns Hopkins University, and Boston University in 1992/1993.

This document has been prepared using the LaTeX system. We thank Donald Knuth for TeX, Leslie Lamport for LaTeX, and Jan van der Steen at CWI for online help. Some figures were prepared by John Tromp using the xpic program.

The London Mathematical Society kindly gave permission to reproduce a long extract by A.M. Turing. The Indian Statistical Institute, through the editor of *Sankhyā*, kindly gave permission to quote A.N. Kolmogorov.

We gratefully acknowledge the financial support by NSF Grant DCR-8606366, ONR Grant N00014-85-k-0445, ARO Grant DAAL03-86-K-0171, the Natural Sciences and Engineering Research Council of Canada through operating grants OGP-0036747, OGP-046506, and International Scientific Exchange Awards ISE0046203, ISE0125663, and NWO Grant NF 62-376. The book was conceived in late Spring 1986 in the Valley of the Moon in Sonoma County, California. The actual writing lasted on and off from autumn 1987 until summer 1993.

One of us [PV] gives very special thanks to his lovely wife Pauline for insisting from the outset on the significance of this enterprise. The Aiken Computation Laboratory of Harvard University, Cambridge, Massachusetts, USA; the Computer Science Department of York University, Ontario, Canada; the Computer Science Department of the University

of Waterloo, Ontario, Canada; and CWI, Amsterdam, the Netherlands provided the working environments in which this book could be written.

Preface to the Second Edition

When this book was conceived ten years ago, few scientists realized the width of scope and the power for applicability of the central ideas. Partially because of the enthusiastic reception of the first edition, open problems have been solved and new applications have been developed. We have added new material on the relation between data compression and minimum description length induction, computational learning, and universal prediction; circuit theory; distributed algorithmics; instance complexity; CD compression; computational complexity; Kolmogorov random graphs; shortest encoding of routing tables in communication networks; computable universal distributions; average case properties; the equality of statistical entropy and expected Kolmogorov complexity; and so on. Apart from being used by researchers and as reference work, the book is now commonly used for graduate courses and seminars. In recognition of this fact, the second edition has been produced in textbook style. We have preserved as much as possible the ordering of the material as it was in the first edition. The many exercises bunched together at the ends of some chapters have been moved to the appropriate sections. The comprehensive bibliography on Kolmogorov complexity at the end of the book has been updated, as have the "History and References" sections of the chapters. Many readers were kind enough to express their appreciation for the first edition and to send notification of typos, errors, and comments. Their number is too large to thank them individually, so we thank them all collectively.

Outlines of One-Semester Courses

We have mapped out a number of one-semester courses on a variety of topics. These topics range from basic courses in theory and applications to special interest courses in learning theory, randomness, or information theory using the Kolmogorov complexity approach.

PREREQUISITES: Sections 1.1, 1.2, 1.7 (except Section 1.7.4).

I. Course on Basic Algorithmic Complexity and Applications

TYPE OF COMPLEXITY	THEORY	APPLICATIONS
plain complexity	2.1, 2.2, 2.3	4.4, Chapter 6
prefix complexity	1.11.2, 3.1 3.3, 3.4	5.1, 5.1.3, 5.2, 5.5 8.2, 8.3 8
resource-bounded complexity	7.1, 7.5, 7.7	7.2, 7.3, 7.6, 7.7

II. Course on Algorithmic Complexity

Type of Complexity	Basics	Randomness	Algorithmic Properties
state × symbol	1.12		
plain complexity	2.1, 2.2, 2.3	2.4	2.7
prefix complexity	1.11.2, 3.1 3.3, 3.4	3.5	3.7, 3.8
monotone complexity	4.5 (intro)	4.5.4	

III. Course on Algorithmic Randomness

Randomness Tests According to	Complexity Used	Finite Strings	Infinite Sequences
von Mises			1.9
Martin-Löf	2.1, 2.2	2.4	2.5
prefix complexity	1.11.2, 3.1, 3.3, 3.4	3.5	3.6, 4.5.6
general discrete	1.6 (intro), 4.3.1	4.3	
general continuous	1.6 (intro), 4.5 (intro), 4.5.1		4.5

IV. Course on Algorithmic Information Theory and Applications

Type of Complexity Used	Basics	Entropy	Symmetry of Information
classical information theory	1.11	1.11	1.11
plain complexity	2.1, 2.2	2.8	2.8
prefix complexity	3.1, 3.3, 3.4		3.8, 3.9.1
resource-bounded	7.1		Exercises 7.1.11 7.1.12
applications		8.1, 8.4, 8.5	Theorem 7.2.6 Exercise 6.10.15

V. Course on Algorithmic Probability Theory, Learning, Inference and Prediction

Theory	Basics	Universal Distribution	Applications to Inference
classical probability	1.6, 1.11.2		1.6
algorithmic complexity	2.1, 2.2, 2.3 3.1, 3.3, 3.4		8
algorithmic discrete probability	4.2, 4.1 4.3 (intro)	4.3.1, 4.3.2 4.3.3, 4.3.4, 4.3.6	
algorithmic contin. probability	4.5 (intro)	4.5.1, 4.5.2 4.5.4, 4.5.8	5.2
Solomonoff's inductive inference	5.1, 5.1.3, 5.2 5.4, 8	5.3, 5.4.3, 5.5 5.5.8	5.1.3

VI. Course on the Incompressibility Method

Chapter 2 (Sections 2.1, 2.2, 2.4, 2.6, 2.8), Chapter 3 (mainly Sections 3.1, 3.3), Section 4.4, and Chapters 6 and 7. The course covers the basics of the theory with many applications in proving upper and lower bounds on the running time and space use of algorithms.

VII. Course on Randomness, Information, and Physics

Course III and Chapter 8. In physics the applications of Kolmogorov complexity include theoretical illuminations of foundational issues. For example, the approximate equality of statistical entropy and expected Kolmogorov complexity, the nature of "entropy," a fundamental resolution of the "Maxwell's Demon" paradox. However, also more concrete applications like "information distance" and "thermodynamics of computation" are covered.

Contents

Preface to the First Edition v
How to Use This Book viii
Acknowledgments x
Preface to the Second Edition xii
Outlines of One-Semester Courses xii
List of Figures xix

1 Preliminaries **1**
1.1 A Brief Introduction 1
1.2 Prerequisites and Notation 6
1.3 Numbers and Combinatorics 8
1.4 Binary Strings 12
1.5 Asymptotic Notation 15
1.6 Basics of Probability Theory 18
1.7 Basics of Computability Theory 24
1.8 The Roots of Kolmogorov Complexity 47
1.9 Randomness 49
1.10 Prediction and Probability 59
1.11 Information Theory and Coding 65
1.12 State × Symbol Complexity 84
1.13 History and References 86

2 Algorithmic Complexity **93**

2.1 The Invariance Theorem 96
2.2 Incompressibility 108
2.3 C as an Integer Function 119
2.4 Random Finite Sequences 127
2.5 *Random Infinite Sequences 136
2.6 Statistical Properties of Finite Sequences 158
2.7 Algorithmic Properties of C 167
2.8 Algorithmic Information Theory 179
2.9 History and References 185

3 Algorithmic Prefix Complexity **189**

3.1 The Invariance Theorem 192
3.2 *Sizes of the Constants 197
3.3 Incompressibility 202
3.4 K as an Integer Function 206
3.5 Random Finite Sequences 208
3.6 *Random Infinite Sequences 211
3.7 Algorithmic Properties of K 224
3.8 *Complexity of Complexity 226
3.9 *Symmetry of Algorithmic Information 229
3.10 History and References 237

4 Algorithmic Probability **239**

4.1 Enumerable Functions Revisited 240
4.2 Nonclassical Notation of Measures 242
4.3 Discrete Sample Space 245
4.4 Universal Average-Case Complexity 268
4.5 Continuous Sample Space 272
4.6 Universal Average-Case Complexity, Continued 307
4.7 History and References 307

5 Inductive Reasoning **315**

5.1 Introduction 315
5.2 Solomonoff's Theory of Prediction 324
5.3 Universal Recursion Induction 335
5.4 Simple Pac-Learning 339
5.5 Hypothesis Identification by Minimum Description Length 351
5.6 History and References 372

6 The Incompressibility Method **379**

6.1 Three Examples 380
6.2 High- Probability Properties 385
6.3 Combinatorics 389
6.4 Kolmogorov Random Graphs 396
6.5 Compact Routing 404
6.6 Average-Case Complexity of Heapsort 412
6.7 Longest Common Subsequence 417
6.8 Formal Language Theory 420
6.9 Online CFL Recognition 427
6.10 Turing Machine Time Complexity 432
6.11 Parallel Computation 445
6.12 Switching Lemma 449
6.13 History and References 452

7 Resource-Bounded Complexity **459**

7.1 Mathematical Theory 460
7.2 Language Compression 476
7.3 Computational Complexity 488
7.4 Instance Complexity 495
7.5 *Kt* Complexity and Universal Optimal Search 502
7.6 Time-Limited Universal Distributions 506
7.7 Logical Depth 510
7.8 History and References 516

8 Physics, Information, and Computation **521**

8.1 Algorithmic Complexity and Shannon's Entropy 522

8.2 Reversible Computation 528

8.3 Information Distance 537

8.4 Thermodynamics 554

8.5 Entropy Revisited 565

8.6 Compression in Nature 583

8.7 History and References 586

References **591**

Index **618**

List of Figures

1.1 Turing machine 28
1.2 Inferred probability for increasing n 60
1.3 Binary tree for $E(1) = 0$, $E(2) = 10$, $E(3) = 110$, $E(4) = 111$ 72
1.4 Binary tree for $E(1) = 0$, $E(2) = 01$, $E(3) = 011$, $E(4) = 0111$ 73

2.1 The graph of the integer function $C(x)$ 121
2.2 The graph of the integer function $C(x|l(x))$ 123
2.3 Test of Example 2.4.1 128
2.4 Complexity oscillations of initial segments of infinite high-complexity sequences 139
2.5 Three notions of "chaotic" infinite sequences 148

3.1 The 425-bit universal combinator U' in pixels 201
3.2 The graphs of $K(x)$ and $K(x|l(x))$ 207
3.3 Complexity oscillations of a typical random sequence ω . 215
3.4 K-complexity criteria for randomness of infinite sequences 215
3.5 Complexity oscillations of Ω 216

4.1 Graph of $\mathbf{m}(x)$ with lower bound $1/x \cdot \log x \cdot \log\log x \cdots$ 249

4.2 Relations between five complexities 285

5.1 Trivial consistent automaton 317
5.2 Smallest consistent automaton 317
5.3 Sample data set 365
5.4 Imperfect decision tree 366
5.5 Perfect decision tree 367

6.1 Single-tape Turing machine 381
6.2 The two possible nni's on (u, v): swap $B \leftrightarrow C$ or $B \leftrightarrow D$ 416
6.3 The nni distance between (i) and (ii) is 2 416
6.4 Multitape Turing machine 428

8.1 Reversible Boolean gates 530
8.2 Implementing reversible AND gate and NOT gate 531
8.3 Controlling billiard ball movements 532
8.4 A billiard ball computer 533
8.5 Combining irreversible computations of y from x and x from y to achieve a reversible computation of y from x . 543
8.6 Reversible execution of concatenated programs for $(y|x)$ and $(z|y)$ to transform x into z 545
8.7 Carnot cycle 555
8.8 Heat engine 556
8.9 State space 559
8.10 Atomic spin in CuO_2 at low temperature 562
8.11 Regular "up" and "down" spins 563
8.12 Algorithmic entropy: left a random micro state, right a regular micro state 567
8.13 Szilard engine 568
8.14 Adiabatic demagnetization to achieve low temperature . 583
8.15 The maze: a binary tree constructed from matches 584
8.16 Time required for *Formica sanguinea* scouts to transmit information about the direction to the syrup to the forager ants 585

1

Preliminaries

1.1 A Brief Introduction

Suppose we want to describe a given object by a finite binary string. We do not care whether the object has many descriptions; however, each description should describe but one object. From among all descriptions of an object we can take the length of the shortest description as a measure of the object's complexity. It is natural to call an object "simple" if it has at least one short description, and to call it "complex" if all of its descriptions are long.

But now we are in danger of falling into the trap so eloquently described in the Richard-Berry paradox, where we define a natural number as "the least natural number that cannot be described in less than twenty words." If this number does exist, we have just described it in thirteen words, contradicting its definitional statement. If such a number does not exist, then all natural numbers can be described in fewer than twenty words. We need to look very carefully at the notion of "description."

Assume that each description describes at most one object. That is, there is a specification method D that associates at most one object x with a description y. This means that D is a function from the set of descriptions, say Y, into the set of objects, say X. It seems also reasonable to require that for each object x in X, there is a description y in Y such that $D(y) = x$. (Each object has a description.) To make descriptions useful we like them to be finite. This means that there are only countably many descriptions. Since there is a description for each object, there are also only countably many describable objects. How do we measure the complexity of descriptions?

Taking our cue from the theory of computation, we express descriptions as finite sequences of 0's and 1's. In communication technology, if the specification method D is known to both a sender and a receiver, then a message x can be transmitted from sender to receiver by transmitting the sequence of 0's and 1's of a description y with $D(y) = x$. The cost of this transmission is measured by the number of occurrences of 0's and 1's in y, that is, by the length of y. The least cost of transmission of x is given by the length of a shortest y such that $D(y) = x$. We choose this least cost of transmission as the descriptional complexity of x under specification method D.

Obviously, this descriptional complexity of x depends crucially on D. The general principle involved is that the syntactic framework of the description language determines the succinctness of description.

In order to objectively compare descriptional complexities of objects, to be able to say "x is more complex than z," the descriptional complexity of x should depend on x alone. This complexity can be viewed as related to a universal description method that is a priori assumed by all senders and receivers. This complexity is optimal if no other description method assigns a lower complexity to any object.

We are not really interested in optimality with respect to all description methods. For specifications to be useful at all it is necessary that the mapping from y to $D(y)$ can be executed in an effective manner. That is, it can at least in principle be performed by humans or machines. This notion has been formalized as that of "partial recursive functions." According to generally accepted mathematical viewpoints it coincides with the intuitive notion of effective computation.

The set of partial recursive functions contains an optimal function that minimizes description length of every other such function. We denote this function by D_0. Namely, for any other recursive function D, for all objects x, there is a description y of x under D_0 that is shorter than any description z of x under D. (That is, shorter up to an additive constant that is independent of x.) Complexity with respect to D_0 minorizes the complexities with respect to all partial recursive functions.

We identify the length of the description of x with respect to a fixed specification function D_0 with the "algorithmic (descriptional) complexity" of x. The optimality of D_0 in the sense above means that the complexity of an object x is invariant (up to an additive constant independent of x) under transition from one optimal specification function to another. Its complexity is an objective attribute of the described object alone: it is an intrinsic property of that object, and it does not depend on the description formalism. This complexity can be viewed as "absolute information content": the amount of information that needs to be transmitted between all senders and receivers when they communicate the message in

absence of any other a priori knowledge that restricts the domain of the message.

Broadly speaking, this means that all description syntaxes that are powerful enough to express the partial recursive functions are approximately equally succinct. All algorithms can be expressed in each such programming language equally succinctly, up to a fixed additive constant term. The remarkable usefulness and inherent rightness of the theory of Kolmogorov complexity stems from this independence of the description method.

Thus, we have outlined the program for a general theory of algorithmic complexity. The four major innovations are as follows:

1. In restricting ourselves to formally effective descriptions, our definition covers every form of description that is intuitively acceptable as being effective according to general viewpoints in mathematics and logic.

2. The restriction to effective descriptions entails that there is a universal description method that minorizes the description length or complexity with respect to any other effective description method. This would not be the case if we considered, say, all noneffective description methods. Significantly, this implies Item 3.

3. The description length or complexity of an object is an intrinsic attribute of the object independent of the particular description method or formalizations thereof.

4. The disturbing Richard-Berry paradox above does not disappear, but resurfaces in the form of an alternative approach to proving Kurt Gödel's (1906–1978) famous result that not every true mathematical statement is provable in mathematics.

Example 1.1.1 **(Gödel's incompleteness result)** A formal system (consisting of definitions, axioms, rules of inference) is *consistent* if no statement that can be expressed in the system can be proved to be both true and false in the system. A formal system is *sound* if only true statements can be proved to be true in the system. (Hence, a sound formal system is consistent.)

Let x be a finite binary string. We write "x is random" if the shortest binary description of x with respect to the optimal specification method D_0 has length at least x. A simple counting argument shows that there are random x's of each length.

Fix any sound formal system F in which we can express statements like "x is random." Suppose F can be described in f bits—assume, for example, that this is the number of bits used in the exhaustive description of

F in the first chapter of the textbook *Foundations of F*. We claim that for all but finitely many random strings x, the sentence "x is random" is not provable in F. Assume the contrary. Then given F, we can start to exhaustively search for a proof that some string of length $n \gg f$ is random, and print it when we find such a string x. This procedure to print x of length n uses only $\log n + f$ bits of data, which is much less than n. But x is random by the proof and the fact that F is sound. Hence, F is not consistent, which is a contradiction. ◇

This shows that although most strings are random, it is impossible to effectively prove them random. In a way, this explains why the incompressibility method in Chapter 6 is so successful. We can argue about a "typical" individual element, which is difficult or impossible by other methods.

Example 1.1.2 **(Lower bounds)** The secret of the successful use of descriptional complexity arguments as a proof technique is due to a simple fact: the overwhelming majority of strings have almost no computable regularities. We have called such a string "random." There is no shorter description of such a string than the literal description: it is incompressible. Incompressibility is a noneffective property in the sense of Example 1.1.1.

Traditional proofs often involve all instances of a problem in order to conclude that some property holds for at least one instance. The proof would be more simple, if only that one instance could have been used in the first place. Unfortunately, that instance is hard or impossible to find, and the proof has to involve all the instances. In contrast, in a proof by the incompressibility method, we first choose a random (that is, incompressible) individual object that is known to exist (even though we cannot construct it). Then we show that if the assumed property did not hold, then this object could be compressed, and hence it would not be random. Let us give a simple example.

A prime number is a natural number that is not divisible by natural numbers other than itself and 1. We prove that for infinitely many n, the number of primes less than or equal to n is at least $\log n / \log\log n$. The proof method is as follows. For each n, we construct a description from which n can be effectively retrieved. This description will involve the primes less than n. For some n this description must be long, which shall give the desired result.

Assume that $p_1, p_2, \ldots, p_m$ is the list of all the primes less than n. Then,

$$n = p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m}$$

can be reconstructed from the vector of the exponents. Each exponent is at most $\log n$ and can be represented by $\log\log n$ bits. The description of n (given $\log n$) can be given in $m \log\log n$ bits.

It can be shown that each n that is random (given $\log n$) cannot be described in fewer than $\log n$ bits, whence the result follows. Can we do better? This is slightly more complicated. Let $l(x)$ denote the length of the binary representation of x. We shall show that for infinitely many n of the form $n = m \log^2 m$, the number of distinct primes less than n is at least m.

Firstly, we can describe any given integer N by $E(m)N/p_m$, where $E(m)$ is a prefix-free encoding (page 71) of m, and p_m is the largest prime dividing N. For random N, the length of this description, $l(E(m)) + \log N - \log p_m$, must exceed $\log N$. Therefore, $\log p_m < l(E(m))$. It is known (and easy) that we can set $l(E(m)) \leq \log m + 2 \log\log m$. Hence, $p_m < m \log^2 m$. Setting $n := m \log^2 m$, and observing from our previous result that p_m must grow with N, we have proven our claim. The claim is equivalent to the statement that for our special sequence of values of n the number of primes less than n exceeds $n/\log^2 n$. The idea of connecting primality and prefix code-word length is due to P. Berman, and the present proof is due to J. Tromp.

Chapter 6 introduces the incompressibility method. Its utility is demonstrated in a variety of examples of proving mathematical and computational results. These include questions concerning the average case analysis of algorithms (such as Heapsort), sequence analysis, average case complexity in general, formal languages, combinatorics, time and space complexity analysis of various sequential or parallel machine models, language recognition, and string matching. Other topics like the use of resource-bounded Kolmogorov complexity in the analysis of computational complexity classes, the universal optimal search algorithm, and "logical depth" are treated in Chapter 7. ◇

Example 1.1.3 **(Prediction)** We are given an initial segment of an infinite sequence of zeros and ones. Our task is to predict the next element in the sequence: zero or one? The set of possible sequences we are dealing with constitutes the "sample space"; in this case, the set of one-way infinite binary sequences. We assume some probability distribution μ over the sample space, where $\mu(x)$ is the probability of the initial segment of a sequence being x. Then the probability of the next bit being "0," after an initial segment x, is clearly $\mu(0|x) = \mu(x0)/\mu(x)$. This problem constitutes, perhaps, the central task of inductive reasoning and artificial intelligence. However, the problem of induction is that in general we do not know the distribution μ, preventing us from assessing the actual probability. Hence, we have to use an estimate.

Now assume that μ is computable. (This is not very restrictive, since any distribution used in statistics is computable, provided the parameters are computable.) We can use Kolmogorov complexity to give a very good

estimate of μ. This involves the so-called "universal distribution" $\mathbf{M}$. Roughly speaking, $\mathbf{M}(x)$ is close to 2^{-l}, where l is the length in bits of the shortest effective description of x. Among other things, $\mathbf{M}$ has the property that it assigns at least as high a probability to x as any computable μ (up to a multiplicative constant factor depending on μ but not on x). What is particularly important to prediction is the following:

Let S_n denote the μ-expectation of the square of the error we make in estimating the probability of the nth symbol by $\mathbf{M}$. Then it can be shown that the sum $\sum_n S_n$ is bounded by a constant. In other words, S_n converges to zero faster than $1/n$. Consequently, any actual (computable) distribution can be estimated and predicted with great accuracy using only the single universal distribution.

Chapter 5 develops a general theory of inductive reasoning and applies the notions introduced to particular problems of inductive inference, prediction, mistake bounds, computational learning theory, and minimum description length induction methods in statistics. In particular, it is demonstrated that data compression improves generalization and prediction performance. ◇

The purpose of the remainder of this chapter is to define several concepts we require, if not by way of introduction, then at least to establish notation.

1.2 Prerequisites and Notation

We usually deal with nonnegative integers, sets of nonnegative integers, and mappings from nonnegative integers to nonnegative integers. A, B, C, ... denote sets. $\mathcal{N}$, $\mathcal{Z}$, $\mathcal{Q}$, $\mathcal{R}$ denote the sets of nonnegative integers (natural numbers including zero), integers, rational numbers, and real numbers, respectively. For each such set A, by A^+ we denote the subset of A consisting of positive numbers.

We use the following set-theoretical notations. $x \in A$ means that x is a member of A. In $\{x : x \in A\}$, the symbol ":" denotes set formation. $A \bigcup B$ is the *union* of A and B, $A \bigcap B$ is the *intersection* of A and B, and $\bar{A}$ is the *complement* of A when the universe $A \bigcup \bar{A}$ is understood. $A \subseteq B$ means A is a *subset* of B. $A = B$ means A and B are *identical* as sets (have the same members).

The *cardinality* (or diameter) of a finite set A is the number of elements it contains and is denoted as $d(A)$. If $A = \{a_1, \ldots, a_n\}$, then $d(A) = n$. The *empty* set $\{\}$, with no elements in it, is denoted by $\emptyset$. In particular, $d(\emptyset) = 0$.

Given x and y, the ordered pair (x, y) consists of x and y in that order. $A \times B$ is the *Cartesian product* of A and B, the set $\{(x, y) : x \in A$ and

$y \in B\}$. The n-fold Cartesian product of A with itself is denoted as A^n. If $R \subseteq A^2$, then R is called a *binary relation*. The same definitions can be given for n-tuples, $n > 2$, and the corresponding relations are *n-ary*. We say that an n-ary relation R is *single-valued* if for every $(x_1, \ldots, x_{n-1})$ there is at most one y such that $(x_1, \ldots, x_{n-1}, y) \in R$. Consider the *domain* $\{(x_1, \ldots, x_{n-1}) :$ there is a y such that $(x_1, \ldots, x_{n-1}, y) \in R\}$ of a single-valued relation R. Clearly, a single-valued relation $R \subseteq A^{n-1} \times B$ can be considered as a mapping from its domain into B. Therefore, we also call a single-valued n-ary relation a *partial function* of $n-1$ variables ("partial" because the domain of R may not comprise all of A^{n-1}). We denote functions by $\phi, \psi, \ldots$ or $f, g, h, \ldots$ Functions defined on the n-fold Cartesian product A^n are denoted with possibly a superscript denoting the number of variables, like $\phi^{(n)} = \phi^{(n)}(x_1, \ldots, x_n)$.

We use the notation $\langle \cdot \rangle$ for some standard one-to-one encoding of $\mathcal{N}^n$ into $\mathcal{N}$. We will use $\langle \cdot \rangle$ especially as a *pairing function* over $\mathcal{N}$ to associate a unique natural number $\langle x, y \rangle$ with each pair (x, y) of natural numbers. An example is $\langle x, y \rangle$ defined by $y + (x + y + 1)(x + y)/2$. This mapping can be used recursively: $\langle x, y, z \rangle = \langle x, \langle y, z \rangle \rangle$.

If ϕ is a *partial function* from A to B, then for each $x \in A$ either $\phi(x) \in B$ or $\phi(x)$ is undefined. If x is a member of the domain of ϕ, then $\phi(x)$ is called a *value* of ϕ, and we write $\phi(x) < \infty$ and ϕ is called *convergent* or *defined* at x; otherwise we write $\phi(x) = \infty$ and we call ϕ *divergent* or *undefined* at x. The set of values of ϕ is called the *range* of ϕ. If ϕ converges at every member of A, it is a *total function*, otherwise a *strictly partial function*. If each member of a set B is also a value of ϕ, then ϕ is said to *map onto* B, otherwise to *map into* B. If for each pair x and y, $x \neq y$, for which ϕ converges $\phi(x) \neq \phi(y)$ holds, then ϕ is a *one-to-one mapping*, otherwise a *many-to-one mapping*. The function $f : A \to \{0, 1\}$ defined by $f(x) = 1$ if $\phi(x)$ converges, and $f(x) = 0$ otherwise, is called the *characteristic function* of the domain of ϕ.

If ϕ and ψ are two partial functions, then $\psi\phi$ (equivalently, $\psi(\phi(x))$) denotes their *composition*, the function defined by $\{(x, y) :$ there is a z such that $\phi(x) = z$ and $\psi(z) = y\}$. The *inverse* ϕ^{-1} of a one-to-one partial function ϕ is defined by $\phi^{-1}(y) = x$ iff $\phi(x) = y$.

A set A is called *countable* if it is either empty or there is a total one-to-one mapping from A to the natural numbers $\mathcal{N}$. We say A is *countably infinite* if it is both countable and infinite. By 2^A we denote the *set of all subsets* of A. The set $2^{\mathcal{N}}$ has the cardinality of the *continuum* and is therefore uncountably infinite.

For binary relations, we use the terms *reflexive*, *transitive*, *symmetric, equivalence, partial order,* and *linear* (or *total*) *order* in the usual meaning. Partial orders can be *strict* ($<$) or *nonstrict* ($\leq$).

If we use the logarithm notation $\log x$ without subscript, then we shall always mean base 2. By $\ln x$ we mean the *natural logarithm* $\log_e x$, where $e = 2.71\ldots$.

We use the quantifiers $\exists$ ("there exists"), $\forall$ ("for all"), $\exists^\infty$ ("there exist infinitely many"), and the awkward $\forall^\infty$ ("for all but finitely many"). This way, $\forall^\infty x[\phi(x)]$ iff $\neg\exists^\infty x[\neg\phi(x)]$.

1.3 Numbers and Combinatorics

The *absolute value* of a real number r is denoted by $|r|$ and is defined as $|r| = -r$ if $r < 0$ and r otherwise. The *floor* of a real number r, denoted by $\lfloor r \rfloor$, is the greatest integer n such that $n \leq r$. Analogously, the *ceiling* of a real number r, denoted by $\lceil r \rceil$, is the least integer n such that $n \geq r$.

Example 1.3.1 $|-1| = |1| = 1$. $\lfloor 0.5 \rfloor = 0$ and $\lceil 0.5 \rceil = 1$. Analogously, $\lfloor -0.5 \rfloor = -1$ and $\lceil -0.5 \rceil = 0$. But $\lfloor 2 \rfloor = \lceil 2 \rceil = 2$ and $\lfloor -2 \rfloor = \lceil -2 \rceil = -2$. $\diamond$

A *permutation* of n objects is an arrangement of n distinct objects in an ordered sequence. For example, the six different permutations of objects a, b, c are

$$abc, acb, bac, bca, cab, cba.$$

The number of permutations of n objects is found most easily by imagining a sequential process to choose a permutation. There are n choices of which object to place in the first position; after filling the first position there remain $n-1$ objects and therefore $n-1$ choices of which object to place in the second position, and so on. Therefore, the number of permutations of n objects is $n \times (n-1) \times \cdots \times 2 \times 1$, denoted by $n!$ and is referred to as n *factorial*. In particular, $0! = 1$.

A *variation* of k out of n objects is an arrangement consisting of the first k elements of a permutation of n objects. For example, the twelve variations of two out of four objects a, b, c, d are

$$ab, ac, ad, ba, bc, bd, ca, cb, cd, da, db, dc.$$

The number of variations of k out of n is $n!/(n-k)!$, as follows by the previous argument. While there is no accepted standard notation, we denote the number of variations as $(n)_k$. In particular, $(n)_0 = 1$.

The *combinations* of n objects taken k at a time ("n choose k") are the possible choices of k different elements from a collection of n objects. The six different combinations of two out of four objects a, b, c, d are

$$\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}.$$

We can consider a combination as a variation in which the order does not count. We have seen that there are $n(n-1)\cdots(n-k+1)$ ways to choose the first k elements of a permutation. Every k-combination appears precisely $k!$ times in these arrangements, since each combination occurs in all its permutations. Therefore, the number of combinations, denoted by $\binom{n}{k}$, is

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots(1)}.$$

In particular, $\binom{n}{0} = 1$. The quantity $\binom{n}{k}$ is also called a *binomial coefficient*. It has an extraordinary number of applications. Perhaps the foremost relation associated with it is the Binomial Theorem, discovered in 1676 by Isaac Newton

$$(x+y)^n = \sum_k \binom{n}{k} x^k y^{n-k},$$

with n a positive integer. Note that in the summation k need not be restricted to $0 \le k \le n$, but can range over $-\infty < k < +\infty$, since for $k < 0$ or $k > n$ the terms are all zero.

Example 1.3.2 An important relation following from the Binomial Theorem is found by substituting $y = 1$:

$$(x+1)^n = \sum_k \binom{n}{k} x^k.$$

Substituting also $x = 1$ we find

$$2^n = \sum_k \binom{n}{k}.$$

◇

Exercises

1.3.1. [13] A "stock" of bridge cards consists of four suits and thirteen face values, respectively. Each card is defined by its suit and face value.

(a) How many cards are there?

(b) Each player gets a "hand" consisting of thirteen cards. How many different hands are there?

(c) What is the probability of getting a full suit as a hand? Assume this is the probability of obtaining a full suit when drawing thirteen cards, successively without replacement, from a full stock.

Comments. (a) $4 \times 13 = 52$. (b) $\binom{52}{13} = 635013559600$. (c) $4/\binom{52}{13}$.

1.3.2. [12] Consider a random distribution of k distinguishable balls in n cells, that is, each of the n^k possible arrangements has probability n^{-k}. Show that the probability P_i that a specified cell contains exactly i balls $(0 \le i \le k)$ is given by $P_i = \binom{k}{i}(1/n)^i(1-1/n)^{k-i}$.

Comments. Source: W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, Wiley, 1968.

1.3.3. [08] Show that $\binom{n}{k} = \frac{(n)_k}{k!}$ and $\binom{n}{k} = \binom{n}{n-k}$.

1.3.4. [M34] Prove the following identity, which is very useful in the sequel of this book.

$$\log \binom{n}{k} = k \log \frac{n}{k} + (n-k) \log \frac{n}{n-k} + \frac{1}{2} \log \frac{n}{k(n-k)} + O(1).$$

1.3.5. [15] (a) Prove that the number of ways n distinguishable balls can be placed in k numbered cells such that the first cell contains n_1 balls, the second cell n_2 balls, up to the kth cell contains n_k balls with $n_1 + \cdots + n_k = n$ is

$$\binom{n}{n_1, \ldots, n_k} = \frac{n!}{n_1! \cdots n_k!}.$$

This number is called a *multinomial coefficient*. Note that the order of the cells is essential in that the partitions $(n_1 = 1, n_2 = 2)$ and $(n_1 = 2, n_2 = 1)$ are different. The order of the elements within a cell is irrelevant.

(b) Show that

$$(x_1 + \cdots + x_k)^n = \sum \binom{n}{n_1, \ldots, n_k} x_1^{n_1} \cdots x_k^{n_k},$$

with the sum taken for all $n_1 + \cdots + n_k = n$.

(c) The *number of ordered different partitions* of n in r nonnegative integral summands is denoted by $A_{n,r}$. Compute $A_{n,r}$ in the form of a binomial coefficient.

Comments. $(1,0)$ and $(0,1)$ are different partitions, so $A_{1,2} = 2$. Source: W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, Wiley, 1968.

1.3.6. [14] Define the *occupancy numbers* for n balls distributed over k cells as a k-tuple of integers $(n_1, n_2, \ldots, n_k)$ satisfying $n_1 + n_2 + \cdots + n_k = n$ with $n_i \ge 0$ $(1 \le i \le k)$. That is, the first cell contains n_1 balls, the second cell n_2 balls, and so on.

(a) Show that there are $\binom{n}{n_1,\ldots,n_k}$ placements of n balls in k cells resulting in the numbers $(n_1,\ldots,n_k)$.

(b) There are k^n possible placements of n balls in k cells altogether. Compute the fraction that results in the given occupancy numbers.

(c) Assume that all k^n possible placements of n balls in k cells are equally probable. Conclude that the probability of obtaining the given occupancy numbers is

$$\frac{n!}{n_1!\cdots n_k!}k^{-n}.$$

Comments. In physics this is known as the *Maxwell-Boltzmann statistics* (here "statistics" is used as a synonym to "distribution"). Source: W. Feller, *Ibid.*

1.3.7. [15] We continue with the previous Exercise. In physical situations the assumption of equiprobability of possible placements seems unavoidable, for example, molecules in a volume of gas divided into (hypothetical) cells of equal volume. Numerous attempts have been made to prove that physical particles behave in accordance with the Maxwell-Boltzmann distribution. However, it has been shown conclusively that *no known particles* behave according to this distribution.

(a) In the *Bose-Einstein* distribution we count only *distinguishable* distributions of n balls over k cells without regard for the identities of the balls. We are only interested in the number of solutions of $n_1+n_2+\cdots+n_k=n$. Show that this number is $A_{n,k}=\binom{k+n-1}{n}=\binom{k+n-1}{k-1}$. Conclude that the probability of obtaining each given occupancy number is equally $1/A_{n,k}$. (Illustration: the distinguishable distributions of two balls over two cells are |**, *|*, and **|. Hence, according to Bose-Einstein statistics there are only three possible outcomes for two coin flips: head-head, head-tail, and tail-tail, and each outcome has equal probability $\frac{1}{3}$.)

(b) In the *Fermi-Dirac* distribution, (1) two or more particles cannot occupy the same cell and (2) all distinguishable arrangements satisfying (1) have the same probability. Note that (1) requires $n\le k$. Prove that in the Fermi-Dirac distribution there are in total $\binom{k}{n}$ possible arrangements. Conclude that the probability for each possible occupancy number is equally $1/\binom{k}{n}$.

Comments. According to modern physics, photons, nuclei, and atoms containing an even number of elementary particles behave according to model (a), and electrons, neutrons, and protons behave according to model (b). This shows that nature does not necessarily satisfy our a priori assumptions, however plausible they may be. Source: W. Feller, *Ibid.*

1.4 Binary Strings

We are concerned with *strings* over a nonempty set $\mathcal{B}$ of *basic elements*. Unless otherwise noted, we use $\mathcal{B} = \{0, 1\}$. Instead of "string" we also use "word" and "sequence," synonymously. The way we use it, "strings" and "words" are usually finite, while "sequences" are usually infinite. The set of all finite strings over $\mathcal{B}$ is denoted by $\mathcal{B}^*$, defined as

$$\mathcal{B}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\},$$

with ϵ denoting the *empty* string, with no letters. *Concatenation* is a binary operation on the elements of $\mathcal{B}^*$ that associates xy with each ordered pair of elements (x, y) in the Cartesian product $\mathcal{B}^* \times \mathcal{B}^*$. Clearly,

1. $\mathcal{B}^*$ is closed under the operation of concatenation; that is, if x and y are elements of $\mathcal{B}^*$, then so is xy;
2. concatenation is an associative operation on $\mathcal{B}^*$; that is, $(xy)z = x(yz) = xyz$; and
3. concatenation on $\mathcal{B}^*$ has the *unit* element ϵ; that is, $\epsilon x = x\epsilon = x$.

We now consider a *correspondence* of finite binary strings and natural numbers. The standard binary representation has the disadvantage that either some strings do not represent a natural number, or each natural number is represented by more than one string. For example, either "010" does not represent "2," or both "010" and "10" represent "2." However, we can map $\mathcal{B}^*$ one-to-one onto the natural numbers by associating each string with its index in the lexicographical ordering

$$(\epsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), \ldots. \tag{1.1}$$

This way we represent $x = 2^{n+1} - 1 + \sum_{i=0}^{n} a_i 2^i$ by $a_n \ldots a_1 a_0$. This is equivalent to $x = \sum_{i=0}^{n} b_i 2^i$ with $b_i \in \{1, 2\}$ and $b_i = a_i + 1$ for $0 \leq i \leq n$.

This way we have a binary representation for the natural numbers that is different from the standard binary representation. It is convenient not to distinguish between the first and second element of the same pair, and call them "string" or "number" arbitrarily. That is, we consider both the string 01 and the natural number 4 as the same object. For example, we may write $01 = 4$. We denote these objects in general with lowercase roman letters. A string consisting of n zeros is denoted by 0^n.

If x is a string of n 0's and 1's, then x_i denotes the ith *bit* (binary digit) of x for all i, $1 \leq i \leq n$, and $x_{i:j}$ denotes the $(j - i + 1)$-bits segment $x_i x_{i+1} \ldots x_j$. For $x = 1010$ we have $x_1 = x_3 = 1$ and $x_2 = x_4 = 0$; for $x = x_1 x_2 \ldots x_n$ we have $x_{1:i} = x_1 x_2 \ldots x_i$. The *reverse*, x^R, of a string $x = x_1 x_2 \ldots x_n$ is $x_n x_{n-1} \ldots x_1$.

The *length* of a finite binary string x is the number of bits it contains and is denoted by $l(x)$. If $x = x_1 x_2 \dots x_n$, then $l(x) = n$. In particular, $l(\epsilon) = 0$.

Thus, $l(xy) = l(x) + l(y)$, and $l(x^R) = l(x)$. Recall that we use the above pairing of binary strings and natural numbers. Thus, $l(4) = 2$ and $01 = 4$. The *number of elements* (*cardinality*) in a finite set A is denoted by $d(A)$. Therefore, $d(\{u : l(u) = n\}) = 2^n$ and $d(\{u : l(u) \leq n\}) = 2^{n+1} - 1$.

Let D be any function $D : \{0,1\}^* \to \mathcal{N}$. Considering the domain of D as the set of *code words*, and the range of D as the set of *source words*, $D(y) = x$ is interpreted as "y is a code word for the source word x, and D is the *decoding* function." (In the introduction we called D a specification method.) The set of all code words for source word x is the set $D^{-1}(x) = \{y : D(y) = x\}$. Hence, $E = D^{-1}$ can be called the *encoding* substitution (E is not necessarily a function). Let $x, y \in \{0,1\}^*$. We call x a *prefix* of y if there is a z such that $y = xz$. A set $A \subseteq \{0,1\}^*$ is *prefix-free*, if no element in A is the prefix of another element in A. A function $D : \{0,1\}^* \to \mathcal{N}$ defines a *prefix-code* if its domain is prefix-free. (Coding theory is treated in Section 1.11.1.) A simple prefix-code we use throughout is obtained by reserving one symbol, say 0, as a stop sign and encoding $x \in \mathcal{N}$ as $1^x 0$. We can prefix an object with its length and iterate this idea to obtain ever shorter codes:

$$E_i(x) = \begin{cases} 1^x 0 & \text{for } i = 0, \\ E_{i-1}(l(x))x & \text{for } i > 0. \end{cases} \tag{1.2}$$

Thus, $E_1(x) = 1^{l(x)}0x$ and has length $l(E_1(x)) = 2l(x)+1$. This encoding is sufficiently important to have a simpler notation:

$$\bar{x} = 1^{l(x)}0x,$$
$$l(\bar{x}) = 2l(x) + 1.$$

Sometimes we need the shorter prefix-code $E_2(x)$,

$$E_2(x) = \overline{l(x)}x,$$
$$l(E_2(x)) = l(x) + 2l(l(x)) + 1.$$

We call $\bar{x}$ the *self-delimiting* version of the binary string x. Now we can effectively recover both x and y unambiguously from the binary string $\bar{x}y$. If $\bar{x}y = 111011011$, then $x = 110$ and $y = 11$. If $\bar{x}\bar{y} = 1110110101$ then $x = 110$ and $y = 1$.

Example 1.4.1 It is convenient to consider also the set of *one-way infinite* sequences $\mathcal{B}^\infty$. If ω is an element of $\mathcal{B}^\infty$, then $\omega = \omega_1\omega_2 \dots$ and $\omega_{1:n} = \omega_1\omega_2 \dots \omega_n$.

The set of infinite sequences of elements in a finite, nonempty basic set $\mathcal{B}$ corresponds with the set $\mathcal{R}$ of real numbers in the following way:

Let $\mathcal{B} = \{0, 1, \ldots, k-1\}$ with $k \geq 2$. If r is a real number $0 < r < 1$ then there is a sequence $\omega_1\omega_2\ldots$ of elements $\omega_n \in \mathcal{B}$ such that

$$r = \sum_n \omega_n / k^n,$$

and that sequence is unique except when r is of the form q/k^n, in which case there are exactly two such sequences, one of which has infinitely many 0's. Conversely, if $\omega_1\omega_2\ldots$ is an infinite sequence of integers with $0 \leq \omega_n < k$, then the series

$$\sum_n \omega_n / k^n$$

converges to a real number r with $0 \leq r \leq 1$. This sequence is called the *k-ary* expansion of r. In the following we identify a real number r with its k-ary expansion (if there are two k-ary expansions, then we identify r with the expansion with infinitely many 0's).

Define the set $S \subseteq \mathcal{B}^\infty$ as the set of sequences that do not end with infinitely many digits "$k-1$." Then, S is in one-to-one relation with the set of real numbers in the interval $[0, 1)$.

Let x be a finite string over $\mathcal{B}$. The set of all one-way infinite sequences starting with x is called a *cylinder* and is denoted by $?_x$ and is defined by $?_x = \{\omega : \omega_{1:l(x)} = x\}$. Geometrically speaking, the cylinder $?_x$ can be identified with the *half-open interval* $[0.x, 0.x + 2^{-l(x)})$ in the real interval $[0, 1)$. Observe that the usual geometric length of interval $?_x$ equals $2^{-l(x)}$. Furthermore, $?_y \subseteq ?_x$ iff x is a prefix of y. The prefix relation induces a partial order on the cylinders of $\mathcal{B}^\infty$. ◇

Exercises

1.4.1. [03] If $\bar{x}\bar{y}z = 10010111$, what are x, y, z in decimal numbers?

Comments. 1, 2, 6.

1.4.2. [07] Show that for $x \in \mathcal{N}$ we have $l(x) = \lfloor \log(x+1) \rfloor$.

1.4.3. [10] Let $E : \mathcal{N} \to \{0,1\}^*$ be a total one-to-one function whose range is prefix-free. E defines a prefix-code. Define the mapping $\langle\cdot\rangle : \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ by $\langle x, y\rangle = E(x)y$.

(a) Show that $\langle\cdot\rangle$ is total and one-to-one.

(b) Show that we can extend this scheme to k-tuples $(n_1, n_2, \ldots, n_k)$ of natural numbers to obtain a total one-to-one mapping from $\mathcal{N} \times \mathcal{N} \times \cdots \times \mathcal{N}$ into $\mathcal{N}$.

Comments. Define the mapping for (x, y, z) as $\langle x, \langle y, z\rangle\rangle$, and iterate this construction.

1.4.4. [10] Let E be as above. Define the mapping $\langle\cdot\rangle : \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ by $\langle x, y\rangle = E(x)E(y)$.

(a) Show that $\langle\cdot\rangle$ is a total one-to-one mapping and a prefix-code.

(b) Show that we can extend this scheme to k-tuples $(n_1, n_2, \ldots, n_k)$ of natural numbers to obtain a total one-to-one mapping from $\mathcal{N} \times \mathcal{N} \times \cdots \times \mathcal{N}$ into $\mathcal{N}$ that is a prefix-code.

Comments. Define the mapping for (x, y, z) as $\langle x, \langle y, z\rangle\rangle$ and iterate this construction. Another way is to map $(x, y, \ldots, z)$ to $E(x)E(y)\ldots E(z)$.

1.4.5. [10] (a) Show that $E(x) = \bar{x}$ is a prefix-code.

(b) Consider a variant of the $\bar{x}$ code such that $x = x_1x_2\ldots x_n$ is encoded as $x_11x_21\ldots1x_{n-1}1x_n0$. Show that this is a prefix-code for the binary nonempty strings with $l(\bar{x}) = 2l(x)$.

(c) Consider $x = x_1x_2\ldots x_n$ encoded as $x_1x_1x_2x_2\ldots\ x_{n-1}x_{n-1}x_n\neg x_n$. Show that this is a prefix-code for the nonempty binary strings.

(d) Give a prefix-code $\tilde{x}$ for the set of all binary strings x including ϵ, such that $l(\tilde{x}) = 2l(x) + 2$.

1.5 Asymptotic Notation

It is often convenient to express approximate equality or inequality of one quantity with another. If f and g are functions of a real variable, then it is customary to denote $\lim_{n\to\infty} f(n)/g(n) = 1$ by $f(n) \sim g(n)$, and we write "f goes asymptotically to g."

P. Bachman introduced a convenient notation for dealing with approximations in his book *Analytische Zahlentheorie* in 1892. This "big-O" notation allows us to write $l(x) = \log x + O(1)$ (no subscript on the logarithm means base 2).

We use the notation $O(f(n))$ whenever we want to denote a quantity that does not exceed $f(n)$ by more than a fixed multiplicative factor. This is useful in case we want to simplify the expression involving this quantity by suppressing unnecessary detail, but also in case we do not know this quantity explicitly. Bachman's notation is the first of a family of *order of magnitude* symbols: O, o, Ω, and Θ. If f and g are functions on the real numbers, then

1. $f(x) = O(g(x))$ if there are constants $c, x_0 > 0$ such that $|f(x)| \leq c|g(x)|$, for all $x \geq x_0$;
2. $f(x) = o(g(x))$ if $\lim_{x\to\infty} f(x)/g(x) = 0$;
3. $f(x) = \Omega(g(x))$ if $f(x) \neq o(g(x))$; and