ALGORITHMIC INFORMATION THEORY

Cambridge Tracts in Theoretical Computer Science

Titles in the Series

1. G. J. Chaitin *Algorithmic Information Theory*
2. L. C. Paulson *Logic and Computation*
3. J. M. Spivey *Understanding Z*
4. G. E. Revesz *Lambda Calculus, Combinators and Functional Programming*
5. S. Vickers *Topology via Logic*
6. A. M. Ramsay *Formal Methods in Artificial Intelligence*
7. J.-Y. Girard *Proofs and Types*

ALGORITHMIC INFORMATION THEORY

GREGORY J. CHAITIN
IBM Research Division Thomas J. Watson Research Center
Yorktown Heights, New York

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York NY 10011–4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 1987
Reprinted with revisions 1988, 1990, 1992
First paperback edition 2004

A catalogue record for this book is available from the British Library

ISBN 0 521 34306 2 hardback
ISBN 0 521 61604 2 paperback

The author is pleased to acknowledge permission to make free use of previous publications:

Chapter 6 is based on his 1975 paper "A theory of program size formally identical to information theory" published in volume 22 of the *Journal of the ACM*, copyright © 1975, Association for Computing Machinery, Inc., reprinted by permission.

Chapters 7, 8, and 9 are based on his 1987 paper "Incompleteness theorems for random reals" published in volume 8 of *Advances in Applied Mathematics*, copyright © 1987 by Academic Press, Inc.

The author wishes to thank Ralph Gomory, Gordon Lasher, and the Physics Department of the Watson Research Center.

CONTENTS

Cambridge University Press
0521616042 - Algorithmic Information Theory
Gregory J. Chaitin
Frontmatter
More information

FOREWORD

Turing's deep 1937 paper made it clear that Gödel's astonishing earlier results on arithmetic undecidability related in a very natural way to a class of computing automata, nonexistent at the time of Turing's paper, but destined to appear only a few years later, subsequently to proliferate as the ubiquitous stored-program computer of today. The appearance of computers, and the involvement of a large scientific community in elucidation of their properties and limitations, greatly enriched the line of thought opened by Turing. Turing's distinction between computational problems was rawly binary: some were solvable by algorithms, others not. Later work, of which an attractive part is elegantly developed in the present volume, refined this into a multiplicity of scales of computational difficulty, which is still developing as a fundamental theory of information and computation that plays much the same role in computer science that classical thermodynamics plays in physics: by defining the outer limits of the possible, it prevents designers of algorithms from trying to create computational structures which provably do not exist. It is not surprising that such a thermodynamics of information should be as rich in philosophical consequence as thermodynamics itself.

This quantitative theory of description and computation, or Computational Complexity Theory as it has come to be known, studies the various kinds of resources required to describe and execute a computational process. Its most striking conclusion is that there exist computations and classes of computations having innocent-seeming definitions but nevertheless requiring inordinate quantities of some computational resource. Resources for which results of this kind have been established include:

(a) The mass of text required to describe an object;

(b) The volume of intermediate data which a computational process would need to generate;

(c) The time for which such a process will need to execute, either on a standard "serial" computer or on computational structures unrestricted in the degree of parallelism which they can employ.

Of these three resource classes, the first is relatively static, and pertains to the fundamental question of object describability; the others are dynamic since they relate to the resources required for a computation to execute. It is with the first kind of resource that this book is concerned. The crucial fact here is that there exist symbolic objects (i.e., texts) which are "algorithmically inexplicable," i.e., cannot be specified by any text shorter than themselves. Since texts of this sort have the properties associated with the random sequences of classical probability theory, the theory of describability developed in Part II of the present work yields a very interesting new view of the notion of randomness.

The first part of the book prepares in a most elegant, even playful, style for what follows; and the text as a whole reflects its author's wonderful enthusiasm for profundity and simplicity of thought in subject areas ranging over philosophy, computer technology, and mathematics.

J. T. Schwartz
Courant Institute
February, 1987

PREFACE

The aim of this book is to present the strongest possible version of Gödel's incompleteness theorem, using an information-theoretic approach based on the size of computer programs.

One half of the book is concerned with studying Ω, the halting probability of a universal computer if its program is chosen by tossing a coin. The other half of the book is concerned with encoding Ω as an algebraic equation in integers, a so-called exponential diophantine equation.

Gödel's original proof of his incompleteness theorem is essentially the assertion that one cannot always prove that a program will fail to halt. This is equivalent to asking whether it ever produces any output. He then converts this into an arithmetical assertion. Over the years this has been improved; it follows from the work on Hilbert's 10th problem that Gödel's theorem is equivalent to the assertion that one cannot always prove that a diophantine equation has no solutions if this is the case.

In our approach to incompleteness, we shall ask whether or not a program produces an infinite amount of output rather than asking whether it produces any; this is equivalent to asking whether or not a diophantine equation has infinitely many solutions instead of asking whether or not it is solvable.

If one asks whether or not a diophantine equation has a solution for N different values of a parameter, the N different answers to this question are not independent; in fact, they are only $\log_2 N$ bits of information. But if one asks whether or not there are infinitely many solutions for N different values of a parameter, then there are indeed cases in which the N different answers to these questions are independent mathematical facts, so that knowing one answer is no help in knowing any of the others. The equation encoding Ω has this property.

When mathematicians can't understand something they usually assume that it is their fault, but it may just be that there is no pattern or law to be discovered!

How to read this book: This entire monograph is essentially a proof of one theorem, Theorem D in Chapter 8. The exposition is completely self-contained, but the collection CHAITIN (1987c) is a useful source of background material. While the reader is assumed to be familiar with the basic concepts of recursive function or computability theory and probability theory, at a level easily acquired from DAVIS (1965) and FELLER (1970), we make no use of individual results from these fields that we do not reformulate and prove here. Familiarity with LISP programming is helpful but not necessary, because we give a self-contained exposition of the unusual version of pure LISP that we use, including a listing of an interpreter. For discussions of the history and significance of metamathematics, see DAVIS (1978), WEBB (1980), TYMOCZKO (1986), and RUCKER (1987).

Although the ideas in this book are not easy, we have tried to present the material in the most concrete and direct fashion possible. We give many examples, and computer programs for key algorithms. In particular, the theory of program-size in LISP presented in Chapter 5 and Appendix B, which has not appeared elsewhere, is intended as an illustration of the more abstract ideas in the following chapters.

Cambridge University Press
0521616042 - Algorithmic Information Theory
Gregory J. Chaitin
Frontmatter
More information

FIGURES