Graduate Texts in Mathematics **146**

Graduate Texts in Mathematics

1 TAKEUTI/ZARING. Introduction to Axiomatic Set Theory. 2nd ed.
2 OXTOBY. Measure and Category. 2nd ed.
3 SCHAEFFER. Topological Vector Spaces.
4 HILTON/STAMMBACH. A Course in Homological Algebra.
5 MAC LANE. Categories for the Working Mathematician.
6 HUGHES/PIPER. Projective Planes.
7 SERRE. A Course in Arithmetic.
8 TAKEUTI/ZARING. Axiometic Set Theory.
9 HUMPHREYS. Introduction to Lie Algebras and Representation Theory.
10 COHEN. A Course in Simple Homotopy Theory.
11 CONWAY. Functions of One Complex Variable. 2nd ed.
12 BEALS. Advanced Mathematical Analysis.
13 ANDERSON/FULLER. Rings and Categories of Modules. 2nd ed.
14 GOLUBITSKY/GUILEMIN. Stable Mappings and Their Singularities.
15 BERBERIAN. Lectures in Functional Analysis and Operator Theory.
16 WINTER. The Structure of Fields.
17 ROSENBLATT. Random Processes. 2nd ed.
18 HALMOS. Measure Theory.
19 HALMOS. A Hilbert Space Problem Book. 2nd ed.
20 HUSEMOLLER. Fibre Bundles. 3rd ed.
21 HUMPHREYS. Linear Algebraic Groups.
22 BARNES/MACK. An Algebraic Introduction to Mathematical Logic.
23 GREUB. Linear Algebra. 4th ed.
24 HOLMES. Geometric Functional Analysis and Its Applications.
25 HEWITT/STROMBERG. Real and Abstract Analysis.
26 MANES. Algebraic Theories.
27 KELLEY. General Topology.
28 ZARISKI/SAMUEL. Commutative Algebra. Vol. I.
29 ZARISKI/SAMUEL. Commutative Algebra. Vol. II.
30 JACOBSON. Lectures in Abstract Algebra I. Basic Concepts.
31 JACOBSON. Lectures in Abstract Algebra II. Linear Algebra.
32 JACOBSON. Lectures in Abstract Algebra III. Theory of Fields and Galois Theory.
33 HIRSCH. Differential Topology.
34 SPITZER. Principles of Random Walk. 2nd ed.
35 WERMER. Banach Algebras and Several Complex Variables. 2nd ed.
36 KELLEY/NAMIOKA et al. Linear Topological Spaces.
37 MONK. Mathematical Logic.
38 GRAUERT/FRITZSCHE. Several Complex Variables.
39 ARVESON. An Invitation to C^*-Algebras.
40 KEMENY/SNELL/KNAPP. Denumerable Markov Chains. 2nd ed.
41 APOSTOL. Modular Functions and Dirichlet Series in Number Theory. 2nd ed.
42 SERRE. Linear Representations of Finite Groups.
43 GILLMAN/JERISON. Rings of Continuous Functions.
44 KENDIG. Elementary Algebraic Geometry.
45 LOÈVE. Probability Theory I. 4th ed.
46 LOÈVE. Probability Theory II. 4th ed.
47 MOISE. Geometric Topology in Dimensions 2 and 3.

continued after index

Douglas S. Bridges

Computability

A Mathematical Sketchbook

With 29 Illustrations

Springer Science+Business Media, LLC

Douglas S. Bridges
Department of Mathematics
University of Waikato
Private Bag 3105
Hamilton, New Zealand

Mathematics Subject Classifications (1991): 03Dxx

Library of Congress Cataloging-in-Publication Data
Bridges, D.S. (Douglas S.), 1945-
Computability : a mathematical sketchbook / Douglas S. Bridges.
p. cm. – (Graduate texts in mathematics)
Includes bibliographical references and index.
ISBN 978-1-4612-6925-0 ISBN 978-1-4612-0863-1 (eBook)
DOI 10.1007/978-1-4612-0863-1
1. Computable functions. I. Title. II. Series.
QA9.59.B75 1994
511.3–dc20 93-21313

Printed on acid-free paper.

Production managed by Hal Henglein; manufacturing supervised by Vincent Scelta.
Photocomposed pages prepared from the author's LaTeX file.

9 8 7 6 5 4 3 2 1

ISBN 978-1-4612-6925-0

For Vivien, Iain, Hamish, and Catriona

'I can't believe that!' said Alice. 'Can't you?' the Queen said in a pitying tone. 'Try again: draw a long breath and shut your eyes.' Alice laughed. 'There's no use trying,' she said: 'One can't believe impossible things.' 'I daresay you haven't had much practice,' said the Queen.

LEWIS CARROLL, *Through the Looking Glass.*

Preface

My intention in writing this book is to provide mathematicians and mathematically literate computer scientists with a brief but rigorous introduction to a number of topics in the abstract theory of computation, otherwise known as *computability theory* or *recursion theory.* It develops major themes in computability, such as Rice's Theorem and the Recursion Theorem, and provides a systematic account of Blum's abstract complexity theory up to his famous Speed-up Theorem.

A relatively unusual aspect of the book is the material on computable real numbers and functions, in Chapter 4. Parts of this material are found in a number of books, but I know of no other at the senior/beginning graduate level that introduces elementary recursive analysis as a natural development of computability theory for functions from natural numbers to natural numbers.[1] This part of the book is definitely for mathematicians rather than computer scientists and has a prerequisite of a first course in elementary real analysis; it can be omitted, without rendering the subsequent chapters unintelligible, in a course including the more standard topics in computability theory found in Chapters 4-6.

I believe, against the trend towards weighty, all-embracing treatises (*vide* the typical modern calculus text), that many mathematicians would like to be able to purchase books that give them insight into unfamiliar branches of the subject in a relatively short compass and without requiring a major investment of time, effort, or money. Following that belief, I have had to exclude from this book many topics—such as detailed proofs of the equivalence of various mathematical models of computation, the theory of degrees of unsolvability, and polynomial and nonpolynomial complexity—whose absence will be deplored by at least some of the experts in the field. I hope that my readers will be inspired to pursue their study of recursion theory in such major works as [9, 24, 28, 29].

A number of excellent texts on computability theory are primarily aimed at computer scientists rather than mathematicians, and so do not always maintain the level of rigour that would be expected in a modern text on, say, abstract algebra. I have tried to maintain that higher level of rigour

[1] Some of the work in this book—notably, Proposition (4.28) and the application of the Recursion Theorem preceding Exercises (5.14)—appears to be original.

throughout, even at the risk of deflecting the interest of mathematically insecure computer scientists.

Ideally, all mathematics and computer science majors should be exposed to at least some of the material found in this book. It horrifies me that in some universities such majors can still graduate ignorant of the theoretical limitations of the computer, as expressed, for example, by the undecidability of the halting problem (Theorem (4.2)). A short course on computability, accessible even to students below junior level, would comprise Chapters 1-3 and the material in Chapter 4 up to Exercises (4.7). A longer course for more advanced undergraduates would also include Rice's theorem and the Recursion Theorem, from Chapter 5, and at least parts of Chapter 6. The entire book, including the difficult material on recursive analysis from Chapter 4, would be suitable for a course for bright seniors or beginning graduate students.

I have tried to make the book suitable for self-study. To this end, it includes solutions for most of the exercises. Those exercises for which no solutions are given have been marked with the asterisk (*); of varying levels of difficulty, they provide the instructor with material for homework and tests. *The exercises form an integral part of the book* and are not just there for the student's practice; many of them develop material that is used in later proofs, which is another reason for my inclusion of solutions.

My interest in constructive mathematics [5] leads me to comment here on the logic of computability theory. This is *classical logic,* the logic used by almost all mathematicians in their daily work. However, the use of classical logic has some perhaps undesirable consequences. Consider the following definition of a function f on the set $\mathbf{N}$ of natural numbers: for all n, $f(n)$ equals 1 if the Continuum Hypothesis is true, and equals 0 if the Continuum Hypothesis is false.[2] Since 'most mathematicians are formalists on weekdays and Platonists on Sundays', at least on Sundays most of us would accept this as a good definition of a function f. According to classical logic, f is computable because there exists an algorithm that computes it; that algorithm is either the one which, applied to any natural number n, outputs 1, or else the one which, applied to any natural number n, outputs 0. But the Continuum Hypothesis is independent of the axioms of ZFC (Zermelo-Fraenkel set theory plus the axiom of choice), the standard framework of mathematics, so we will never be able to tell, using ZFC alone, which of the two algorithms actually is the one that computes f.

It appears from this example, eccentric though it may be, that the standard theory of computation does not exactly match computational practice,

---

[2]The **Continuum Hypothesis** (CH) says that the smallest cardinal number greater than $\aleph_0$, the cardinality of $\mathbf{N}$, is $2^{\aleph_0}$, the cardinality of the set of all subsets of $\mathbf{N}$. The work of Cohen [13] and Gödel [17] shows that neither CH nor its negation can be proved within Zermelo-Fraenkel set theory plus the axiom of choice; see also [3], pages 420-428.

in which we would expect to pin down the algorithms that we use. A facetious question may reinforce my point: what would happen to an employee who, in response to a request that he write software to perform a certain computation, presented his boss with two programs and the information that, although one of those programs performed the required computation, nobody could ever tell which one?

With classical logic there seems to be no way to distinguish between functions that are computed by programs which we can pin down and those that are computable but for which there is no hope of our telling which of a range of programs actually performs the desired computation. To handle this problem successfully, we need a different logic, one capable of distinguishing between *existence in principle* and *existence in practice.* For example, with constructive (intuitionistic) logic the problem disappears,[3] since f is then not properly defined: it is only properly defined if we can decide the truth or falsehood of the Continuum Hypothesis (which we cannot) and therefore which of the two possible algorithms computes f.

Having said this, let me stress that, despite the inability of classical logic to make certain distinctions of the type I have just dealt with, *I have followed standard practice and used classical logic throughout this book.*

Not only the logic but also most of the material that I have chosen is standard, although some of the exercises and examples are new. I have drawn on a number of books, including [34] for the treatment of Turing machines in Chapter 1; [20] for the first parts of Chapters 4 and 5; and [9, 14, 29] for parts of Chapter 7.

The origins of my book lie in courses I gave at the University of Buckingham (England), New Mexico State University (USA), and the University of Waikato (New Zealand). I am grateful to the students in those classes for the patience with which they received various slowly improving draft versions.[4] Special thanks are due to Fred Richman for many illuminating conversations about recursion theory; to Paul Halmos for his advice and encouragement; and to Cris Calude, Nick Dudley Ward, Graham French, Hazel Locke, and Steve Merrin, all of whom have read versions of the text and made many helpful corrections and suggestions. As always, it is my wife and children who suffered most as the prolonged birth of this work took so much of my care and attention; I present the book to them with love and gratitude.

May 1993 *Douglas S. Bridges*

---

[3]For a development of computability theory using intuitionistic logic see Chapter 3 of [8].

[4]The first drafts of this book were prepared using the T^3 *Scientific Word Processing System.* The final version was produced by converting the drafts to $\TeX$ and then using *Scientific Word.* T^3 and *Scientific Word* are both products of TCI Software Research, Inc. The diagrams were drawn with *Aldus Freehand* v. 3.1 (©Aldus Corporation).

Contents

Preface vii

Preliminaries 1

1 What Is a Turing Machine? 5

2 Computable Partial Functions 19

3 Effective Enumerations 35

4 Computable Numbers and Functions 47

5 Rice's Theorem and the Recursion Theorem 75

6 Abstract Complexity Theory 93

Solutions to Exercises 117

Solutions for Chapter 1 118
Solutions for Chapter 2 120
Solutions for Chapter 3 130
Solutions for Chapter 4 136
Solutions for Chapter 5 156
Solutions for Chapter 6 166

References 173

Index 176