TEXTS IN COMPUTER SCIENCE

Editors
David Gries
Fred B. Schneider

Springer Science+Business Media, LLC

TEXTS IN COMPUTER SCIENCE

Alagar and Periyasamy, Specification of Software Systems

Apt and Olderog, Verification of Sequential and Concurrent Programs, Second Edition

Back and von Wright, Refinement Calculus

Beidler, Data Structures and Algorithms

Bergin, Data Structure Programming

Brooks, C Programming: The Essentials for Engineers and Scientists

Brooks, Problem Solving with Fortran 90

Dandamudi, Introduction to Assembly Language Programming

Fitting, First-Order Logic and Automated Theorem Proving, Second Edition

Grillmeyer, Exploring Computer Science with Scheme

Homer and Selman, Computability and Complexity Theory

Immerman, Descriptive Complexity

Jalote, An Integrated Approach to Software Engineering, Second Edition

Kizza, Ethical and Social Issues in the Information Age

Kozen, Automata and Computability

Li and Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, Second Edition

(continued after index)

Steven Homer Alan L. Selman

COMPUTABILITY AND COMPLEXITY THEORY

With 17 Illustrations

Springer

Steven Homer
Department of Computer Science
Boston University
111 Cummington Street
Boston, MA 02215, USA
homer@cs.bu.edu

Alan L. Selman
Department of Computer Science and Engineering
226 Bell Hall
University at Buffalo
Buffalo, NY 14260-2000, USA
selman@cse.buffalo.edu

Series Editors

David Gries
Department of Computer Science
415 Boyd Studies Research Center
The University of Georgia
Athens, GA 30605, USA

Fred B. Schneider
Department of Computer Science
Upson Hall
Cornell University
Ithaca, NY 14853-7501, USA

Library of Congress Cataloging-in-Publication Data
Homer, S. (Steven)
Computability and complexity theory / Steven Homer, Alan L. Selman.
p. cm. — (Texts in computer science)
Includes bibliographical references and index.
ISBN 978-1-4419-2865-8 ISBN 978-1-4757-3544-4 (eBook)
DOI 10.1007/978-1-4757-3544-4
1. Computer science. 2. Computable functions. 3. Computational complexity.
I. Selman, Alan L. II. Title. III. Series.
QA76 .H6236 2001
004—dc21 00-053829

Printed on acid-free paper.

© 2001 Springer Science+Business Media New York
Originally published by Springer-Verlag New York, Inc. in 2001
Softcover reprint of the hardcover 1st edition 2001

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use of general descriptive names, trade names, trademarks, etc., in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

Production managed by MaryAnn Brickner; manufacturing supervised by Joe Quatela.
Typeset pages prepared using the authors' LaTeX 2_ε files by Integre Technical Publishing Company, Inc.

9 8 7 6 5 4 3 2 1 SPIN 10769347

We dedicate this book to our wives, Michelle and Sharon

Preface

The theory of computing provides computer science with concepts, models, and formalisms for reasoning about both the resources needed to carry out computations and the efficiency of the computations that use these resources. It provides tools to measure the difficulty of combinatorial problems both absolutely and in comparison with other problems. Courses in this subject help students gain analytic skills and enable them to recognize the limits of computation. For these reasons, a course in the theory of computing is usually required in the graduate computer science curriculum.

The harder question to address is which topics such a course should cover. We believe that students should learn the fundamental models of computation, the limitations of computation, and the distinctions between feasible and intractable. In particular, the phenomena of NP-completeness and NP-hardness have pervaded much of science and transformed computer science. One option is to survey a large number of theoretical subjects, typically focusing on automata and formal languages. However, these subjects are less important to theoretical computer science, and to computer science as a whole, now than in the past. Many students have taken such a course as part of their undergraduate education. We chose not to take that route because computability and complexity theory are the subjects that we feel deeply about and that we believe are important for students to learn. Furthermore, a graduate course should be scholarly. It is better to treat important topics thoroughly than to survey the field.

This textbook is intended for use in an introductory graduate course in theoretical computer science. It contains material that should be core knowledge in the theory of computation for all graduate students in computer science. It is self-contained and is best suited for a one-semester course. Most of the text can be covered in one semester by moving expeditiously through the core material of

Chapters 2 through 6 and then covering parts of Chapter 7. We will give more details about this below.

As a graduate course, students should have some prerequisite preparation. The ideal preparation would be the kind of course that we mentioned above: an undergraduate course that introduced topics such as automata theory, formal languages, computability theory, or complexity theory. We stress, however, that there is nothing in such a course that a student needs to know before studying this text. Our personal experience suggests that we cannot presume that all of our students have taken such an undergraduate course. For those students who have not, we advise that they need at least some prior exposure that will have developed mathematical skills. Prior courses in mathematical logic, algebra (at the level of groups, rings, or fields), or number theory, for example, would all serve this purpose.

Despite the diverse backgrounds of our students, we have found that graduate students are capable of learning sophisticated material when it is explained clearly and precisely. That has been our goal in writing this book.

This book also is suitable for advanced undergraduate students who have satisfied the prerequisites. It is an appropriate first course in complexity theory for students who will continue to study and work in this subject area.

The text begins with a preliminary chapter that gives a brief description of several topics in mathematics. We included this in order to keep the book self-contained and to ensure that all students have a common notation. Some of these sections simply enable students to understand some of the important examples that arise later. For example, we include a section on number theory and algebra that includes all that is necessary for students to understand that primality belongs to NP.

The text starts properly with classical computability theory. We build complexity theory on top of that. Doing so has the pedagogical advantage that students learn a qualitative subject before advancing to a quantitative one. Also, the concepts build from one to the other. For example, although we give a complete proof that the satisfiability problem is NP-complete, it is easy for students to understand that the bounded halting problem is NP-complete, because they already know that the classical halting problem is c.e.-complete.

We use the terms *partial computable* and *computably enumerable (c.e.)* instead of the traditional terminology, *partial recursive* and *recursively enumerable (r.e.)*, respectively. We do so simply to eliminate confusion. Students of computer science know of "recursion" as a programming paradigm. We do not prove here that Turing-computable partial functions are equivalent to partial recursive functions, so by not using that notation, we avoid the matter altogether. Although the notation we are using has been commonplace in the computability theory and mathematical logic community for several years, instructors might want to advise their students that the older terminology seems commonplace within the theoretical computer science community. Computable functions are defined on the set of words over a finite alphabet, which we identify with the set of natural numbers in a straightforward manner. We use the term *effective*, in the nontechnical, intu-

itive sense, to denote computational processes on other data types. For example, we will say that a set of Turing machines is "effectively enumerable" if its set of indices is computably enumerable.

Chapter 4 concludes with a short list of topics that students should know from the chapters on computability theory before proceeding to study complexity theory. We advise instructors who wish to minimize coverage of computability theory to refer to this list. Typically, we do not cover the second section on the recursion theorem (Section 3.10) in a one-semester course. Although we do not recommend it, it is possible to begin the study of complexity theory after learning the first five sections of Chapter 3 and at least part of Section 3.9 on oracle Turing machines, Turing reductions, and the arithmetical hierarchy.

In Chapter 5 we treat general properties of complexity classes and relationships between complexity classes. These include important older results such as the space and time hierarchy theorems, as well as the more recent result of Immerman and Szelepcsényi that space-bounded classes are closed under complements. Instructors might be anxious to get to NP-complete problems (Chapter 6) and NP-hard problems (Chapter 7) but students need to learn the basic results of complexity theory and it is instructive for them to understand the relationships between P, NP, and other deterministic and nondeterministic, low-level complexity classes. Students should learn that nondeterminism is not well understood in general, that P =? NP is not an isolated question, and that other classes have complete problems as well (which we take up in Chapter 7). Nevertheless, Chapter 5 is a long chapter. Many of the results in this chapter are proved by complicated Turing-machine simulations and counting arguments, which give students great insight, but can be time-consuming to cover. For this reason, instructors might be advised to survey some of this material if the alternative would mean not having sufficient time for the later chapters.

Homework exercises are an important part of this book. They are embedded in the text where they naturally arise, and students should not proceed without working on them. Many are simple exercises, whereas others are challenging. Often we leave important but easy-to-prove propositions as exercises. We provide additional problems at the end of chapters, which extend and apply the material covered there.

Once again, our intent has been to write a text that is suitable for all graduate students, that provides the right background for those who will continue to study complexity theory, and that can be taught in one semester. There are several important topics in complexity theory that cannot be treated properly in a one-semester course. Currently we are writing a second part to this text, which will be suitable for an optional second semester course, covering nonuniform complexity (Boolean circuits), parallelism, probabilistic classes, and interactive protocols.

Boston, Massachusetts — *Steven Homer*
Buffalo, New York — *Alan L. Selman*
January, 2001

Contents

Preface **vii**

1 Preliminaries **1**
1.1 Words and Languages 1
1.2 K-adic Representation 2
1.3 Partial Functions 3
1.4 Graphs 4
1.5 Propositional Logic 6
1.5.1 Boolean Functions 8
1.6 Cardinality 8
1.6.1 Ordered Sets 10
1.7 Elementary Algebra 11
1.7.1 Rings and Fields 11
1.7.2 Groups 15
1.7.3 Number Theory 17

2 Introduction to Computability **22**
2.1 Turing Machines 23
2.2 Turing-Machine Concepts 26
2.3 Variations of Turing Machines 28
2.3.1 Multitape Turing Machines 29
2.3.2 Nondeterministic Turing Machines 31
2.4 Church's Thesis 34
2.5 RAMs 36
2.5.1 Turing Machines for RAMS 39

3 Undecidability **41**
3.1 Decision Problems 41
3.2 Undecidable Problems 43
3.3 Pairing Functions 46
3.4 Computably Enumerable Sets 47
3.5 Halting Problem, Reductions, and Complete Sets 50
3.5.1 Complete Problems 52
3.6 *S-m-n* Theorem 53
3.7 Recursion Theorem 55
3.8 Rice's Theorem 57
3.9 Turing Reductions and Oracle Turing Machines 59
3.10 Recursion Theorem, Continued 66
3.11 References 69
3.12 Additional Homework Problems 70

4 Introduction to Complexity Theory **72**
4.1 Complexity Classes and Complexity Measures 74
4.1.1 Computing Functions 76
4.2 Prerequisites 77

5 Basic Results of Complexity Theory **78**
5.1 Linear Compression and Speedup 80
5.2 Constructible Functions 86
5.2.1 Simultaneous Simulation 87
5.3 Tape Reduction 90
5.4 Inclusion Relationships 97
5.4.1 Relations between the Standard Classes 105
5.5 Separation Results 107
5.6 Translation Techniques and Padding 111
5.6.1 Tally Languages 113
5.7 Relations between the Standard Classes—Continued 115
5.7.1 Complements of Complexity Classes: The Immerman–Szelepcsényi Theorem 116
5.8 Additional Homework Problems 120

6 Nondeterminism and NP-Completeness **122**
6.1 Characterizing NP 123
6.2 The Class P 124
6.3 Enumerations 126
6.4 NP-Completeness 128
6.5 The Cook–Levin Theorem 130
6.6 More NP-Complete Problems 136
6.6.1 The Diagonal Set Is NP-Complete 137
6.6.2 Some Natural NP-Complete Problems 138

6.7 Additional Homework Problems 142

7 Relative Computability **145**
7.1 NP-Hardness 147
7.2 Search Problems 151
7.3 The Structure of NP 153
7.3.1 Composite Number and Graph Isomorphism 158
7.3.2 Reflection 161
7.4 The Polynomial Hierarchy 162
7.5 Complete Problems for Other Complexity Classes 170
7.5.1 PSPACE 170
7.5.2 Exponential Time 174
7.5.3 Polynomial Time and Logarithmic Space 175
7.5.4 A Note on Provably Intractable Problems 179
7.6 Additional Homework Problems 179

References **181**

Author Index **187**

Subject Index **191**