Preface

Theoretical computer science treats any computational subject for which a good model can be created. Research on formal models of computation was initiated in the 1930s and 1940s by Turing, Post, Kleene, Church, and others. In the 1950s and 1960s programming languages, language translators, and operating systems were under development and therefore became both the subject and basis for a great deal of theoretical work. The power of computers of this period was limited by slow processors and small amounts of memory, and thus theories (models, algorithms, and analysis) were developed to explore the efficient use of computers as well as the inherent complexity of problems. The former subject is known today as *algorithms and data structures*, the latter *computational complexity*.

The focus of theoretical computer scientists in the 1960s on languages is reflected in the first textbook on the subject, *Formal Languages and Their Relation to Automata* by John Hopcroft and Jeffrey Ullman. This influential book led to the creation of many language-centered theoretical computer science courses; many introductory theory courses today continue to reflect the content of this book and the interests of theoreticians of the 1960s and early 1970s.

Although the 1970s and 1980s saw the development of models and methods of analysis directed at understanding the limits on the performance of computers, this attractive new material has not been made available at the introductory level. This book is designed to remedy this situation.

This book is distinguished from others on theoretical computer science by its primary focus on real problems, its emphasis on concrete models of machines and programming styles, and the number and variety of models and styles it covers. These include the logic circuit, the finite-state machine, the pushdown automaton, the random-access machine, memory hierarchies, the PRAM (parallel random-access machine), the VLSI (very large-scale integrated) chip, and a variety of parallel machines.

The book covers the traditional topics of formal languages and automata and complexity classes but also gives an introduction to the more modern topics of space-time tradeoffs, memory hierarchies, parallel computation, the VLSI model, and circuit complexity. These modern topics are integrated throughout the text, as illustrated by the early introduction of **P**-complete and **NP**-complete problems. The book provides the first textbook treatment of space-time tradeoffs and memory hierarchies as well as a comprehensive introduction to traditional computational complexity. Its treatment of circuit complexity is modern and substantative, and parallelism is integrated throughout.

Plan of the Book

The book has three parts. Part I (Chapter 1) is an overview. Part II, consisting of Chapters 2–7, provides an introduction to general computational models. Chapter 2 introduces logic circuits and derives upper bounds on the size and depth of circuits for important problems. The finite-state, random-access, and Turing machine models are defined in Chapter 3 and circuits are presented that simulate computations performed by these machines. From such simulations arise results of two kinds. First, computational inequalities of the form $C(f) \leq \kappa ST$ are derived for problems f run on the random-access machine, where $C(f)$ is the size of the smallest circuit for f, κ is a constant, and S and T are storage space and computation time. If ST is too small relative to $C(f)$, the problem f cannot be solved. Second, the same circuit simulations are interpreted to identify **P**-complete and **NP**-complete problems. **P**-complete problems can all be solved in polynomial time but are believed hard to solve fast on parallel machines. The **NP**-complete problems include many important scheduling and optimization problems and are believed not solvable in polynomial time on serial machines.

Part II also contains traditional material on formal languages and automata. Chapter 4 explores the connection between two machine models (the finite-state machine and the push-down automaton) and language types in the Chomsky hierarchy. Chapter 5 examines Turing machines. It shows that the languages recognized by them are the phrase-structure languages, the most expressive of the language types in the Chomsky hierarchy. This chapter also examines universal Turing machines, reducibility, unsolvable problems, and the functions computed by Turing machines.

Finally, Part II contains Chapters 6 and 7 which introduce algebraic and combinatorial circuits and parallel machine models, respectively. Algebraic and combinatorial circuits are graphs of straight-line programs of the kind typically used for matrix multiplication and inversion, solving linear systems of equations, computing the fast Fourier transform, performing convolutions, and merging and sorting. Chapter 6 contains reference material on problems used in later chapters to illustrate models and lower-bound arguments. Parallel machine models such as the PRAM and networks of computers organized as meshes and hypercubes are studied in Chapter 7. A framework is given for the design of algorithms and derivation of lower bounds on performance.

Part III, a comprehensive treatment of computational complexity, consists of Chapters 8–12. Chapter 8 provides a comprehensive survey of traditional computational complexity. Using serial and parallel machine models, it examines time- and space-bounded complexity classes, including the **P**-complete, **NP**-complete and **PSPACE**-complete languages as well as the circuit complexity classes **NC** and **P/poly**. This chapter also establishes the connections between de-

terministic and nondeterministic space complexity classes and shows that the nondeterministic space classes are closed under complements.

Circuit complexity is the topic of Chapter 9. Methods for deriving lower bounds on circuit size and depth are given for general circuits, formulas, monotone circuits, and bounded-depth circuits. This modern treatment of circuit complexity complements Chapter 2, which derives tight upper bounds on circuit size and depth.

Space–time tradeoffs are studied in Chapter 10 using two computational models, the branching program and the pebble game, which capture the notions of space and time for many programs for which branching is and is not allowed, respectively. Methods for deriving lower bounds on the exchange of space for time are presented and applied to a representative set of problems.

Chapter 11 examines models for memory hierarchy systems. It uses the pebble game with pebbles of multiple colors to designate storage locations at different levels of a hierarchy, and also employs block and RAM-based models. Again, lower bounds on performance are derived and compared with the performance of algorithms. This chapter also has a brief treatment of the LRU and FIFO memory-management algorithms that uses competitive analysis to compare their performance to that of the optimal algorithm.

The book closes with Chapter 12 on the VLSI model for integrated circuits. In this model both chip area A and time T are important, and methods are given for deriving lower bounds on measures such as AT^2. Chip layouts and VLSI algorithms are also exhibited whose performance comes close to matching the lower bounds.

Use of the Book

Many different courses can be designed around this book. A core undergraduate computer science course can be taught using Parts I and II and some material from Chapter 8. The first course on theoretical computer science for majors at Brown uses most of Chapters 1–5 except for the advanced material in Chapters 2 and 3. It uses a few elementary sections from Chapters 10 and 11 to emphasize space–time tradeoffs, which play a central role in Chapter 3 and lead into the study of formal languages and automata in Chapter 4. After covering the material of Chapter 5, a few lectures are given on **NP**-complete problems from Chapter 8.

This introductory course has four programming assignments in Scheme that illustrate the ideas embodied in Chapters 2, 3 and 5. The first program solves the circuit-value problem, that is, it executes a straight-line program, thereby producing the outputs defined by this program. The second program writes a straight-line program simulating T steps by a finite-state machine. The third program writes a straight-line program simulating T steps by a one-tape Turing machine (this is the reduction involved in the Cook-Levin theorem) and the fourth one simulates a universal Turing machine.

Several different advanced courses can be assembled from the material of Part III and introductory material of Part II. For example, a course on concrete computational complexity can be assembled around Chapters 10 and 11, which examine tradeoffs between space and time in primary and secondary memory. This course would presume or include introductory material from Chapter 3.

An advanced course emphasizing traditional computational complexity can be based primarily on computability (Chapter 5) and complexity classes (Chapter 8) and some material on circuit complexity from Chapter 9.

An advanced course on circuit complexity can be assembled from Chapter 2 on logic circuits and Chapter 9 on circuit complexity. The former describes efficient circuits for a variety of functions while the latter surveys methods for deriving lower bounds to circuit complexity.

The titles of sections containing advanced material carry an **asterisk**.

Acknowledgments

The raw material for this book is the fruit of the labors of many hundreds of people who have sought to understand computation. It is a great privilege to have the opportunity to convey this exciting body of material to a new audience.

Because the writing of a book involves years of solitary work, it is far too easy for authors to lose sight of their audience. For this reason I am indebted to a number of individuals who have read my book critically. José G. Castaños, currently a Brown Ph.D. candidate and my advisee, has been of immense help to me in this regard. He has read many drafts of the book and has given me the benefit of his keen sense of what is acceptable to my readers. José has also served as a teaching assistant for the undergraduate theory course for which this book was used and contributed importantly to the course and the book in this capacity. Dimitrios Michailidis, also a Brown Ph.D. candidate, has also been a great help; he has read several drafts of the book and has spotted many errors and lacunae. Bill Smart, a third Brown Ph.D. candidate, also carefully read the first nine chapters. I have also benefited greatly from the evaluations done for my publisher by Richard Chang, University of Maryland, Baltimore County; Michael A. Keenan, Columbus State University; Philip Lewis, State University of New York, Stony Brook; George Lukas, University of Massachusetts at Boston; Stephen R. Mahaney, Rutgers University; Friedhelm Meyer auf der Heide, University of Paderborn, Germany; Boleslaw Mikolajczak, University of Massachusetts, Dartmouth; Ramamohan Paturi, University of California, San Diego; Professor Gabriel Robins, and Jeffery Westbrook, AT&T Labs–Research. Others, including Ray Greenlaw of the University of New Hampshire, read an early version of the manuscript for other publishers and offered valuable advice. Gary Rommel of the Eastern Connecticut State College and the Hartford Graduate Center provided feedback on classroom use of the book. Finally, I am indebted to students in my undergraduate and graduate courses at Brown whose feedback has been invaluable.

I very much appreciate advice on the content and organization of the book provided by many individuals including my faculty colleagues the late Paris Kanellakis, Philip Klein, and Franco Preparata as well as Akira Maruoka, a visitor to Brown. Together Franco and I also produced the brief analysis of circuit depth given in Section 2.12.2. Alan Selman offered valuable comments on Chapter 8. Akira Maruoka and Johan Håstad read and commented on the sections of Chapter 9 containing their work. Alan Selman and Ken Regan provided help in identifying references and Allan Borodin commented on many of the chapter notes. I wish to thank Jun Tarui for suggesting that I consider rewriting my 1976 book, *The Complexity of Computing*, which led to my writing this book. I also extend my sincere thanks to Andy Yao for his generous comments on the book for the publisher. Many others contributed to this book in one way or another, including Chuck Fiduccia, Paul Fischer, Bill McColl, Tony Medeiros, Mike Paterson, Eric Rudder, Elizabeth and Kevin Savage, Mark Snir, and many students in my courses.

I express my gratitude to Carter Shanklin, Executive Editor for Corporate & Professional Publishing at Addison Wesley Longman, for his confidence in me and this project. I also thank

Alwyn Velásquez for his attractive design of the book, Patricia Unubun, Production Editor on this project, for her secure guidance of the book in its final stages, and Dimitrios Michailidis, an expert in LaTeX, for his preparation of the macros used to typeset the book and his very close reading of the complete, formatted document, for which I am most appreciative. I offer my sincere thanks to Katrina Avery for her high-quality copyediting, Rosemary Simpson for her excellent index which is a great addition to this book, and Cynthia Benn for her careful proofreading of the manuscript. The attractive cover of this book was designed by Michael LeGrand and Scott Klemmer, two very talented juniors at Brown University.

Finally, this book would not have been written without the loving support of my wife Patricia and our young sons, Christopher and Timothy. Their patience and understanding for my long absences during the four and one-half years this project was in process is deeply appreciated.