Theory Of Computer Science By S S Sane

"Of all the books I have covered in the Forum to date, this set is the most unique and possibly the most useful to the SIGACT community, in support both of teaching and research.... The books can be used by anyone wanting simply to gain an understanding of one of these areas, or by someone desiring to be in research in a topic, or by instructors wishing to find timely information on a subject they are teaching outside their major areas of expertise." -- Rocky Ross, "SIGACT News" "This is a reference which has a place in every computer science library." -- Raymond Lauzzana, "Languages of Design" The Handbook of Theoretical Computer Science provides professionals and students with a comprehensive overview of the main results and developments in this rapidly evolving field. Volume A covers models of computation, complexity theory, data structures, and efficient computation in many recognized subdisciplines of theoretical computer science. Volume B takes up the theory of automata and rewriting systems, the foundations of modern programming languages, and logics for program specification and verification, and presents several studies on the theoretic modeling of advanced information processing. The two volumes contain thirty-seven chapters, with extensive chapter references and individual tables of contents for each chapter. There are 5,387 entry subject indexes that include notational symbols, and a list of contributors and affiliations in each volume.

This is a textbook on applied probability and statistics with computer science applications for students at the upper undergraduate level. It may also be used as a self study book for the practicing computer science professional. The successful first edition of this book proved extremely useful to students who need to use probability, statistics and queueing theory to solve problems in other fields, such as engineering, physics, operations research, and management science. The book has also been successfully used for courses in queueing theory for operations research students. This second edition includes a new chapter on regression as well as more than twice as many exercises at the end of each chapter. While the emphasis is the same as in the first edition, this new book makes more extensive use of available personal computer software, such as Minitab and Mathematica.

Here, the author, develops a type theory, studies its properties, and explains its uses in applications to computer science. In particular, type theory is shown to offer a powerful and uniform language for programming, program specification and development, and logical reasoning.

This book constitutes the refereed proceedings of the International Seminar on Proof Theory in Computer Science, PTCS 2001, held in Dagstuhl Castle, Germany, in October 2001. The 13 thoroughly revised full papers were carefully reviewed and selected for inclusion in the book. Among the topics addressed are higher type recursion, lambda calculus, complexity theory, transfinite induction, categories, induction-recursion, post-Turing analysis, natural deduction, implicit characterization, iterate logic, and Java programming.

On May 1, 2004, the world of theoretical computer science su?ered a stunning loss: Shimon Even passed away. Few computer scientists have had as long, s-tained, and in?uential a career as Shimon. Shimon Even was born in Tel-Aviv in 1935. He received a B.Sc. in Elect- cal Engineering from the Technion in 1959, an M.A. in Mathematics from the University of Northern Carolina in 1961, and a Ph.D. in Applied Mathematics from Harvard University in 1963. He held positions at the Technion (1964–67 and 1974–2003), Harvard University (1967–69), the Weizmann Institute (1969– 74), and the Tel-Aviv Academic College (2003-04). He visited many universities and research institutes, including Bell Laboratories, Boston University, Cornell, Duke, Lucent Technologies, MIT, Paderborn, Stanford, UC-Berkeley, USC and UT-Dallas. Shimon Even played a major role in establishing computer science education in Israel and led the development of academic programs in two major insti- tions: the Weizmann Institute and the Technion. In 1969 he established at the Weizmann the ?rst computer science education program in Israel, and led this program for ?ve years. In 1974 he joined the newly formed computer science department at the Technion and shaped its academic development for several decades. These two academic programs turned out to have a lasting impact on the evolution of computer science in Israel.

Theory and theoreticians have played a major role in computer science. Many insights into the nature of efficient computations were gained and theory was crucial for some of the most celebrated engineering triumphs of computer science (e.g., in compiler design, databases, multitask operating systems, to name just a few). Theoretical computer science (TCS) functions as a communication bridge between computer science and other subjects, notably, mathematics, linguistics, biology; it is a champion in developing unconventional models of computation (DNA, quantum). This book collects personal accounts and reflections of fourteen eminent scientists who have dedicated themselves to the craft of TCS. Contributions focus on authors specific interests, experiences, and reminiscences. The emerging picture, which is just one among other possible ones, should be a catalyst for further developments and continuations. Was most interested to learn about the project, which should be a worthwhile one." N. Chomsky, MIT. "The human story of creativity is inspiring and documents a very noble activity - the creation of knowledge in its most beautiful and useful form - the creation of a science. Supplying the technical and intellectual tools to probe some of the most fascinating questions about the nature of thought and intelligence, theoretical computer science is trying to grasp the limits of rational thought, the limits of knowable. This book will contribute to the understanding of the creation of a magnificent science." J. Hartmanis, NSF. "This is obviously an extremely worthwhile project." D. E. Knuth, Stanford University.

In Memory of Dieter Rötting. 24.8.1937 - 4.6.1984. On the Occasion of the 50th Anniversary of His Birth

A computational perspective on partial order and lattice theory, focusing on algorithms and their applications This book provides a uniform treatment of the theory and applications of lattice theory. The applications covered include tracking dependency in distributed systems, combinatorics, detecting global predicates in distributed systems, set families, and integer partitions. The book presents algorithmic proofs of theorems whenever possible. These proofs are written in the calculational style advocated by Dijkstra, with arguments explicitly spelled out step by step. The author's intent is for readers to learn not only the proofs, but the heuristics that guide said proofs. Introduction to Lattice Theory with Computer Science Applications: Examines; posets, Dilworth's theorem, merging algorithms, lattices, lattice completion, morphisms, modular and distributive lattices, slicing, interval orders, tractable posets, lattice enumeration algorithms, and dimension theory Provides end of chapter exercises to help readers retain newfound knowledge on each subject Includes supplementary material at www.ece.utexas.edu/~garg Introduction to Lattice Theory with Computer Science Applications is written for students of computer science, as well as practicing mathematicians.

A wide coverage of topics in category theory and computer science is developed in this text, including introductory treatments of cartesian closed categories, sketches and elementary categorical model theory, and triples. Over 300 exercises are included.

This book provides a good introduction to the classical elementary number theory and the modern algorithmic number theory, and their applications in computing and information technology, including computer systems design, cryptography and network security. In this second edition proofs of many theorems have been provided, further additions and corrections were made.

An introduction to computational complexity theory, its connections and interactions with mathematics, and its central role in the natural and social sciences, technology, and philosophy Mathematics and Computation provides a broad, conceptual overview of computational complexity theory—the mathematical study of efficient computation. With important practical applications to computer science and industry, computational complexity theory has evolved into a highly interdisciplinary field, with strong links to most mathematical areas and to a growing number of scientific endeavors. Avi Wigderson takes a sweeping survey of complexity theory, emphasizing the field's insights and challenges. He explains the ideas and motivations leading to key models, notions, and results. In particular, he looks at algorithms and complexity, computations and proofs, randomness and interaction, quantum and arithmetic computation, and cryptography and learning, all as parts of a cohesive whole with numerous cross-influences. Wigderson illustrates the immense breadth of the field, its beauty and richness, and its diverse and growing interactions with other areas of mathematics. He ends with a comprehensive look at the theory of computation, its methodology and aspirations, and the unique and fundamental ways in which it has shaped and will further shape science, technology, and society. For further reading, an extensive bibliography is provided for all topics covered. Mathematics and Computation is useful for undergraduate and graduate students in mathematics, computer science, and related fields, as well as researchers and teachers in these fields. Many parts require little background, and serve as an invitation to newcomers seeking an introduction to the theory of computation. Comprehensive coverage of computational complexity theory, and beyond High-level, intuitive exposition, which brings conceptual clarity to this central and dynamic scientific discipline Historical accounts of the evolution and motivations of central concepts and models A broad view of the theory of computation's influence on science, technology, and society Extensive bibliography

This introductory text provides both a foundation in a popular programming language (Turbo PASCAL) and an introduction to the principles and applications of the field. It stresses applications that demonstrate computers' many roles in our lives

This an introduction to the theory of computational learning.

Basic Category Theory for Computer Scientists provides a straightforward presentation of the basic constructions and terminology of category theory, including limits, functors, natural transformations, adjoints, and cartesian closed categories. Category theory is a branch of pure mathematics that is becoming an increasingly important tool in theoretical computer science, especially in programming language semantics, domain theory, and concurrency, where it is already a standard language of discourse. Assuming a minimum of mathematical preparation, Basic Category Theory for Computer Scientists provides a straightforward presentation of the basic constructions and terminology of category theory, including limits, functors, natural transformations, adjoints, and cartesian closed categories. Four case studies illustrate applications of category theory to programming language design, semantics, and the solution of recursive domain equations. A brief literature survey offers suggestions for further study in more advanced texts. Contents Tutorial • Applications • Further Reading

Now you can clearly present even the most complex computational theory topics to your students with Sipser's distinct, market-leading INTRODUCTION TO THE THEORY OF COMPUTATION, 3E. The number one choice for today's computational theory course, this highly anticipated revision retains the unmatched clarity and thorough coverage that make it a leading text for upper-level undergraduate and introductory graduate students. This edition continues author Michael Sipser's well-known, approachable style with timely revisions, additional exercises, and more memorable examples in key areas. A new first-of-its-kind theoretical treatment of deterministic context-free languages is ideal for a better understanding of parsing and LR(k) grammars. This edition's refined presentation ensures a trusted accuracy and clarity that make the challenging study of computational theory accessible and intuitive to students while maintaining the subject's rigor and formalism. Readers gain a solid understanding of the fundamental mathematical properties of computer hardware, software, and applications with a blend of practical and philosophical coverage and mathematical treatments, including advanced theorems and proofs. INTRODUCTION TO THE THEORY OF COMPUTATION, 3E's comprehensive coverage makes this an ideal ongoing reference tool for those studying theoretical computing. Important Notice: Media content referenced within the product description or the product text may not be available in the ebook version.

An up-to-date and comprehensive account of set-oriented symbolic manipulation and automated reasoning methods. This book is of interest to graduates and researchers in theoretical computer science and computational logic and automated reasoning.

Theory of Computer ScienceAutomata, Languages and ComputationPHI Learning Pvt. Ltd.

Finite model theory,as understoodhere, is an areaof mathematicallogic that has developed in close connection with applications to computer science, in particular the theory of computational complexity and database theory. One of the fundamental insights of mathematical logic is that our understanding of mathematical phenomena is enriched by elevating the languages we use to describe mathematical structures to objects of explicit study. If mathematics is the science of patterns, then the media through which we discern patterns, as well as the structures in which we discern them, command our attention. It isthis aspect oflogicwhichis mostprominentin model theory,“thebranchof mathematical logic which deals with the relation between a formal language and its interpretations”. No wonder, then, that mathematical logic, and ?nite model theory in particular, should ?nd manifold applications in computer science: from specifying programs to querying databases, computer science is rife with phenomena whose understanding requires close attention to the interaction between language and structure. This volume gives a broadoverviewof some central themes of ?nite model theory: expressive power, descriptive complexity, and zero–one laws, together with selected applications to database theory and arti?cial intelligence, es- cially constraint databases and constraint satisfaction problems. The ?nal chapter provides a concise modern introduction to modal logic,which emp- sizes the continuity in spirit and technique with ?nite model theory. This book constitutes the refereed proceedings of the 7th

International Conference on Category Theory and Computer Science, CTCS'97, held in Santa Margheria Ligure, Italy, in September 1997. Category theory attracts interest in the theoretical computer science community because of its ability to establish connections between different areas in computer science and mathematics and to provide a few generic principles for organizing mathematical theories. This book presents a selection of 15 revised full papers together with three invited contributions. The topics addressed include reasoning principles for types, rewriting, program semantics, and structuring of logical systems.
This Third Edition, in response to the enthusiastic reception given by academia and students to the previous edition, offers a cohesive presentation of all aspects of theoretical computer science, namely automata, formal languages, computability, and complexity. Besides, it includes coverage of mathematical preliminaries. NEW TO THIS EDITION • Expanded sections on pigeonhole principle and the principle of induction (both in Chapter 2) • A rigorous proof of Kleene's theorem (Chapter 5) • Major changes in the chapter on Turing machines (TMs) – A new section on high-level description of TMs – Techniques for the construction of TMs – Multitape TM and nondeterministic TM • A new chapter (Chapter 10) on decidability and recursively enumerable languages • A new chapter (Chapter 12) on complexity theory and NP-complete problems • A section on quantum computation in Chapter 12. • KEY FEATURES • Objective-type questions in each chapter—with answers provided at the

end of the book. • Eighty-three additional solved examples—added as Supplementary Examples in each chapter. • Detailed solutions at the end of the book to chapter-end exercises. The book is designed to meet the needs of the undergraduate and postgraduate students of computer science and engineering as well as those of the students offering courses in computer applications. The past 50 years have witnessed a revolution in computing and related communications technologies. The contributions of industry and university researchers to this revolution are manifest; less widely recognized is the major role the federal government played in launching the computing revolution and sustaining its momentum. Funding a Revolution examines the history of computing since World War II to elucidate the federal government's role in funding computing research, supporting the education of computer scientists and engineers, and equipping university research labs. It reviews the economic rationale for government support of research, characterizes federal support for computing research, and summarizes key historical advances in which government-sponsored research played an important role. Funding a Revolution contains a series of case studies in relational databases, the Internet, theoretical computer science, artificial intelligence, and virtual reality that demonstrate the complex interactions among government, universities, and industry that have driven the field. It offers a series of lessons that identify factors contributing to the success of the nation's computing enterprise and the government's role within it. Named a Notable Book in the 21st Annual Best of

Computing list by the ACM! Robert Sedgewick and Kevin Wayne's Computer Science: An Interdisciplinary Approach is the ideal modern introduction to computer science with Java programming for both students and professionals. Taking a broad, applications-based approach, Sedgewick and Wayne teach through important examples from science, mathematics, engineering, finance, and commercial computing. The book demystifies computation, explains its intellectual underpinnings, and covers the essential elements of programming and computational problem solving in today's environments. The authors begin by introducing basic programming elements such as variables, conditionals, loops, arrays, and I/O. Next, they turn to functions, introducing key modular programming concepts, including components and reuse. They present a modern introduction to object-oriented programming, covering current programming paradigms and approaches to data abstraction. Building on this foundation, Sedgewick and Wayne widen their focus to the broader discipline of computer science. They introduce classical sorting and searching algorithms, fundamental data structures and their application, and scientific techniques for assessing an implementation's performance. Using abstract models, readers learn to answer basic questions about computation, gaining insight for practical application. Finally, the authors show how machine architecture links the theory of computing to real computers, and to the field's history and evolution. For each concept, the authors present all the information readers need to build confidence, together

with examples that solve intriguing problems. Each chapter contains question-and-answer sections, self-study drills, and challenging problems that demand creative solutions. Companion web site (introcs.cs.princeton.edu/java) contains Extensive supplementary information, including suggested approaches to programming assignments, checklists, and FAQs Graphics and sound libraries Links to program code and test data Solutions to selected exercises Chapter summaries Detailed instructions for installing a Java programming environment Detailed problem sets and projects Companion 20-part series of video lectures is available at informit.com/title/9780134493831
This textbook is uniquely written with dual purpose. It cover cores material in the foundations of computing for graduate students in computer science and also provides an introduction to some more advanced topics for those intending further study in the area. This innovative text focuses primarily on computational complexity theory: the classification of computational problems in terms of their inherent complexity. The book contains an invaluable collection of lectures for first-year graduates on the theory of computation. Topics and features include more than 40 lectures for first year graduate students, and a dozen homework sets and exercises.
This book constitutes the refereed proceedings of the Second IFIP WG 1.8 International Conference on Topics in Theoretical Computer Science, TTCS 2017, held in Tehran, Iran, in September 2017. The 8 papers presented in this volume were carefully reviewed and selected from 20 submissions. They were organized in

topical sections named: algorithms and complexity; and logic, semantics, and programming theory.

This revised and extensively expanded edition of Computability and Complexity Theory comprises essential materials that are core knowledge in the theory of computation. The book is self-contained, with a preliminary chapter describing key mathematical concepts and notations. Subsequent chapters move from the qualitative aspects of classical computability theory to the quantitative aspects of complexity theory. Dedicated chapters on undecidability, NP-completeness, and relative computability focus on the limitations of computability and the distinctions between feasible and intractable. Substantial new content in this edition includes: a chapter on nonuniformity studying Boolean circuits, advice classes and the important result of Karp?Lipton. a chapter studying properties of the fundamental probabilistic complexity classes a study of the alternating Turing machine and uniform circuit classes. an introduction of counting classes, proving the famous results of Valiant and Vazirani and of Toda a thorough treatment of the proof that IP is identical to PSPACE With its accessibility and well-devised organization, this text/reference is an excellent resource and guide for those looking to develop a solid grounding in the theory of computing. Beginning graduates, advanced undergraduates, and professionals involved in theoretical computer science, complexity theory, and computability will find the book an essential and practical learning tool. Topics and features: Concise, focused materials cover the most fundamental concepts and results in the field of modern complexity theory, including the theory of NP-completeness, NP-hardness, the polynomial hierarchy, and complete problems for other complexity classes Contains information that otherwise exists only in research literature and presents it in a unified, simplified manner Provides key mathematical background information, including sections on logic and number theory and algebra Supported by numerous exercises and supplementary problems for reinforcement and self-study purposes

Games provide mathematical models for interaction. Numerous tasks in computer science can be formulated in game-theoretic terms. This fresh and intuitive way of thinking through complex issues reveals underlying algorithmic questions and clarifies the relationships between different domains. This collection of lectures, by specialists in the field, provides an excellent introduction to various aspects of game theory relevant for applications in computer science that concern program design, synthesis, verification, testing and design of multi-agent or distributed systems. Originally devised for a Spring School organised by the GAMES Networking Programme in 2009, these lectures have since been revised and expanded, and range from tutorials concerning fundamental notions and methods to more advanced presentations of current research topics. This volume is a valuable guide to current research on game-based methods in computer science for undergraduate and graduate students. It will also interest researchers working in mathematical logic, computer science and game theory.

The foundation of computer science is built upon the following questions: What is an algorithm? What can be computed and what cannot be computed? What does it mean for a function to be computable? How does computational power depend upon programming constructs? Which algorithms can be considered feasible? For more than 70 years, computer scientists are searching for answers to such qu- tions. Their ingenious techniques used in answering these questions form the theory of computation. Theory of computation deals with the most fundamental ideas of computer s- ence in an abstract but easily understood form. The notions and techniques employed are widely spread across various topics and are found in almost every branch of c- puter science. It has thus become more than a necessity to revisit the foundation, learn the techniques, and apply them with con?dence. Overview and Goals This book is about this solid, beautiful, and pervasive foundation of computer s-ence. It introduces the fundamental notions, models, techniques, and results that form the basic paradigms of computing. It gives an introduction to the concepts and mathematics that computer scientists of our day use to model, to argue about, and to predict the behavior of algorithms and computation. The topics chosen here have shown remarkable persistence over the years and are very much in current use.

This volume brings together the work of several prominent researchers who have collaborated with Janusz Brzozowski, or worked in topics he developed, in the areas of regular languages, syntactic semigroups of formal languages, the dot-depth hierarchy, and formal modeling of circuit testing and software specification using automata theory.

Because of its inherent simplicity, graph theory has a wide range of applications in engineering, and in physical sciences. It has of course uses in social sciences, in linguistics and in numerous other areas. In fact, a graph can be used to represent almost any physical situation involving discrete objects and the relationship among them. Now with the solutions to engineering and other problems becoming so complex leading to larger graphs, it is virtually difficult to analyze without the use of computers. This book is recommended in IIT Kharagpur, West Bengal for B.Tech Computer Science, NIT Arunachal

Pradesh, NIT Nagaland, NIT Agartala, NIT Silchar, Gauhati University, Dibrugarh University, North Eastern Regional Institute of Management, Assam Engineering College, West Bengal Univerity of Technology (WBUT) for B.Tech, M.Tech Computer Science, University of Burdwan, West Bengal for B.Tech. Computer Science, Jadavpur University, West Bengal for M.Sc. Computer Science, Kalyani College of Engineering, West Bengal for B.Tech. Computer Science. Key Features: This book provides a rigorous yet informal treatment of graph theory with an emphasis on computational aspects of graph theory and graph-theoretic algorithms. Numerous applications to actual engineering problems are incorpo-rated with software design and optimization topics.

The author examines logic and methodology of design from the perspective of computer science. Computers provide the context for this examination both by discussion of the design process for hardware and software systems and by consideration of the role of computers in design in general. The central question posed by the author is whether or not we can construct a theory of design.

Type theory is one of the most important tools in the design of higher-level programming languages, such as ML. This book introduces and teaches its techniques by focusing on one particularly neat system and studying it in detail. By concentrating on the principles that make the theory work in practice, the author covers all the key ideas without getting involved in the complications of more advanced systems. This book takes a type-assignment approach to type theory, and the system considered is the simplest polymorphic one. The author covers all the basic ideas, including the system's relation to propositional logic, and gives a careful treatment of the type-checking algorithm that lies at the heart of every such system. Also featured are two other interesting algorithms that until now have been buried in inaccessible technical literature. The mathematical presentation is rigorous but clear, making it the first book at this level that can be used as an introduction to type theory for computer scientists.

In 1936, when he was just twenty-four years old, Alan Turing wrote a remarkable paper in which he outlined the theory of computation, laying out the ideas that underlie all modern computers. This groundbreaking and powerful theory now forms the basis of computer science. In Turing's Vision, Chris Bernhardt explains the theory, Turing's most important contribution, for the general reader. Bernhardt argues that the strength of Turing's theory is its simplicity, and that, explained in a straightforward manner, it is eminently understandable by the nonspecialist. As Marvin Minsky writes, "The sheer simplicity of the theory's foundation and extraordinary short path from this foundation to its logical and surprising conclusions give the theory a mathematical beauty that alone guarantees it a permanent place in computer theory." Bernhardt begins with the foundation and systematically builds to the surprising conclusions. He also views Turing's theory in the context of mathematical history, other views of computation (including those of Alonzo Church), Turing's later work, and the birth of the modern computer. In the paper, "On Computable Numbers, with an Application to the Entscheidungsproblem," Turing thinks carefully about how humans perform computation, breaking it down into a sequence of steps, and then constructs theoretical machines capable of performing each step. Turing wanted to show that there were problems that were beyond any computer's ability to solve; in particular, he wanted to find a decision problem that he could prove was undecidable. To explain Turing's ideas, Bernhardt examines three well-known decision problems to explore the concept of undecidability; investigates theoretical computing machines, including Turing machines; explains universal machines; and proves that certain problems are undecidable, including Turing's problem concerning computable numbers.

This 1991 book presents a novel, yet systematic and practical way of implementing concepts so that they become useful in the design and analysis of computer systems.

The Handbook of Theoretical Computer Scienceprovides professionals and students with a comprehensive overview of the main results and developments in this rapidly evolving field. Volume A covers models of computation, complexity theory, data structures, and efficient computation in many recognized subdisciplines of theoretical computer science. Volume B takes up the theory of automata and rewriting systems, the foundations of modern programming languages, and logics for program specification and verification, and presents several studies on the theoretic modeling of advanced information processing. The two volumes contain thirty-seven chapters, with extensive chapter references and individual tables of contents for each chapter. There are 5,387 entry subject indexes that include notational symbols, and a list of contributors and affiliations in each volume.

This book constitutes the proceedings of the 9th International Computer Science Symposium in Russia, CSR 2014, held in Moscow, Russia, in June 2014. The 27 full papers presented in this volume were carefully reviewed and selected from 76 submissions. In addition the book contains 4 invited lectures. The scope of the proposed topics is quite broad and covers a wide range of areas in theoretical computer science and its applications.

Computer science seeks to provide a scientific basis for the study of inform a tion processing, the solution of problems by algorithms, and the design and programming of computers. The last forty years have seen increasing sophistication in the science, in the microelectronics which has made machines of staggering complexity economically feasible, in the advances in programming methodology which allow immense programs to be designed with increasing speed and reduced error, and in the development of mathematical techniques to allow the rigorous specification of program, process, and machine. The present volume is one of a series, The AKM Series in Theoretical Computer Science, designed to make key mathe matical developments in computer science readily accessible to under graduate and beginning graduate students. Specifically, this volume takes readers with little or no mathematical background beyond high school algebra, and gives them a taste of a number of topics in theoretical computer science while laying the mathematical foundation for the later, more detailed, study of such topics as formal language theory, computability theory, programming language semantics, and the study of program verification and correctness. Chapter 1 introduces the basic concepts of set theory, with special emphasis on functions and relations, using a simple algorithm to provide motivation. Chapter 2 presents the notion of inductive proof and gives the reader a good grasp on one of the most important notions of computer science: the recursive definition of functions and data structures.

This text addresses some theoretical issues surrounding computer science. It provides an introduction to the theory of computation, and covers programming languages, finite state machines, grammars, Boolean circuits, computational complexity, feasible problems, and intractable problems.

There are several theories of programming. The first usable theory, often called "Hoare's Logic", is still probably the most widely known. In it, a specification is a pair of predicates: a precondition and postcondition (these and all technical terms will be defined in due course). Another popular and closely related theory by Dijkstra uses the weakest precondition predicate transformer, which is a function from programs and postconditions to preconditions. lones's Vienna Development Method has been used to advantage in some industries; in it, a specification is a pair of predicates (as in Hoare's Logic), but the second predicate is a relation. Temporal Logic is yet another formalism that introduces some special operators and quantifiers to describe some aspects of computation. The theory in this book is simpler than any of those just mentioned. In it, a specification is just a boolean expression. Refinement is just ordinary implication. This theory is also more general than those just mentioned, applying to both terminating and nonterminating computation, to both sequential and parallel computation, to both stand-alone and interactive computation. And it includes time bounds, both for algorithm classification and for tightly constrained real-time applications.