Preface

Students of science and engineering are required to study mathematics during their first years at a university. Traditionally, they concentrate on calculus, linear algebra and differential equations, but in computer science and engineering, logic, combinatorics and discrete mathematics are more appropriate. Logic is particularly important because it is the mathematical basis of software: it is used to formalize the semantics of programming languages and the specification of programs, and to verify the correctness of programs.

Mathematical Logic for Computer Science is a *mathematics* textbook, just as a first-year calculus text is a mathematics textbook. A scientist or engineer needs more than just a facility for manipulating formulas and a firm foundation in mathematics is an excellent defense against technological obsolescence. Tempering this requirement for mathematical competence is the realization that applications use only a fraction of the theoretical results. Just as the theory of calculus can be taught to students of engineering without the full generality of measure theory, students of computer science need not be taught the full generality of uncountable structures. Fortunately (as shown by Raymond M. Smullyan), tableaux provide an elegant way to teach mathematical logic that is both theoretically sound and yet sufficiently elementary for the undergraduate.

Audience

The book is intended for undergraduate computer science students. No specific mathematical knowledge is assumed aside from informal set theory which is summarized in an appendix, but elementary knowledge of concepts from computer science (graphs, languages, programs) is used.

Organization

The book can be divided into four parts. Within each part the chapters should be read sequentially; the prerequisites between the parts are described here.

Propositional Logic: Chapter 2 is on the syntax and semantics of propositional logic. It introduces the method of semantic tableaux as a decision procedure for the logic. This chapter is an essential prerequisite for reading the rest of the book. Chapter 3 introduces deductive systems (axiomatic proof systems). The next three chapters present techniques that are used in practice for tasks such as automatic theorem proving and program verification: Chap. 4 on resolution, Chap. 5 on binary decision diagrams and Chap. 6 on SAT solvers.

First-Order Logic: The same progression is followed for first-order logic. There are two chapters on the basic theory of the logic: Chap. 7 on syntax, semantics and semantic tableaux, followed by Chap. 8 on deductive systems. Important application of first-order logic are automatic theorem proving using resolution (Chap. 10) and logic programming (Chap. 11). These are preceded by Chap. 9 which introduces an essential extension of the logic to terms and functions. Chapter 12 surveys fundamental theoretical results in first-order logic. The chapters on first-order logic assume as prerequisites the corresponding chapters on propositional logic; for example, you should read Chap. 4 on resolution in the propositional logic before the corresponding Chap. 10 in first-order logic.

Temporal Logic: Again, the same progression is followed: Chap. 13 on syntax, semantics and semantic tableaux, followed by Chap. 14 on deductive systems. The prerequisites are the corresponding chapters on propositional logic since first-order temporal logic is not discussed.

Program Verification: One of the most important applications of mathematical logic in computer science is in the field of program verification. Chapter 15 presents a deductive system for the verification of sequential programs; the reader should have mastered Chap. 3 on deductive systems in propositional logic before reading this chapter. Chapter 16 is highly dependent on earlier chapters: it includes deductive proofs, the use of temporal logic, and implementations using binary decision diagrams and satisfiability solvers.

Supplementary Materials

Slides of the diagrams and tables in the book (in both PDF and LaTeX) can be downloaded from http://www.springer.com/978-1-4471-4128-0, which also contains instructions for obtaining the answers to the exercises (qualified instructors only). The source code and documentation of Prolog programs for most of the algorithms in the book can be downloaded from http://code.google.com/p/mlcs/.

Third Edition

The third edition has been totally rewritten for clarity and accuracy. In addition, the following major changes have been made to the content:

- The discussion of logic programming has been shortened somewhat and the Prolog programs and their documentation have been removed to a freely available archive.
- The chapter on the $\mathscr{Z}$ notation has been removed because it was difficult to do justice to this important topic in a single chapter.
- The discussion of model checking in Chap. 16 has been significantly expanded since model checking has become a widely used technique for program verification.
- Chapter 6 has been added to reflect the growing importance of SAT solvers in all areas of computer science.

Notation

If and only if is abbreviated *iff*. Definitions by convention use *iff* to emphasize that the definition is restrictive. For example: A natural number is even iff it can be expressed as $2k$ for some natural number k. In the definition, *iff* means that numbers expressed as $2k$ are even and these are the only even numbers.

Definitions, theorems and examples are consecutively numbered within each chapter to make them easy to locate. The end of a definition, example or proof is denoted by ■.

Advanced topics and exercises, as well as topics outside the mainstream of the book, are marked with an asterisk.

Acknowledgments

I am indebted to Jørgen Villadsen for his extensive comments on the second edition which materially improved the text. I would like to thank Joost-Pieter Katoen and Doron Peled for reviewing parts of the manuscript. I would also like to thank Helen Desmond, Ben Bishop and Beverley Ford of Springer for facilitating the publication of the book.

Rehovot, Israel Mordechai (Moti) Ben-Ari