

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 \mathcal{L} 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

Contents

1	Introduction	1
1.1	The Origins of Mathematical Logic	1
1.2	Propositional Logic	2
1.3	First-Order Logic	3
1.4	Modal and Temporal Logics	4
1.5	Program Verification	5
1.6	Summary	5
1.7	Further Reading	6
1.8	Exercise	6
	References	6
2	Propositional Logic: Formulas, Models, Tableaux	7
2.1	Propositional Formulas	7
2.2	Interpretations	16
2.3	Logical Equivalence	21
2.4	Sets of Boolean Operators *	26
2.5	Satisfiability, Validity and Consequence	29
2.6	Semantic Tableaux	33
2.7	Soundness and Completeness	39
2.8	Summary	44
2.9	Further Reading	45
2.10	Exercises	45
	References	47
3	Propositional Logic: Deductive Systems	49
3.1	Why Deductive Proofs?	49
3.2	Gentzen System \mathcal{G}	51
3.3	Hilbert System \mathcal{H}	55
3.4	Derived Rules in \mathcal{H}	58
3.5	Theorems for Other Operators	62
3.6	Soundness and Completeness of \mathcal{H}	64

3.7	Consistency	66
3.8	Strong Completeness and Compactness *	67
3.9	Variant Forms of the Deductive Systems *	68
3.10	Summary	71
3.11	Further Reading	71
3.12	Exercises	72
	References	73
4	Propositional Logic: Resolution	75
4.1	Conjunctive Normal Form	75
4.2	Clausal Form	77
4.3	Resolution Rule	80
4.4	Soundness and Completeness of Resolution *	82
4.5	Hard Examples for Resolution *	88
4.6	Summary	92
4.7	Further Reading	92
4.8	Exercises	92
	References	93
5	Propositional Logic: Binary Decision Diagrams	95
5.1	Motivation Through Truth Tables	95
5.2	Definition of Binary Decision Diagrams	97
5.3	Reduced Binary Decision Diagrams	98
5.4	Ordered Binary Decision Diagrams	102
5.5	Applying Operators to BDDs	104
5.6	Restriction and Quantification *	107
5.7	Summary	109
5.8	Further Reading	110
5.9	Exercises	110
	References	110
6	Propositional Logic: SAT Solvers	111
6.1	Properties of Clausal Form	111
6.2	Davis-Putnam Algorithm	115
6.3	DPLL Algorithm	116
6.4	An Extended Example of the DPLL Algorithm	117
6.5	Improving the DPLL Algorithm	122
6.6	Stochastic Algorithms	125
6.7	Complexity of SAT *	126
6.8	Summary	128
6.9	Further Reading	128
6.10	Exercises	128
	References	129
7	First-Order Logic: Formulas, Models, Tableaux	131
7.1	Relations and Predicates	131
7.2	Formulas in First-Order Logic	133

7.3	Interpretations	136
7.4	Logical Equivalence	140
7.5	Semantic Tableaux	143
7.6	Soundness and Completion of Semantic Tableaux	150
7.7	Summary	153
7.8	Further Reading	153
7.9	Exercises	153
	References	154
8	First-Order Logic: Deductive Systems	155
8.1	Gentzen System \mathcal{G}	155
8.2	Hilbert System \mathcal{H}	158
8.3	Equivalence of \mathcal{H} and \mathcal{G}	160
8.4	Proofs of Theorems in \mathcal{H}	161
8.5	The C-Rule *	163
8.6	Summary	165
8.7	Further Reading	165
8.8	Exercises	165
	References	166
9	First-Order Logic: Terms and Normal Forms	167
9.1	First-Order Logic with Functions	167
9.2	PCNF and Clausal Form	172
9.3	Herbrand Models	177
9.4	Herbrand's Theorem *	180
9.5	Summary	182
9.6	Further Reading	182
9.7	Exercises	182
	References	183
10	First-Order Logic: Resolution	185
10.1	Ground Resolution	185
10.2	Substitution	187
10.3	Unification	189
10.4	General Resolution	195
10.5	Soundness and Completeness of General Resolution *	198
10.6	Summary	202
10.7	Further Reading	202
10.8	Exercises	202
	References	203
11	First-Order Logic: Logic Programming	205
11.1	From Formulas in Logic to Logic Programming	205
11.2	Horn Clauses and SLD-Resolution	209
11.3	Search Rules in SLD-Resolution	213
11.4	Prolog	216
11.5	Summary	220

11.6	Further Reading	221
11.7	Exercises	221
	References	222
12	First-Order Logic: Undecidability and Model Theory *	223
12.1	Undecidability of First-Order Logic	223
12.2	Decidable Cases of First-Order Logic	226
12.3	Finite and Infinite Models	227
12.4	Complete and Incomplete Theories	228
12.5	Summary	229
12.6	Further Reading	229
12.7	Exercises	230
	References	230
13	Temporal Logic: Formulas, Models, Tableaux	231
13.1	Introduction	231
13.2	Syntax and Semantics	233
13.3	Models of Time	237
13.4	Linear Temporal Logic	240
13.5	Semantic Tableaux	244
13.6	Binary Temporal Operators *	258
13.7	Summary	260
13.8	Further Reading	261
13.9	Exercises	261
	References	262
14	Temporal Logic: A Deductive System	263
14.1	Deductive System \mathcal{L}	263
14.2	Theorems of \mathcal{L}	264
14.3	Soundness and Completeness of \mathcal{L} *	269
14.4	Axioms for the Binary Temporal Operators *	271
14.5	Summary	271
14.6	Further Reading	272
14.7	Exercises	272
	References	272
15	Verification of Sequential Programs	273
15.1	Correctness Formulas	274
15.2	Deductive System \mathcal{HL}	275
15.3	Program Verification	277
15.4	Program Synthesis	279
15.5	Formal Semantics of Programs *	283
15.6	Soundness and Completeness of \mathcal{HL} *	289
15.7	Summary	293
15.8	Further Reading	293
15.9	Exercises	293
	References	295

- 16 Verification of Concurrent Programs 297**
 - 16.1 Definition of Concurrent Programs 298
 - 16.2 Formalization of Correctness 300
 - 16.3 Deductive Verification of Concurrent Programs 303
 - 16.4 Programs as Automata 307
 - 16.5 Model Checking of Invariance Properties 311
 - 16.6 Model Checking of Liveness Properties 314
 - 16.7 Expressing an LTL Formula as an Automaton 315
 - 16.8 Model Checking Using the Synchronous Automaton 317
 - 16.9 Branching-Time Temporal Logic * 319
 - 16.10 Symbolic Model Checking * 322
 - 16.11 Summary 323
 - 16.12 Further Reading 324
 - 16.13 Exercises 324
 - References 325
- Appendix Set Theory 327**
 - A.1 Finite and Infinite Sets 327
 - A.2 Set Operators 328
 - A.3 Sequences 330
 - A.4 Relations and Functions 331
 - A.5 Cardinality 333
 - A.6 Proving Properties of Sets 335
 - References 336
- Index of Symbols 337**
- Name Index 339**
- Subject Index 341**

Chapter 1

Introduction

1.1 The Origins of Mathematical Logic

Logic formalizes valid methods of reasoning. The study of logic was begun by the ancient Greeks whose educational system stressed competence in reasoning and in the use of language. Along with rhetoric and grammar, logic formed part of the *trivium*, the first subjects taught to young people. Rules of logic were classified and named. The most widely known set of rules are the *syllogisms*; here is an example of one form of syllogism:

Premise All rabbits have fur.

Premise Some pets are rabbits.

Conclusion Some pets have fur.

If both premises are true, the rules ensure that the conclusion is true.

Logic must be formalized because reasoning expressed in informal natural language can be flawed. A clever example is the following ‘syllogism’ given by Smullyan (1978, p. 183):

Premise Some cars rattle.

Premise My car is some car.

Conclusion My car rattles.

The formalization of logic began in the nineteenth century as mathematicians attempted to clarify the foundations of mathematics. One trigger was the discovery of non-Euclidean geometries: replacing Euclid’s parallel axiom with another axiom resulted in a different theory of geometry that was just as consistent as that of Euclid. Logical systems—axioms and rules of inference—were developed with the understanding that different sets of axioms would lead to different theorems. The questions investigated included:

Consistency A logical system is consistent if it is impossible to prove both a formula and its negation.

Independence The axioms of a logical system are independent if no axiom can be proved from the others.

Soundness All theorems that can be proved in the logical system are true.

Completeness All true statements can be proved in the logical system.

Clearly, these questions will only make sense once we have formally defined the central concepts of *truth* and *proof*.

During the first half of the twentieth century, logic became a full-fledged topic of modern mathematics. The framework for research into the foundations of mathematics was called *Hilbert's program*, (named after the great mathematician David Hilbert). His central goal was to prove that mathematics, starting with arithmetic, could be axiomatized in a system that was both consistent and complete. In 1931, Kurt Gödel showed that this goal cannot be achieved: any consistent axiomatic system for arithmetic is incomplete since it contains true statements that cannot be proved within the system.

In the second half of the twentieth century, mathematical logic was applied in computer science and has become one of its most important theoretical foundations. Problems in computer science have led to the development of many new systems of logic that did not exist before or that existed only at the margins of the classical systems. In the remainder of this chapter, we will give an overview of systems of logic relevant to computer science and sketch their applications.

1.2 Propositional Logic

Our first task is to formalize the concept of the *truth* of a statement. Every statement is assigned one of two values, conventionally called *true* and *false* or *T* and *F*. These should be considered as arbitrary symbols that could easily be replaced by any other pair of symbols like 1 and 0 or even ♣ and ♠.

Our study of logic commences with the study of *propositional logic* (also called the *propositional calculus*). The *formulas* of the logic are built from *atomic propositions*, which are statements that have no internal structure. Formulas can be combined using *Boolean operators*. These operators have conventional names derived from natural language (*and*, *or*, *implies*), but they are given a formal meaning in the logic. For example, the Boolean operator *and* is defined as the operator that gives the value *true* if and only if applied to two formulas whose values are *true*.

Example 1.1 The statements ‘one plus one equals two’ and ‘Earth is farther from the sun than Venus’ are both true statements; therefore, by definition, so is the following statement:

‘one plus one equals two’ *and* ‘Earth is farther from the sun than Venus’.

Since ‘Earth is farther from the sun than Mars’ is a false statement, so is:

‘one plus one equals two’ *and* ‘Earth is farther from the sun than Mars’. ■

Rules of *syntax* define the legal structure of formulas in propositional logic. The *semantics*—the meaning of formulas—is defined by *interpretations*, which assign