

Richard Lassaigne, Michel de Rougemont

Logic and Complexity

Contents

Introduction	1
Part 1. Basic model theory and computability	3
Chapter 1. Propositional Logic	5
1. Propositional Language	5
1.1. Construction of formulas	5
1.2. Proof by induction	7
1.3. Decomposition of a formula	7
2. Semantics	9
2.1. Tautologies, Equivalent formulas	10
2.2. Logical Consequence	11
2.3. Value of a formula and substitution	11
2.4. Complete systems of connectives	14
3. Normal Forms	15
3.1. Disjunctive and conjunctive normal forms	15
3.2. Functions associated to formulas	15
3.3. Transformation methods	17
3.4. Clausal form	18
3.5. Ordered Binary Decision Diagrams	19
4. Exercises	23
Chapter 2. Deduction systems	25
1. Examples of tableaux	25
2. Tableaux method	28
2.1. Trees	28
2.2. Construction of tableaux	29
2.3. Development and closure	30
3. Completeness theorem	31
3.1. Provable formulas	31
3.2. Soundness	31
3.3. Completeness	32
4. Natural deduction	33
5. Compactness theorem	36
6. Exercises	38
Chapter 3. First Order Logic	41

1. First order languages	41
1.1. Construction of terms	42
1.2. Construction of formulas	43
1.3. Free and bound variables	44
2. Semantics	45
2.1. Structures and languages	45
2.2. Structures and satisfaction of formulas	46
2.3. Valid formulas. Equivalent formulas	48
2.4. Substitution	50
3. Prenex formulas. Skolem forms	51
3.1. Prenex formulas and normal forms	51
3.2. Skolem forms	53
4. Exercises	55
Chapter 4. Completeness of first order logic	57
1. Natural deduction system	57
2. Soundness	59
3. Completeness	59
4. Examples of theories	61
4.1. Successor function	62
4.2. Discrete successor function	62
4.3. Graphs	63
4.4. Robinson's Arithmetic	64
5. Exercises	65
Chapter 5. Models of computation	67
1. General model of computation	67
2. Turing machines	68
2.1. Deterministic Turing machines	68
2.2. Non deterministic Turing machines	71
2.3. Machines with multiple tapes	74
2.4. Turing machine with oracles	76
2.5. Alternating machines	76
3. Other sequential models of computation	78
3.1. Finite automata	78
3.2. RAM, Random Access Machines	80
4. Exercises	82
Chapter 6. Recursion and decidability	83
1. Recursive functions	83
1.1. Primitive recursive functions	83
1.2. Construction of primitive recursive functions	84
1.3. Coding sequences	86
1.4. Non primitive recursion	87
1.5. Recursive functions	88
1.6. Church's thesis	89

2. Recursion and Computation	90
2.1. Recursive functions are Turing computable	90
2.2. Turing computable functions are recursive	91
2.3. Universal Turing machine	93
3. Recursive systems	95
3.1. Equivalence with the recursive functions	98
3.2. Implementation of recursive functions	98
4. Recursion and decidability	101
4.1. Recursive and recursively enumerable sets	101
4.2. The halting problem	103
4.3. Reductions	104
4.4. The s - m - n and Rice's theorems	105
5. Exercises	107
Chapter 7. Incompleteness of Peano Arithmetic	109
1. Arithmetic theory and representable functions	109
1.1. Peano arithmetic	109
1.2. Arithmetical proofs	110
1.3. Non standard models of arithmetic	111
1.4. Representable functions	112
2. Coding arithmetical proofs	114
2.1. Coding terms and formulas	115
2.2. Coding sequences of formulas	116
2.3. Arithmetical theorems are recursively enumerable	116
3. Undecidability and incompleteness	118
4. The validity problem over finite structures	119
5. The arithmetical hierarchy	121
6. Exercises	128
Part 2. Descriptive Complexity	131
Chapter 8. Complexity: time and space	133
1. Complexity classes	134
1.1. Complexity of decision problems	134
1.2. Examples of decision problems	137
1.3. Complexity of search problems	138
2. Hierarchies for time and space	141
2.1. Simulation of several tapes	141
2.2. The role of constants	142
2.3. Relations between classes	142
2.4. Hierarchies	143
2.5. Non deterministic space classes	145
3. Other models and measures	149
3.1. Random access machines	149
3.2. Complexity on general structures	150
3.3. Average and smoothed complexities	150

3.4. Other complexities	151
4. Exercises	153
Chapter 9. First order definability	155
1. Explicit definitions	155
1.1. Definability on a class of structures	156
1.2. Partial elementary equivalence	157
1.3. Ehrenfeucht-Fraïssé games	158
1.4. Logical characterization of r -equivalence	160
1.5. Applications of Ehrenfeucht-Fraïssé games	162
1.6. Relational structures	164
2. 0-1 Law	166
2.1. Loś-Vaught's test	167
2.2. The theory of almost all relational structures	167
2.3. Convergence and 0-1 law	168
3. Implicit definitions	170
3.1. Craig's interpolation theorem	170
3.2. Beth's definability theorem	172
3.3. Finite structures	173
4. Exercises	175
Chapter 10. Inductive definitions and second order logic	177
1. Inductive definitions	177
1.1. Inductive systems	178
1.2. Fixed point logic	180
1.3. Substitution	181
1.4. Stage Comparison	182
1.5. Fixed point hierarchy	184
1.6. Datalog	186
2. Second order logic	188
2.1. Second order formulas and interpretation	188
2.2. Inductive definitions and second order logic	189
3. Exercises	192
Chapter 11. Time complexity : the classes P and NP	195
1. The class NP and NP-complete problems	195
1.1. Polynomial time reductions	196
1.2. NP-complete problems	196
1.3. Logical characterization of NP	203
1.4. Examples of definable NP problems	206
2. The logical characterizations of P and FP classes	208
2.1. P : the class of global inductive relations	208
2.2. FP : the class of global recursive functions	212
3. Exercises	215
Chapter 12. Models of parallel computations	217

1. The boolean circuits	217
2. Complexity in parallel models	221
2.1. Other circuits	223
3. Examples of NC problems	223
3.1. Matrix multiplication	223
3.2. Csanky's matrix inversion algorithm	224
4. The Parallel RAM	227
5. Exercises	230
Chapter 13. Space complexity: the classes L, FL, NL and PSPACE	233
1. The classes L, NL and FL	234
1.1. NL-complete problems	234
1.2. Alternation with space constraints	235
1.3. Logical characterization of NL	236
1.4. Logical characterization of FL	240
2. Sequential space and parallel time	245
2.1. The parallel time thesis	247
2.2. P-complete problems	247
3. The class PSPACE	248
3.1. The quantified boolean formulas and the class AP	248
3.2. Stochastic satisfiability	250
3.3. Problems with uncertainty	251
4. Exercises	253
Chapter 14. Definability of optimization and counting problems	255
1. Optimization problems	255
2. Counting problems	261
2.1. The counting classes	261
2.2. Definability of counting functions	263
3. Exercises	266
Part 3. Approximation and classes above NP	267
Chapter 15. Probabilistic Classes	269
1. Probabilistic decision classes	269
1.1. Error amplification	272
1.2. Comparing PP and #P	273
1.3. Other probabilistic classes	273
2. Examples of probabilistic algorithms	273
2.1. Random walk in an undirected graph	274
2.2. Perfect matching in a bipartite graph	276
2.3. Solovay-Strassen primality test	279
3. Exercises	281
Chapter 16. Probabilistic verification	283
1. Interactive proofs	284

2. Examples of interactive protocols	285
2.1. The problem of non-isomorphism of graphs	285
2.2. A protocol for the permanent	287
2.3. A protocol for QBF	289
3. Multiprover Protocols	293
4. Probabilistic checkable proofs: PCP	294
4.1. A PCP($n^3, 1$) for 3SAT	297
5. Exercises	301
Chapter 17. Approximation	303
1. Optimization problems	303
1.1. Examples of approximation algorithms	305
1.2. Approximation of optimization problems	308
1.3. Reductions and completeness	313
1.4. Non Approximability	315
2. Counting problems	318
2.1. Approximation of counting problems	318
2.2. Approximation of #DNF	319
3. Approximate verification	321
4. Exercises	324
Chapter 18. Classes above NP	327
1. The polynomial hierarchy	327
1.1. Second order definability	327
1.2. Relative computability	328
1.3. Alternating machines	330
2. Generalizations of NP	331
2.1. The classes $\oplus P$, PP and BPP	332
2.2. Universal Hashing	335
2.3. The Valiant-Vazirani theorem	337
2.4. Toda's theorem	338
3. Exponential time	342
4. Exercises	346
Bibliography	347
List of Figures	353
Index	357

Introduction

The book presents some of the fundamental ideas of Logic for Computer Science. It introduces classical notions of mathematical logic to Computer Scientists and the new ideas brought by the theory of complexity. Given a problem, it is important to know if there is an algorithmic solution, i.e. if the problem can be solved by an automatic procedure based on a finite set of instructions. The classical notions of mathematical logic, such as decidability, completeness and incompleteness are used to answer this question. When an algorithmic solution exists, it is then important to know if there exists an efficient solution, i.e. an automatic procedure which uses few resources such as time or space. Complexity theory introduces notions such as NP-completeness, reductions, approximations which are used to answer this refined question. We present these concepts from the viewpoint of logic, called *descriptive complexity* and emphasize the roles of randomness and approximation. The presentation is divided into three parts:

Part 1. Model theory and recursive functions. We introduce the basic model theory of propositional, 1st order, inductive definitions and 2nd order logic. Then, we define recursive functions, show their equivalence with Turing computable functions, prove the completeness of first-order logic and the incompleteness theorems.

Part 2. Descriptive complexity. The functions computable in polynomial time are called effectively computable and can be defined as global recursive functions on classes of finite ordered structures. Decision problems of other complexity classes such as AC, NC, LOGSPACE, NLOGSPACE, NP, PII, have similar logical characterizations and allow to view complexity questions as definability problems in logic. This approach is called descriptive as it is independent of any model of computation.

Part 3. Approximation. Randomized algorithms in polynomial time define the class BPP for decision problems which generalizes the class P and the class IP (Interactive proofs) for verification problems which generalizes the class NP. Other models of verification include PCP (probabilistic checkable proofs) and property testing. Some optimization problems and counting problems can also be approximated according to their logical form. The PCP techniques allow to prove non-approximability results.

The design of programming languages, query languages and of specification languages follows many logical principles presented in the first part of the book. The second part explains the relationship between the definitions of algorithms, queries, properties of programs and their computational complexity. For many classical tasks such as the verification of programs, optimization problems or counting problems, it is essential to first study the complexity of these problems. When the degree of unfeasibility is too high, approximation techniques presented in the third part of the book can be useful.

This book is based on two books of the same authors, in French: *Logique et Fondements de l'Informatique*, published in 1993 and *Logique et Complexité*, published in 1996, (Editions Hermès, <http://www.editions-hermes.fr>). The first part is the translation of some of the chapters of the first book, whereas the second and third parts of the present book are translations of the second book with additional new subjects. We will maintain: <http://www.lri.fr/~mdr/logicbook/> for the corrections of some of the exercises, remarks and updates.

Most of the presented material appears in the books: *Computational complexity* by C. Papadimitriou [Pap94], *Descriptive Complexity* by N. Immerman [Imm99], and *Finite Model Theory* by H. D. Ebbinghaus and J. Flum [EF91].

We wish to thank all our colleagues who helped us on the french books, in particular Stéphane Boucheron, Miklós Santha and Avy Sharell. In addition, we thank Mikael Cozic, Miki Hermann, Isabelle Bril and our translator Elena Calude for their contribution on this english version. We bear the responsibility for the errors which remain in this book.