Contents

1 Introduction and Basic Concepts **1**
1.1 Two Approaches to Formal Reasoning . . . 3
1.1.1 Proof by Deduction . . . 3
1.1.2 Proof by Enumeration . . . 4
1.1.3 Deduction and Enumeration . . . 5
1.2 Basic Definitions . . . 5
1.3 Normal Forms and Some of Their Properties . . . 8
1.4 The Theoretical Point of View . . . 14
1.4.1 The Problem We Solve . . . 17
1.4.2 Our Presentation of Theories . . . 18
1.5 Expressiveness vs. Decidability . . . 18
1.6 Boolean Structure in Decision Problems . . . 20
1.7 Logic as a Modeling Language . . . 22
1.8 Problems . . . 23
1.9 Glossary . . . 24

2 Decision Procedures for Propositional Logic **27**
2.1 Propositional Logic . . . 27
2.1.1 Motivation . . . 27
2.2 SAT Solvers . . . 29
2.2.1 The Progress of SAT Solving . . . 29
2.2.2 The CDCL Framework . . . 31
2.2.3 BCP and the Implication Graph . . . 32
2.2.4 Conflict Clauses and Resolution . . . 38
2.2.5 Decision Heuristics . . . 42
2.2.6 The Resolution Graph and the Unsatisfiable Core . . . 45
2.2.7 Incremental Satisfiability . . . 46
2.2.8 From SAT to the Constraint Satisfaction Problem . . . 48
2.2.9 SAT Solvers: Summary . . . 49
2.3 Problems . . . 50
2.3.1 Warm-up Exercises . . . 50
2.3.2 Propositional Logic . . . 51
2.3.3 Modeling . . . 51
2.3.4 Complexity . . . 52

2.3.5 CDCL SAT Solving . . . 53
2.3.6 Related Problems . . . 54
2.4 Bibliographic Notes . . . 54
2.5 Glossary . . . 58

3 From Propositional to Quantifier-Free Theories 59
3.1 Introduction . . . 59
3.2 An Overview of DPLL(T) . . . 61
3.3 Formalization . . . 64
3.4 Theory Propagation and the DPLL(T) Framework . . . 66
3.4.1 Propagating Theory Implications . . . 66
3.4.2 Performance, Performance... . . . 69
3.4.3 Returning Implied Assignments Instead of Clauses . . . 70
3.4.4 Generating Strong Lemmas . . . 71
3.4.5 Immediate Propagation . . . 72
3.5 Problems . . . 72
3.6 Bibliographic Notes . . . 73
3.7 Glossary . . . 75

4 Equalities and Uninterpreted Functions 77
4.1 Introduction . . . 77
4.1.1 Complexity and Expressiveness . . . 77
4.1.2 Boolean Variables . . . 78
4.1.3 Removing the Constants: a Simplification . . . 78
4.2 Uninterpreted Functions . . . 79
4.2.1 How Uninterpreted Functions Are Used . . . 80
4.2.2 An Example: Proving Equivalence of Programs . . . 81
4.3 Congruence Closure . . . 85
4.4 Functional Consistency Is Not Enough . . . 86
4.5 Two Examples of the Use of Uninterpreted Functions . . . 87
4.5.1 Proving Equivalence of Circuits . . . 89
4.5.2 Verifying a Compilation Process with Translation Validation . . . 91
4.6 Problems . . . 92
4.7 Bibliographic Notes . . . 93
4.8 Glossary . . . 95

5 Linear Arithmetic 97
5.1 Introduction . . . 97
5.1.1 Solvers for Linear Arithmetic . . . 98
5.2 The Simplex Algorithm . . . 99
5.2.1 A Normal Form . . . 99
5.2.2 Basics of the Simplex Algorithm . . . 100
5.2.3 Simplex with Upper and Lower Bounds . . . 102
5.2.4 Incremental Problems . . . 105

5.3 The Branch and Bound Method . . . 106
5.3.1 Cutting Planes . . . 108
5.4 Fourier–Motzkin Variable Elimination . . . 112
5.4.1 Equality Constraints . . . 112
5.4.2 Variable Elimination . . . 112
5.4.3 Complexity . . . 115
5.5 The Omega Test . . . 115
5.5.1 Problem Description . . . 115
5.5.2 Equality Constraints . . . 116
5.5.3 Inequality Constraints . . . 119
5.6 Preprocessing . . . 124
5.6.1 Preprocessing of Linear Systems . . . 124
5.6.2 Preprocessing of Integer Linear Systems . . . 125
5.7 Difference Logic . . . 126
5.7.1 Introduction . . . 126
5.7.2 A Decision Procedure for Difference Logic . . . 128
5.8 Problems . . . 129
5.8.1 Warm-up Exercises . . . 129
5.8.2 The Simplex Method . . . 129
5.8.3 Integer Linear Systems . . . 130
5.8.4 Omega Test . . . 130
5.8.5 Difference Logic . . . 131
5.9 Bibliographic Notes . . . 131
5.10 Glossary . . . 133

6 Bit Vectors 135
6.1 Bit-Vector Arithmetic . . . 135
6.1.1 Syntax . . . 135
6.1.2 Notation . . . 137
6.1.3 Semantics . . . 138
6.2 Deciding Bit-Vector Arithmetic with Flattening . . . 142
6.2.1 Converting the Skeleton . . . 142
6.2.2 Arithmetic Operators . . . 144
6.3 Incremental Bit Flattening . . . 146
6.3.1 Some Operators Are Hard . . . 146
6.3.2 Abstraction with Uninterpreted Functions . . . 148
6.4 Fixed-Point Arithmetic . . . 149
6.4.1 Semantics . . . 149
6.4.2 Flattening . . . 150
6.5 Problems . . . 151
6.5.1 Semantics . . . 151
6.5.2 Bit-Level Encodings of Bit-Vector Arithmetic . . . 152
6.5.3 Using Solvers for Linear Arithmetic . . . 152
6.6 Bibliographic Notes . . . 154
6.7 Glossary . . . 156

7 Arrays **157**
7.1 Introduction . . . 157
7.1.1 Syntax . . . 158
7.1.2 Semantics . . . 159
7.2 Eliminating the Array Terms . . . 159
7.3 A Reduction Algorithm for a Fragment of the Array Theory . 162
7.3.1 Array Properties . . . 162
7.3.2 The Reduction Algorithm . . . 163
7.4 A Lazy Encoding Procedure . . . 165
7.4.1 Incremental Encoding with DPLL(T) . . . 165
7.4.2 Lazy Instantiation of the Read-Over-Write Axiom . . . 165
7.4.3 Lazy Instantiation of the Extensionality Rule . . . 167
7.5 Problems . . . 169
7.6 Bibliographic Notes . . . 170
7.7 Glossary . . . 171

8 Pointer Logic **173**
8.1 Introduction . . . 173
8.1.1 Pointers and Their Applications . . . 173
8.1.2 Dynamic Memory Allocation . . . 174
8.1.3 Analysis of Programs with Pointers . . . 176
8.2 A Simple Pointer Logic . . . 177
8.2.1 Syntax . . . 177
8.2.2 Semantics . . . 179
8.2.3 Axiomatization of the Memory Model . . . 180
8.2.4 Adding Structure Types . . . 181
8.3 Modeling Heap-Allocated Data Structures . . . 182
8.3.1 Lists . . . 182
8.3.2 Trees . . . 183
8.4 A Decision Procedure . . . 185
8.4.1 Applying the Semantic Translation . . . 185
8.4.2 Pure Variables . . . 187
8.4.3 Partitioning the Memory . . . 188
8.5 Rule-Based Decision Procedures . . . 189
8.5.1 A Reachability Predicate for Linked Structures . . . 190
8.5.2 Deciding Reachability Predicate Formulas . . . 191
8.6 Problems . . . 194
8.6.1 Pointer Formulas . . . 194
8.6.2 Reachability Predicates . . . 195
8.7 Bibliographic Notes . . . 196
8.8 Glossary . . . 198

9 Quantified Formulas **199**
9.1 Introduction . . . 199
9.1.1 Example: Quantified Boolean Formulas . . . 201

9.1.2 Example: Quantified Disjunctive Linear Arithmetic . . . 203
9.2 Quantifier Elimination . . . 203
9.2.1 Prenex Normal Form . . . 203
9.2.2 Quantifier Elimination Algorithms . . . 205
9.2.3 Quantifier Elimination for Quantified Boolean Formulas 206
9.2.4 Quantifier Elimination for Quantified Disjunctive Linear Arithmetic . . . 209
9.3 Search-Based Algorithms for QBF . . . 210
9.4 Effectively Propositional Logic . . . 212
9.5 General Quantification . . . 215
9.6 Problems . . . 222
9.6.1 Warm-up Exercises . . . 222
9.6.2 QBF . . . 223
9.6.3 EPR . . . 224
9.6.4 General Quantification . . . 224
9.7 Bibliographic Notes . . . 225
9.8 Glossary . . . 227

10 Deciding a Combination of Theories 229
10.1 Introduction . . . 229
10.2 Preliminaries . . . 229
10.3 The Nelson–Oppen Combination Procedure . . . 231
10.3.1 Combining Convex Theories . . . 231
10.3.2 Combining Nonconvex Theories . . . 234
10.3.3 Proof of Correctness of the Nelson–Oppen Procedure . . 237
10.4 Problems . . . 240
10.5 Bibliographic Notes . . . 240
10.6 Glossary . . . 244

11 Propositional Encodings 245
11.1 Lazy vs. Eager Encodings . . . 245
11.2 From Uninterpreted Functions to Equality Logic . . . 245
11.2.1 Ackermann's Reduction . . . 246
11.2.2 Bryant's Reduction . . . 249
11.3 The Equality Graph . . . 253
11.4 Simplifications of the Formula . . . 255
11.5 A Graph-Based Reduction to Propositional Logic . . . 259
11.6 Equalities and Small-Domain Instantiations . . . 262
11.6.1 Some Simple Bounds . . . 263
11.6.2 Graph-Based Domain Allocation . . . 265
11.6.3 The Domain Allocation Algorithm . . . 266
11.6.4 A Proof of Soundness . . . 269
11.6.5 Summary . . . 271
11.7 Ackermann's vs. Bryant's Reduction: Where Does It Matter? 272
11.8 Problems . . . 273

11.8.1 Reductions . . . 274
11.8.2 Domain Allocation . . . 276
11.9 Bibliographic Notes . . . 276
11.10 Glossary . . . 279

12 Applications in Software Engineering and Computational Biology 281
12.1 Introduction . . . 281
12.2 Bounded Program Analysis . . . 283
12.2.1 Checking Feasibility of a Single Path . . . 283
12.2.2 Checking Feasibility of All Paths in a Bounded Program 287
12.3 Unbounded Program Analysis . . . 289
12.3.1 Overapproximation with Nondeterministic Assignments 289
12.3.2 The Overapproximation Can Be Too Coarse . . . 291
12.3.3 Loop Invariants . . . 294
12.3.4 Refining the Abstraction with Loop Invariants . . . 295
12.4 SMT-Based Methods in Biology . . . 297
12.4.1 DNA Computing . . . 298
12.4.2 Uncovering Gene Regulatory Networks . . . 300
12.5 Problems . . . 302
12.5.1 Warm-up Exercises . . . 302
12.5.2 Bounded Symbolic Simulation . . . 302
12.5.3 Overapproximating Programs . . . 303
12.6 Bibliographic Notes . . . 304

A SMT-LIB: a Brief Tutorial 309
A.1 The SMT-LIB Initiative . . . 309
A.2 The SMT-LIB File Interface . . . 310
A.2.1 Propositional Logic . . . 311
A.2.2 Arithmetic . . . 312
A.2.3 Bit-Vector Arithmetic . . . 313
A.2.4 Arrays . . . 313
A.2.5 Equalities and Uninterpreted Functions . . . 314

B A C++ Library for Developing Decision Procedures 315
B.1 Introduction . . . 315
B.2 Graphs and Trees . . . 316
B.2.1 Adding "Payload" . . . 318
B.3 Parsing . . . 318
B.3.1 A Grammar for First-Order Logic . . . 318
B.3.2 The Problem File Format . . . 320
B.3.3 A Class for Storing Identifiers . . . 321
B.3.4 The Parse Tree . . . 321
B.4 CNF and SAT . . . 322
B.4.1 Generating CNF . . . 322

B.4.2 Converting the Propositional Skeleton 325
B.5 A Template for a Lazy Decision Procedure 325

References **329**

Tools index **347**

Algorithms index **349**

Index **351**

http://www.springer.com/978-3-662-50496-3

Decision Procedures
An Algorithmic Point of View
Kroening, D.; Strichman, O.
2016, XXI, 356 p. 64 illus., 5 illus. in color., Hardcover
ISBN: 978-3-662-50496-3