

Contents

1	Introduction	1
1.1	A Formal Approach to Software Engineering	1
1.1.1	Test and Simulation-Based Reliability	2
1.1.2	An Alternative Approach: Formal Methods	3
1.1.3	Requirements: Functional, Security, and Safety	4
1.2	Formal Methods and Industrial Norms	5
1.3	From Classic Software Engineering to Formal Software Engineering	7
1.4	This Book	11
	References	12
2	An Overview of Formal Methods Tools and Techniques	15
2.1	The Central Problem	16
2.1.1	Some Existing Formal Methods Taxonomies	16
2.1.2	This Overview	17
2.2	Specifying and Analysing	17
2.2.1	Model-Based Specification	18
2.2.2	Algebraic Specification	24
2.2.3	Declarative Modelling	25
2.3	Specifying and Proving	26
2.3.1	Logic in a Nutshell	27
2.3.2	Proof Tools	30
2.3.3	Model Checking	31
2.3.4	Program Logics and Program Annotation	32
2.4	Specifying and Deriving	33
2.4.1	Refinement	34
2.4.2	Extraction	36
2.4.3	Execution	36
2.5	Specifying and Transforming	36
2.6	Conclusions	37
2.6.1	Are Formal Methods Tools Ready for Industry?	38
2.6.2	Is Industry Ready to Use Formal Methods?	39

2.7	To Learn More	39
	References	40
3	Propositional Logic	45
3.1	Syntax	46
3.2	Semantics	47
3.3	Proof System	54
3.4	Soundness and Completeness	61
3.5	Validity Checking: Semantic Methods	64
	3.5.1 Normal Forms in Propositional Logic	65
	3.5.2 Validity of CNF Formulas	68
	3.5.3 Satisfiability of CNF Formulas	69
3.6	Validity Checking: Deductive Methods	71
3.7	To Learn More	74
3.8	Exercises	76
	References	78
4	First-Order Logic	81
4.1	Syntax	82
4.2	Semantics	87
4.3	Proof System	95
4.4	Soundness and Completeness	100
4.5	Validity Checking	102
	4.5.1 Negation and Prenex Normal Forms	102
	4.5.2 Herbrand/Skolem Normal Forms and Semi-Decidability	104
	4.5.3 Decidable Fragments	108
4.6	Variations and Extensions	109
	4.6.1 First-Order Logic with Equality	109
	4.6.2 Many-Sorted First-Order Logic	110
	4.6.3 Second-Order Logic	113
4.7	First-Order Theories	115
	4.7.1 Equality	117
	4.7.2 Natural Numbers	118
	4.7.3 Integers	119
	4.7.4 Arrays	120
	4.7.5 Other Theories	121
	4.7.6 Combining Theories	121
4.8	To Learn More	122
4.9	Exercises	124
	References	127
5	Hoare Logic	129
5.1	Annotated <i>While</i> Programs	130
	5.1.1 Program Semantics	131
	5.1.2 The $\text{While}^{\text{int}}$ Programming Language	134
5.2	Specifications and Hoare Triples	136

- 5.3 Loop Invariants 140
- 5.4 Hoare Calculus 143
- 5.5 The $\text{While}^{\text{array}}$ Programming Language 148
 - 5.5.1 A Rule of Hoare Logic for Array Assignment 150
- 5.6 Loop Termination and Total Correctness 151
- 5.7 Adaptation 151
- 5.8 To Learn More 154
- 5.9 Exercises 155
 - References 157

- 6 Generating Verification Conditions 159**
 - 6.1 Mechanising Hoare Logic 159
 - 6.2 The Weakest Precondition Strategy 162
 - 6.3 An Architecture for Program Verification 167
 - 6.4 A VCGen Algorithm 168
 - 6.4.1 Calculating the Weakest Precondition 168
 - 6.4.2 Calculating Verification Conditions 170
 - 6.4.3 Putting It All Together 170
 - 6.5 Verification Conditions for $\text{While}^{\text{array}}$ Programs 172
 - 6.6 To Learn More 177
 - 6.7 Exercises 178
 - References 179

- 7 Safety Properties 181**
 - 7.1 Error Semantics and Safe Programs 181
 - 7.1.1 $\text{While}^{\text{int}}$ with Errors 184
 - 7.2 Safety-Sensitive Calculus and VCGen 185
 - 7.2.1 Safe $\text{While}^{\text{int}}$ Programs 187
 - 7.3 Bounded Arrays: The $\text{While}^{\text{array}[N]}$ Language 188
 - 7.3.1 Safe $\text{While}^{\text{array}[N]}$ Programs 190
 - 7.3.2 An Alternative Formalisation of Bounded Arrays 192
 - 7.4 To Learn More 193
 - 7.5 Exercises 193
 - References 194

- 8 Procedures and Contracts 195**
 - 8.1 Procedures and Recursion 196
 - 8.1.1 The \sim Notation 197
 - 8.1.2 Recursive Procedures 198
 - 8.1.3 Procedure Calls in System \mathcal{H}_g 199
 - 8.2 Contracts and Mutual Recursion 200
 - 8.2.1 Programming with Contracts 202
 - 8.2.2 Inference System for Parameterless Procedures 203
 - 8.2.3 Verification Conditions for Parameterless Procedures 204
 - 8.3 Frame Conditions 206
 - 8.4 Procedures with Parameters 208

8.4.1	Parameters Passed by Value	211
8.4.2	Parameters Passed by Reference	215
8.4.3	Aliasing	218
8.5	Return Values and Pure Functions	220
8.6	To Learn More	226
8.7	Exercises	226
	References	227
9	Specifying C Programs	229
9.1	An Introduction to ACSL	230
9.1.1	Array-Based Programs	230
9.1.2	Using Axiomatics	231
9.1.3	Function Calls	233
9.1.4	State Labels and Behaviours	234
9.2	To Learn More	238
9.3	Exercises	239
	References	239
10	Verifying C Programs	241
10.1	Safety Verification	242
10.1.1	Arithmetic Overflow Safety	243
10.1.2	Safety of Array Access	244
10.1.3	Adding Loop Invariants	245
10.1.4	Termination Checking and Loop Variants	246
10.1.5	Safety of Function Calls	247
10.2	Functional Correctness: Array Partitioning	248
10.3	Functional Correctness: Multiset Preservation	250
10.4	A Word of Caution	251
10.5	Pointer Variables and Parameters Passed by Reference	253
10.6	To Learn More	254
10.7	Exercises	255
	References	255
	Index	258



<http://www.springer.com/978-0-85729-017-5>

Rigorous Software Development
An Introduction to Program Verification
Almeida, J.B.; Frade, M.J.; Pinto, J.S.; Melo de Sousa, S.
2011, XIII, 307 p. 52 illus., Softcover
ISBN: 978-0-85729-017-5