Contents

1 Introduction 1
- 1.1 Objective 1
- 1.2 Programs and Services 2
- 1.3 Correctness Properties and Assertional Reasoning 5
- 1.4 Services and Systems Framework (SESF) 9
- 1.5 Programs in SESF 10
  - 1.5.1 Input Assumptions 12
  - 1.5.2 Platforms 12
  - 1.5.3 Atomicity Assumptions About the Platform and Effective Atomicity 13
  - 1.5.4 Progress Assumptions About the Platform 14
  - 1.5.5 Aggregate Systems 15
  - 1.5.6 Composite Systems 15
  - 1.5.7 Producer-Consumer-Lock Example 16
- 1.6 Service Programs 18
  - 1.6.1 Distributed Services 21
- 1.7 Implements Definition 21
- 1.8 Using Services to Design Programs 27
- 1.9 Background to SESF 29
- 1.10 Organization of This Book 30
  - 1.10.1 Introducing SESF by Examples (Chaps. 2–5) 31
  - 1.10.2 Theory of SESF (Chaps. 6–8) 32
  - 1.10.3 More Applications of SESF (Chap. 9 Onwards) 32

References 34

2 Simple Lock 41
- 2.1 Introduction 41
  - 2.1.1 Conventions 41
- 2.2 Lock Program 42
  - 2.2.1 Fault-Freedom and Effective Atomicity 44
- 2.3 Lock Service Program 45

2.4 Implements Conditions 47
2.5 Proving the Implements Conditions 48
2.5.1 Proving the Safety Condition: B_0 52
2.5.2 Proving the Progress Condition: $B_1 - B_3$ 53
2.5.3 Assertional Proof of *Inv* C_1 54
2.5.4 Assertional Proof of D_2 55
2.6 Producer and Consumer Using Lock Service 56
2.7 Concluding Remarks 59
Exercises 60
References 61

3 Bounded Buffer 63
3.1 Introduction 63
3.1.1 Conventions 64
3.2 Bounded-Buffer Service 64
3.2.1 Bounded-Buffer Service Inverse 66
3.3 Awaits 66
3.3.1 Bounded-Buffer Implementation Using Awaits 68
3.4 Locks and Condition Variables 70
3.4.1 Bounded-Buffer Implementation Using Locks and Condition Variables 72
3.5 Semaphores 72
3.5.1 Bounded-Buffer Implementation Using Semaphores 74
3.6 Increasing Parallelism 76
3.6.1 Technique 1 76
3.6.2 Technique 2 78
3.7 Bounded-Buffer Service with Cancelable Put and Get 81
3.8 Concluding Remarks 82
Exercises 85
References 87

4 Message-Passing Services 89
4.1 Introduction 89
4.1.1 Conventions 90
4.2 Connection-less Fifo Channels 92
4.2.1 Weaker Progress Assumption 94
4.2.2 Termination and RX Cancellation 94
4.2.3 Fifo Channel Inverse 95
4.3 Connection-less Lossy Channels 97
4.4 Connection-less LRD Channels 97
4.5 Connection-oriented Fifo Channel 100
4.6 Multiplexing Ports onto Channels 104
4.7 Concluding Remarks 106
4.7.1 Conventions 106
References 109

5 Fifo Channels from Unreliable Channels 111
5.1 Introduction 111
5.1.1 Conventions 112
5.2 Distributed Sliding Window Program: Algorithm Level 112
5.2.1 Source 114
5.2.2 Sink 115
5.2.3 Effective Atomicity of `SwpDist` 117
5.3 Analysis of the Sliding Window Algorithm 117
5.3.1 Correct Interpretation Conditions 118
5.3.2 Achieving Correct Interpretation over Lossy Channel ... 119
5.3.3 Achieving Correct Interpretation over LRD Channel 120
5.3.4 Achieving Progress 121
5.4 Distributed Sliding Window Program 122
5.4.1 Source and Sink Programs with Awaits 122
5.4.2 Source and Sink Programs with Locks and Condition Variables 124
5.4.3 Increasing Parallelism 127
5.5 Data Transfer Protocol 129
5.6 Proving That `DtpDist` Implements `FifoChannel` 131
5.7 Graceful-Closing Data Transfer Protocol 132
5.8 Abortable Data Transfer Protocol 134
5.9 Concluding Remarks 134
Exercises 135
References 138

6 Programs, Semantics and Effective Atomicity 141
6.1 Introduction 141
6.2 Programs 142
6.2.1 Inputs and Outputs 144
6.2.2 Input Assumptions 144
6.3 Service Programs 145
6.3.1 Input Functions 145
6.3.2 Output Functions 146
6.3.3 Atomicity and Progress Assumptions 147
6.3.4 Usability and Fault-Freedom 148
6.4 State Transition Semantics of Systems 149
6.4.1 States 149
6.4.2 Transitions 149
6.4.3 Evolutions 150
6.5 Predicates and their Evaluation 151
6.6 Assertions and Their Evaluation 152
6.7 Splitting and Stitching of Evolutions 154
6.8 Auxiliary Variables 156

6.9 Effective Atomicity 157
6.9.1 Defining Effective Atomicity 157
6.9.2 Commuting Statements 158
6.9.3 Some Simple Sufficient Conditions 160
6.10 Proof Rules 161
6.10.1 Hoare-Triples 161
6.10.2 Proof Rules for Safety Assertions 163
6.10.3 Proof Rules for Progress Assertions 164
6.11 Concluding Remarks 165
References 167

7 Implements and Compositionality 173
7.1 Introduction 173
7.1.1 Conventions 173
7.2 Implements 174
7.3 Compositionality 175
7.4 Program Version of Implements for Services without Internal Parameters 176
7.5 Program Version of Implements for Services with Internal Parameters 179
7.6 Proof of Theorem 7.1 180
7.6.1 Proof of Safety Parts of Theorem 7.1 182
7.6.2 Proof of Progress Parts of Theorem 7.1 183
7.7 Proof of Theorem 7.2 183
7.8 Proof of Theorems 7.3 and 7.4 185
7.9 Concluding Remarks 187
Exercises 188
References 188

8 SESF for Time-Constrained Programs 191
8.1 Introduction 191
8.2 Time-Constrained Programs 192
8.2.1 Modeling Time Constraints 193
8.2.2 Modeling Timing Assumptions 194
8.3 Time-Constrained Service Programs 195
8.4 Implements and Compositionality 198
8.4.1 Program Version of Implements 199
8.5 Timing Assumptions for Implementation Programs 200
8.5.1 Transforming Timing Constructs into Blocking Conditions 201
8.6 Time-Constrained Implementation Program 203
8.7 Concluding Remarks 203
Exercises 203
References 205

9 Lock Using Peterson's Algorithm 207
9.1 Introduction 207

9.2 Lock Program and Implements Conditions 208
9.3 Proving the Implements Conditions 210
9.3.1 Proving the Safety Condition: Y_1 210
9.3.2 Proving the Progress Condition: Y_2–Y_3 211
9.4 Concluding Remarks 212
Exercises 212
References 212

10 Lock Using the Bakery Algorithm 213
10.1 Introduction 213
10.2 Simplified Bakery Lock Program and Implements Conditions 214
10.3 Proving the Implements Conditions for Simplified Bakery 216
10.3.1 Proving the Safety Condition: Y_1 217
10.3.2 Proving the Progress Condition: Y_2–Y_3 218
10.4 Original Bakery Lock Program and Implements Conditions 218
10.5 Proving the Implements Condition for Original Bakery 219
10.5.1 Proving the Safety Condition: Y_1 221
10.5.2 Proving the Progress Condition: Y_2–Y_3 222
10.6 Concluding Remarks 222
Exercises 224
References 224

11 Distributed Lock Service 225
11.1 Introduction 225
11.2 Service Program 225
11.2.1 Service Inverse 227
11.3 Concluding Remarks 227
Exercises 229
References 229

12 Distributed Lock Using Timestamps 231
12.1 Introduction 231
12.2 Request Scheduling Problem and Solution Using Timestamps 232
12.3 Distributed Lock Program: Algorithm Level 234
12.3.1 Analysis 236
12.4 Distributed Lock Program 238
12.4.1 Proving the Implements Conditions 240
12.5 Using Cyclic Timestamps 242
12.5.1 Analysis 242
12.6 Concluding Remarks 245
Exercises 246
References 247

13 Channel with Termination Detection Service 249
13.1 Introduction 249
13.2 Service Program 249
13.3 Concluding Remarks 250

Exercises 253
References 253

14 Termination Detection for Diffusing Computations 255
14.1 Introduction 255
14.1.1 Conventions 256
14.2 Distributed Program: Algorithm Level 256
14.2.1 Analysis 259
14.3 Distributed Program 260
14.4 Proving the Implements Conditions 263
14.4.1 Proof of C_1–C_2 264
14.4.2 Proof of C_3–C_6 265
14.5 Concluding Remarks 266
Exercises 266
References 267

15 Object-Transfer Service 269
15.1 Introduction 269
15.2 Service Program 270
15.3 Concluding Remarks 273
Exercises 273
References 274

16 Object Transfer Using Path Reversal 277
16.1 Introduction 277
16.1.1 Conventions 278
16.2 Distributed Path-Reversal Program 278
16.2.1 Serial Evolutions with at Most One Hungry System at Any Time 280
16.2.2 Serial Evolutions with Multiple Hungry Systems at a Time 281
16.2.3 Non-serial Evolutions 282
16.3 Algorithm Analysis: Safety Properties 282
16.3.1 Digraph `LR` 283
16.3.2 Digraph `Pr` 284
16.4 Algorithm Analysis: Monotonic Progress Metric 286
16.5 Algorithm Analysis: Serializability 287
16.6 Distributed Program Implementing Object-Transfer Service 289
16.7 Proving the Implements Conditions 293
16.7.1 Assertions from Path Reversal Analysis 294
16.7.2 Assertions Concerning `objBuff` and `reqBuff` 295
16.7.3 Proof of *Inv* F_1–F_6 295
16.7.4 Proof of E_3–E_5 296
16.8 Concluding Remarks 296
Exercises 297
References 298

17 Distributed Shared Memory Service 299
17.1 Introduction 299
17.2 Service Program 300
17.3 Concluding Remarks 300
Exercises 302
References 303

18 A Single-Copy Distributed Shared Memory 305
18.1 Introduction 305
18.2 Distributed Program 305
18.3 Proving the Implements Conditions 307
18.3.1 Proving Safety: A_1 309
18.3.2 Proving Progress: A_2–A_3 311
18.4 Concluding Remarks 311
Exercises 312
References 312

19 A Multi-copy Distributed Shared Memory 313
19.1 Introduction 313
19.2 Distributed Program 314
19.3 Concluding Remarks 316
Exercises 316
References 317

20 Reliable Transport Service 321
20.1 Introduction 321
20.2 Service Overview 322
20.2.1 Function `j.connect(k)` 324
20.2.2 Function `j.accept()` 325
20.2.3 Functions `j.tx(k,msg)` and `j.rx(k)` 326
20.2.4 Function `j.close(k)` 326
20.3 Service Program 326
20.4 Concluding Remarks 328
Exercises 329
References 330

21 Reliable Transport Protocol 341
21.1 Introduction 341
21.2 Graceful-Closing dtp Program 343
21.3 Protocol Overview 344
21.3.1 Handling Message Losses 347
21.3.2 Handling Old Messages 348
21.3.3 Overlapping Handshakes 349
21.3.4 Buffering Incoming `CCR` Messages in Server Mode 349
21.4 Program `Tp` with Unbounded Incarnation Numbers 349
21.5 Establishing the Implements Condition 352
21.5.1 Assertions to Be Proved 353

21.5.2 Proof of B_1–B_5 and C_1–C_5 354
21.5.3 Proof of B_6–B_8 and C_6–C_9 356
21.6 Using Cyclic Incarnation Numbers 357
21.6.1 Program `Tp` with Modulo-`N` Endpoint Numbers 360
21.7 Conclusion 361
Exercises 362
References 362

A Conventions 371
A.1 Predicates and Assertions 371
A.2 Referencing System Quantities 372
A.3 Types 372
A.4 Sids 372
A.5 Tids 373
A.6 Sets 373
A.7 Bags 374
A.8 Sequences 374
A.9 Maps 376
A.10 Channels 376
A.11 Graphs 377
A.12 Miscellaneous 378

Index 379

http://www.springer.com/978-1-4614-4881-5

Distributed Programming
Theory and Practice
Shankar, A.U.
2013, XVIII, 386 p.,
ISBN: 978-1-4614-4881-5