

Contents

Preface	xi
1 What is Concurrent Programming?	1
1.1 Introduction	1
1.2 Concurrency as abstract parallelism	2
1.3 Multitasking	4
1.4 The terminology of concurrency	4
1.5 Multiple computers	5
1.6 The challenge of concurrent programming	5
2 The Concurrent Programming Abstraction	7
2.1 The role of abstraction	7
2.2 Concurrent execution as interleaving of atomic statements	8
2.3 Justification of the abstraction	13
2.4 Arbitrary interleaving	17
2.5 Atomic statements	19
2.6 Correctness	21
2.7 Fairness	23
2.8 Machine-code instructions	24
2.9 Volatile and non-atomic variables	28
2.10 The BACI concurrency simulator	29
2.11 Concurrency in Ada	31

2.12	Concurrency in Java	34
2.13	Writing concurrent programs in Promela	36
2.14	Supplement: the state diagram for the frog puzzle	37
3	The Critical Section Problem	45
3.1	Introduction	45
3.2	The definition of the problem	45
3.3	First attempt	48
3.4	Proving correctness with state diagrams	49
3.5	Correctness of the first attempt	53
3.6	Second attempt	55
3.7	Third attempt	57
3.8	Fourth attempt	58
3.9	Dekker's algorithm	60
3.10	Complex atomic statements	61
4	Verification of Concurrent Programs	67
4.1	Logical specification of correctness properties	68
4.2	Inductive proofs of invariants	69
4.3	Basic concepts of temporal logic	72
4.4	Advanced concepts of temporal logic	75
4.5	A deductive proof of Dekker's algorithm	79
4.6	Model checking	83
4.7	Spin and the Promela modeling language	83
4.8	Correctness specifications in Spin	86
4.9	Choosing a verification technique	88
5	Advanced Algorithms for the Critical Section Problem	93
5.1	The bakery algorithm	93
5.2	The bakery algorithm for N processes	95
5.3	Less restrictive models of concurrency	96

5.4	Fast algorithms	97
5.5	Implementations in Promela	104
6	Semaphores	107
6.1	Process states	107
6.2	Definition of the semaphore type	109
6.3	The critical section problem for two processes	110
6.4	Semaphore invariants	112
6.5	The critical section problem for N processes	113
6.6	Order of execution problems	114
6.7	The producer–consumer problem	115
6.8	Definitions of semaphores	119
6.9	The problem of the dining philosophers	122
6.10	Barz’s simulation of general semaphores	126
6.11	Udding’s starvation-free algorithm	129
6.12	Semaphores in BACI	131
6.13	Semaphores in Ada	132
6.14	Semaphores in Java	133
6.15	Semaphores in Promela	134
7	Monitors	145
7.1	Introduction	145
7.2	Declaring and using monitors	146
7.3	Condition variables	147
7.4	The producer–consumer problem	151
7.5	The immediate resumption requirement	152
7.6	The problem of the readers and writers	154
7.7	Correctness of the readers and writers algorithm	157
7.8	A monitor solution for the dining philosophers	160
7.9	Monitors in BACI	162

7.10	Protected objects	162
7.11	Monitors in Java	167
7.12	Simulating monitors in Promela	173
8	Channels	179
8.1	Models for communications	179
8.2	Channels	181
8.3	Parallel matrix multiplication	183
8.4	The dining philosophers with channels	187
8.5	Channels in Promela	188
8.6	Rendezvous	190
8.7	Remote procedure calls	193
9	Spaces	197
9.1	The Linda model	197
9.2	Expressiveness of the Linda model	199
9.3	Formal parameters	200
9.4	The master–worker paradigm	202
9.5	Implementations of spaces	204
10	Distributed Algorithms	211
10.1	The distributed systems model	211
10.2	Implementations	215
10.3	Distributed mutual exclusion	216
10.4	Correctness of the Ricart–Agrawala algorithm	223
10.5	The RA algorithm in Promela	225
10.6	Token-passing algorithms	227
10.7	Tokens in virtual trees	230
11	Global Properties	237
11.1	Distributed termination	237

11.2	The Dijkstra–Scholten algorithm	243
11.3	Credit-recovery algorithms	248
11.4	Snapshots	250
12	Consensus	257
12.1	Introduction	257
12.2	The problem statement	258
12.3	A one-round algorithm	260
12.4	The Byzantine Generals algorithm	261
12.5	Crash failures	263
12.6	Knowledge trees	264
12.7	Byzantine failures with three generals	266
12.8	Byzantine failures with four generals	268
12.9	The flooding algorithm	271
12.10	The King algorithm	274
12.11	Impossibility with three generals	280
13	Real-Time Systems	285
13.1	Introduction	285
13.2	Definitions	287
13.3	Reliability and repeatability	288
13.4	Synchronous systems	290
13.5	Asynchronous systems	293
13.6	Interrupt-driven systems	297
13.7	Priority inversion and priority inheritance	299
13.8	The Mars Pathfinder in Spin	303
13.9	Simpson’s four-slot algorithm	306
13.10	The Ravenscar profile	309
13.11	UPPAAL	311
13.12	Scheduling algorithms for real-time systems	312

A	The Pseudocode Notation	317
B	Review of Mathematical Logic	321
B.1	The propositional calculus	321
B.2	Induction	323
B.3	Proof methods	324
B.4	Correctness of sequential programs	326
C	Concurrent Programming Problems	331
D	Software Tools	339
D.1	BACI and jBACI	339
D.2	Spin and jSpin	341
D.3	DAJ	345
E	Further Reading	349
	Bibliography	351
	Index	355

Supporting Resources

Visit www.pearsoned.co.uk/ben-ari to find valuable online resources

Companion Website for students

- Source code for all the algorithms in the book
- Links to sites where software for studying concurrency may be downloaded.

For instructors

- PDF slides of all diagrams, algorithms and scenarios (with \LaTeX source)
- Answers to exercises

For more information please contact your local Pearson Education sales representative or visit www.pearsoned.co.uk/ben-ari