

Contents

Part I Lock-Based Synchronization

1	The Mutual Exclusion Problem.	3
1.1	Multiprocess Program	3
1.1.1	The Concept of a Sequential Process.	3
1.1.2	The Concept of a Multiprocess Program	4
1.2	Process Synchronization	4
1.2.1	Processors and Processes	4
1.2.2	Synchronization	4
1.2.3	Synchronization: Competition.	5
1.2.4	Synchronization: Cooperation.	7
1.2.5	The Aim of Synchronization Is to Preserve Invariants	7
1.3	The Mutual Exclusion Problem	9
1.3.1	The Mutual Exclusion Problem (Mutex)	9
1.3.2	Lock Object.	11
1.3.3	Three Families of Solutions	12
1.4	Summary	13
1.5	Bibliographic Notes	13
2	Solving Mutual Exclusion	15
2.1	Mutex Based on Atomic Read/Write Registers.	15
2.1.1	Atomic Register	15
2.1.2	Mutex for Two Processes: An Incremental Construction	17
2.1.3	A Two-Process Algorithm	19
2.1.4	Mutex for n Processes: Generalizing the Previous Two-Process Algorithm.	22
2.1.5	Mutex for n Processes: A Tournament-Based Algorithm.	26
2.1.6	A Concurrency-Abortable Algorithm.	29

2.1.7	A Fast Mutex Algorithm	33
2.1.8	Mutual Exclusion in a Synchronous System.	37
2.2	Mutex Based on Specialized Hardware Primitives	38
2.2.1	Test&Set, Swap and Compare&Swap	39
2.2.2	From Deadlock-Freedom to Starvation-Freedom.	40
2.2.3	Fetch&Add	44
2.3	Mutex Without Atomicity	45
2.3.1	Safe, Regular and Atomic Registers	45
2.3.2	The Bakery Mutex Algorithm	48
2.3.3	A Bounded Mutex Algorithm.	53
2.4	Summary	58
2.5	Bibliographic Notes	58
2.6	Exercises and Problems	59
3	Lock-Based Concurrent Objects	61
3.1	Concurrent Objects	61
3.1.1	Concurrent Object.	61
3.1.2	Lock-Based Implementation.	62
3.2	A Base Synchronization Object: the Semaphore	63
3.2.1	The Concept of a Semaphore	63
3.2.2	Using Semaphores to Solve the Producer-Consumer Problem.	65
3.2.3	Using Semaphores to Solve a Priority Scheduling Problem	71
3.2.4	Using Semaphores to Solve the Readers-Writers Problem	74
3.2.5	Using a Buffer to Reduce Delays for Readers and Writers.	78
3.3	A Construct for Imperative Languages: the Monitor	81
3.3.1	The Concept of a Monitor	82
3.3.2	A Rendezvous Monitor Object	83
3.3.3	Monitors and Predicates.	85
3.3.4	Implementing a Monitor from Semaphores	87
3.3.5	Monitors for the Readers-Writers Problem.	89
3.3.6	Scheduled Wait Operation	94
3.4	Declarative Synchronization: Path Expressions.	95
3.4.1	Definition	96
3.4.2	Using Path Expressions to Solve Synchronization Problems	97
3.4.3	A Semaphore-Based Implementation of Path Expressions.	98
3.5	Summary	101
3.6	Bibliographic Notes	102
3.7	Exercises and Problems	102

Part II On the Foundations Side: The Atomicity Concept

4	Atomicity: Formal Definition and Properties	113
4.1	Introduction	113
4.2	Computation Model	115
4.2.1	Processes and Operations	115
4.2.2	Objects	116
4.2.3	Histories	117
4.2.4	Sequential History	119
4.3	Atomicity	120
4.3.1	Legal History	120
4.3.2	The Case of Complete Histories	121
4.3.3	The Case of Partial Histories	123
4.4	Object Composability and Guaranteed Termination Property	125
4.4.1	Atomic Objects Compose for Free	125
4.4.2	Guaranteed Termination	127
4.5	Alternatives to Atomicity	128
4.5.1	Sequential Consistency	128
4.5.2	Serializability	130
4.6	Summary	131
4.7	Bibliographic Notes	132

Part III Mutex-Free Synchronization

5	Mutex-Free Concurrent Objects	135
5.1	Mutex-Freedom and Progress Conditions	135
5.1.1	The Mutex-Freedom Notion	135
5.1.2	Progress Conditions	137
5.1.3	Non-blocking with Respect to Wait-Freedom	140
5.2	Mutex-Free Concurrent Objects	140
5.2.1	The Splitter: A Simple Wait-Free Object from Read/Write Registers	140
5.2.2	A Simple Obstruction-Free Object from Read/Write Registers	143
5.2.3	A Remark on Compare&Swap: The ABA Problem	145
5.2.4	A Non-blocking Queue Based on Read/Write Registers and Compare&Swap	146
5.2.5	A Non-blocking Stack Based on Compare&Swap Registers	150
5.2.6	A Wait-Free Stack Based on Fetch&Add and Swap Registers	152

5.3	Boosting Obstruction-Freedom to Stronger Progress in the Read/Write Model	155
5.3.1	Failure Detectors	155
5.3.2	Contention Managers for Obstruction-Free Object Implementations	157
5.3.3	Boosting Obstruction-Freedom to Non-blocking	158
5.3.4	Boosting Obstruction-Freedom to Wait-Freedom	159
5.3.5	Mutex-Freedom Versus Loops Inside a Contention Manager Operation	161
5.4	Summary	162
5.5	Bibliographic Notes	162
5.6	Exercises and Problems	163
6	Hybrid Concurrent Objects	165
6.1	The Notion of a Hybrid Implementation	165
6.1.1	Lock-Based Versus Mutex-Free Operation: Static Hybrid Implementation.	166
6.1.2	Contention Sensitive (or Dynamic Hybrid) Implementation.	166
6.1.3	The Case of Process Crashes	166
6.2	A Static Hybrid Implementation of a Concurrent Set Object	167
6.2.1	Definition and Assumptions	167
6.2.2	Internal Representation and Operation Implementation.	167
6.2.3	Properties of the Implementation	171
6.3	Contention-Sensitive Implementations	172
6.3.1	Contention-Sensitive Binary Consensus	172
6.3.2	A Contention Sensitive Non-blocking Double-Ended Queue	176
6.4	The Notion of an Abortable Object.	181
6.4.1	Concurrency-Abortable Object	181
6.4.2	From a Non-blocking Abortable Object to a Starvation-Free Object	183
6.5	Summary	186
6.6	Bibliographic Notes	186
6.7	Exercises and Problems	187
7	Wait-Free Objects from Read/Write Registers Only	189
7.1	A Wait-Free Weak Counter for Infinitely Many Processes.	189
7.1.1	A Simple Counter Object.	190
7.1.2	Weak Counter Object for Infinitely Many Processes.	191
7.1.3	A One-Shot Weak Counter Wait-Free Algorithm	193
7.1.4	Proof of the One-Shot Implementation	194
7.1.5	A Multi-Shot Weak Counter Wait-Free Algorithm	199

7.2	Store-Collect Object	201
7.2.1	Store-Collect Object: Definition	201
7.2.2	An Adaptive Store-Collect Implementation	204
7.2.3	Proof and Cost of the Adaptive Implementation	208
7.3	Fast Store-Collect Object	211
7.3.1	Fast Store-Collect Object: Definition.	211
7.3.2	A Fast Algorithm for the <code>store_collect()</code> Operation.	212
7.3.3	Proof of the Fast Store-Collect Algorithm	215
7.4	Summary	217
7.5	Bibliographic Notes	217
7.6	Problem.	218
8	Snapshot Objects from Read/Write Registers Only	219
8.1	Snapshot Objects: Definition	219
8.2	Single-Writer Snapshot Object	220
8.2.1	An Obstruction-Free Implementation.	221
8.2.2	From Obstruction-Freedom to Bounded Wait-Freedom	223
8.2.3	One-Shot Single-Writer Snapshot Object: Containment Property	227
8.3	Single-Writer Snapshot Object with Infinitely Many Processes	228
8.4	Multi-Writer Snapshot Object.	230
8.4.1	The Strong Freshness Property	231
8.4.2	An Implementation of a Multi-Writer Snapshot Object	231
8.4.3	Proof of the Implementation.	234
8.5	Immediate Snapshot Objects	238
8.5.1	One-Shot Immediate Snapshot Object: Definition	238
8.5.2	One-Shot Immediate Snapshot Versus One-Shot Snapshot	238
8.5.3	An Implementation of One-Shot Immediate Snapshot Objects	240
8.5.4	A Recursive Implementation of a One-Shot Immediate Snapshot Object	244
8.6	Summary	247
8.7	Bibliographic Notes	247
8.8	Problem.	248

9	Renaming Objects from Read/Write Registers Only	249
9.1	Renaming Objects.	249
9.1.1	The Base Renaming Problem	249
9.1.2	One-Shot Renaming Object	250
9.1.3	Adaptive Implementations	250
9.1.4	A Fundamental Result	251
9.1.5	Long-Lived Renaming.	252
9.2	Non-triviality of the Renaming Problem	252
9.3	A Splitter-Based Optimal Time-Adaptive Implementation	254
9.4	A Snapshot-Based Optimal Size-Adaptive Implementation.	256
9.4.1	A Snapshot-Based Implementation	256
9.4.2	Proof of the Implementation.	258
9.5	Recursive Store-Collect-Based Size-Adaptive Implementation.	259
9.5.1	A Recursive Renaming Algorithm	259
9.5.2	An Example.	262
9.5.3	Proof of the Renaming Implementation	263
9.6	Variant of the Previous Recursion-Based Renaming Algorithm.	266
9.6.1	A Renaming Implementation Based on Immediate Snapshot Objects	266
9.6.2	An Example of a Renaming Execution	268
9.7	Long-Lived Perfect Renaming Based on Test&Set Registers.	269
9.7.1	Perfect Adaptive Renaming	269
9.7.2	Perfect Long-Lived Test&Set-Based Renaming	270
9.8	Summary	271
9.9	Bibliographic Notes	271
9.10	Exercises and Problems	272

Part IV The Transactional Memory Approach

10	Transactional Memory	277
10.1	What Are Software Transactional Memories	277
10.1.1	Transactions = High-Level Synchronization	277
10.1.2	At the Programming Level.	279
10.2	STM System	281
10.2.1	Speculative Executions, Commit and Abort of a Transaction	281
10.2.2	An STM Consistency Condition: Opacity	282
10.2.3	An STM Interface.	282
10.2.4	Incremental Reads and Deferred Updates.	283

	10.2.5	Read-Only Versus Update Transactions	283
	10.2.6	Read Invisibility	284
10.3		A Logical Clock-Based STM System: TL2	284
	10.3.1	Underlying System and Control Variables of the STM System	284
	10.3.2	Underlying Principle: Consistency with Respect to Transaction Birth Date	285
	10.3.3	The Implementation of an Update Transaction	286
	10.3.4	The Implementation of a Read-Only Transaction	288
10.4		A Version-Based STM System: JVSTM	289
	10.4.1	Underlying and Control Variables of the STM System	290
	10.4.2	The Implementation of an Update Transaction	291
	10.4.3	The Implementation of a Read-Only Transaction	293
10.5		A Vector Clock-Based STM System	293
	10.5.1	The Virtual World Consistency Condition	293
	10.5.2	An STM System for Virtual World Consistency	295
	10.5.3	The Algorithms Implementing the STM Operations	296
10.6		Summary	299
10.7		Bibliographic Notes	299
10.8		Exercises and Problems	300

Part V On the Foundations Side: From Safe Bits to Atomic Registers

11		Safe, Regular, and Atomic Read/Write Registers	305
11.1		Safe, Regular, and Atomic Registers	305
	11.1.1	Reminder: The Many Faces of a Register	305
	11.1.2	From Regularity to Atomicity: A Theorem	308
	11.1.3	A Fundamental Problem: The Construction of Registers	310
11.2		Two Very Simple Bounded Constructions	311
	11.2.1	Safe/Regular Registers: From Single-Reader to Multi-Reader	311
	11.2.2	Binary Multi-Reader Registers: From Safe to Regular	313
11.3		From Bits to b -Valued Registers	314
	11.3.1	From Safe Bits to b -Valued Safe Registers	314
	11.3.2	From Regular Bits to Regular b -Valued Registers	315
	11.3.3	From Atomic Bits to Atomic b -Valued Registers	319

11.4	Three Unbounded Constructions	321
11.4.1	SWSR Registers: From Unbounded Regular to Atomic.	322
11.4.2	Atomic Registers: From Unbounded SWSR to SWMR	324
11.4.3	Atomic Registers: From Unbounded SWMR to MWMR	325
11.5	Summary	327
11.6	Bibliographic Notes	327
12	From Safe Bits to Atomic Bits:	
	Lower Bound and Optimal Construction	329
12.1	A Lower Bound Theorem	329
12.1.1	Two Preliminary Lemmas	330
12.1.2	The Lower Bound Theorem	331
12.2	A Construction of an Atomic Bit from Three Safe Bits	334
12.2.1	Base Architecture of the Construction	334
12.2.2	Underlying Principle and Signaling Scheme	335
12.2.3	The Algorithm Implementing the Operation $R.write()$	336
12.2.4	The Algorithm Implementing the Operation $R.read()$	336
12.2.5	Cost of the Construction	338
12.3	Proof of the Construction of an Atomic Bit	338
12.3.1	A Preliminary Theorem	338
12.3.2	Proof of the Construction	340
12.4	Summary	344
12.5	Bibliographic Notes	345
12.6	Exercise	345
13	Bounded Constructions of Atomic b-Valued Registers	347
13.1	Introduction	347
13.2	A Collision-Free (Pure Buffers) Construction	349
13.2.1	Internal Representation of the Atomic b -Valued Register R	349
13.2.2	Underlying Principle: Two-Level Switch to Ensure Collision-Free Accesses to Buffers	349
13.2.3	The Algorithms Implementing the Operations $R.write()$ and $R.read()$	350
13.2.4	Proof of the Construction: Collision-Freedom	352
13.2.5	Correctness Proof	355
13.3	A Construction Based on Impure Buffers	357
13.3.1	Internal Representation of the Atomic b -Valued Register R	357

13.3.2	An Incremental Construction	358
13.3.3	The Algorithms Implementing the Operations $R.write()$ and $R.read()$	360
13.3.4	Proof of the Construction.	360
13.3.5	From SWSR to SWMR b -Valued Atomic Register	367
13.4	Summary	368
13.5	Bibliographic Notes	368

Part VI On the Foundations Side:

The Computability Power of Concurrent Objects (Consensus)

14	Universality of Consensus	371
14.1	Universal Object, Universal Construction, and Consensus Object	371
14.1.1	Universal (Synchronization) Object and Universal Construction	371
14.1.2	The Notion of a Consensus Object	372
14.2	Inputs and Base Principles of Universal Constructions	373
14.2.1	The Specification of the Constructed Object	373
14.2.2	Base Principles of Universal Constructions	374
14.3	An Unbounded Wait-Free Universal Construction.	374
14.3.1	Principles and Description of the Construction	375
14.3.2	Proof of the Construction.	378
14.3.3	Non-deterministic Objects	382
14.3.4	Wait-Freedom Versus Bounded Wait-Freedom.	383
14.4	A Bounded Wait-Free Universal Construction	384
14.4.1	Principles of the Construction	384
14.4.2	Proof of the Construction.	388
14.4.3	Non-deterministic Objects	391
14.5	From Binary Consensus to Multi-Valued Consensus	391
14.5.1	A Construction Based on the Bit Representation of Proposed Values.	392
14.5.2	A Construction for Unbounded Proposed Values	394
14.6	Summary	395
14.7	Bibliographic Notes	396
14.8	Exercises and Problems	396
15	The Case of Unreliable Base Objects.	399
15.1	Responsive Versus Non-responsive Crash Failures	400
15.2	SWSR Registers Prone to Crash Failures.	400
15.2.1	Reliable Register When Crash Failures Are Responsive: An Unbounded Construction	401

15.2.2	Reliable Register When Crash Failures Are Responsive: A Bounded Construction	403
15.2.3	Reliable Register When Crash Failures Are Not Responsive: An Unbounded Construction	406
15.3	Consensus When Crash Failures Are Responsive: A Bounded Construction	408
15.3.1	The “Parallel Invocation” Approach Does Not Work	408
15.3.2	A t -Tolerant Wait-Free Construction	409
15.3.3	Consensus When Crash Failures Are Not Responsive: An Impossibility	410
15.4	Omission and Arbitrary Failures	410
15.4.1	Object Failure Modes	410
15.4.2	Simple Examples	412
15.4.3	Graceful Degradation	413
15.4.4	Fault-Tolerance Versus Graceful Degradation	417
15.5	Summary	418
15.6	Bibliographic Notes	419
15.7	Exercises and Problems	419
16	Consensus Numbers and the Consensus Hierarchy	421
16.1	The Consensus Number Notion	421
16.2	Fundamentals	422
16.2.1	Schedule, Configuration, and Valence	422
16.2.2	Bivalent Initial Configuration	423
16.3	The Weak Wait-Free Power of Atomic Registers	425
16.3.1	The Consensus Number of Atomic Read/Write Registers Is 1	425
16.3.2	The Wait-Free Limit of Atomic Registers	428
16.4	Objects Whose Consensus Number Is 2.	429
16.4.1	Consensus from Test&Set Objects	429
16.4.2	Consensus from Queue Objects	431
16.4.3	Consensus from Swap Objects	432
16.4.4	Other Objects for Wait-Free Consensus in a System of Two Processes	432
16.4.5	Power and Limit of the Previous Objects.	433
16.5	Objects Whose Consensus Number Is $+\infty$	438
16.5.1	Consensus from Compare&Swap Objects	439
16.5.2	Consensus from Mem-to-Mem-Swap Objects	440
16.5.3	Consensus from an Augmented Queue	442
16.5.4	From a Sticky Bit to Binary Consensus	442
16.5.5	Impossibility Result	443

16.6	Hierarchy of Atomic Objects	443
16.6.1	From Consensus Numbers to a Hierarchy	443
16.6.2	On Fault Masking	444
16.6.3	Robustness of the Hierarchy	445
16.7	Summary	445
16.8	Bibliographic Notes	445
16.9	Exercises and Problems	446
17	The Alpha(s) and Omega of Consensus:	
	Failure Detector-Based Consensus	449
17.1	De-constructing Compare&Swap	450
17.2	A Liveness-Oriented Abstraction: The Failure Detector Ω . . .	452
17.2.1	Definition of Ω	452
17.2.2	Ω -Based Consensus: Ω as a Resource Allocator or a Scheduler	453
17.3	Three Safety-Oriented Abstractions: Alpha ₁ , Alpha ₂ , and Alpha ₃	454
17.3.1	A Round-Free Abstraction: Alpha ₁	454
17.3.2	A Round-Based Abstraction: Alpha ₂	455
17.3.3	Another Round-Free Abstraction: Alpha ₃	456
17.3.4	The Rounds Seen as a Resource	457
17.4	Ω -Based Consensus	457
17.4.1	Consensus from Alpha ₁ Objects and Ω	457
17.4.2	Consensus from an Alpha ₂ Object and Ω	459
17.4.3	Consensus from an Alpha ₃ Object and Ω	460
17.4.4	When the Eventual Leader Elected by Ω Does Not Participate	463
17.4.5	The Notion of an Indulgent Algorithm	464
17.4.6	Consensus Object Versus Ω	464
17.5	Wait-Free Implementations of the Alpha ₁ and Alpha ₂ Abstractions	465
17.5.1	Alpha ₁ from Atomic Registers	465
17.5.2	Alpha ₂ from Regular Registers	467
17.6	Wait-Free Implementations of the Alpha ₂ Abstraction from Shared Disks	472
17.6.1	Alpha ₂ from Unreliable Read/Write Disks	472
17.6.2	Alpha ₂ from Active Disks	476
17.7	Implementing Ω	477
17.7.1	The Additional Timing Assumption <i>EWB</i>	478
17.7.2	An <i>EWB</i> -Based Implementation of Ω	479
17.7.3	Proof of the Construction	481
17.7.4	Discussion	484

17.8	Summary	485
17.9	Bibliographic Notes	485
17.10	Exercises and Problems	486
Afterword		489
Bibliography		495
Index		509



<http://www.springer.com/978-3-642-32027-9>

Concurrent Programming: Algorithms, Principles, and
Foundations

Raynal, M.

2013, XXXII, 516 p.,

ISBN: 978-3-642-32027-9