# 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
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.7</td>
<td>A Fast Mutex Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>2.1.8</td>
<td>Mutual Exclusion in a Synchronous System</td>
<td>37</td>
</tr>
<tr>
<td>2.2</td>
<td>Mutex Based on Specialized Hardware Primitives</td>
<td>38</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Test&amp;Set, Swap and Compare&amp;Swap</td>
<td>39</td>
</tr>
<tr>
<td>2.2.2</td>
<td>From Deadlock-Freedom to Starvation-Freedom</td>
<td>40</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Fetch&amp;Add</td>
<td>44</td>
</tr>
<tr>
<td>2.3</td>
<td>Mutex Without Atomicity</td>
<td>45</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Safe, Regular and Atomic Registers</td>
<td>45</td>
</tr>
<tr>
<td>2.3.2</td>
<td>The Bakery Mutex Algorithm</td>
<td>48</td>
</tr>
<tr>
<td>2.3.3</td>
<td>A Bounded Mutex Algorithm</td>
<td>53</td>
</tr>
<tr>
<td>2.4</td>
<td>Summary</td>
<td>58</td>
</tr>
<tr>
<td>2.5</td>
<td>Bibliographic Notes</td>
<td>58</td>
</tr>
<tr>
<td>2.6</td>
<td>Exercises and Problems</td>
<td>59</td>
</tr>
</tbody>
</table>

3 Lock-Based Concurrent Objects .......................................................... 61

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Concurrent Objects</td>
<td>61</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Concurrent Object</td>
<td>61</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Lock-Based Implementation</td>
<td>62</td>
</tr>
<tr>
<td>3.2</td>
<td>A Base Synchronization Object: the Semaphore</td>
<td>63</td>
</tr>
<tr>
<td>3.2.1</td>
<td>The Concept of a Semaphore</td>
<td>63</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Using Semaphores to Solve the Producer-Consumer Problem</td>
<td>65</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Using Semaphores to Solve a Priority Scheduling Problem</td>
<td>71</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Using Semaphores to Solve the Readers-Writers Problem</td>
<td>74</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Using a Buffer to Reduce Delays for Readers and Writers</td>
<td>78</td>
</tr>
<tr>
<td>3.3</td>
<td>A Construct for Imperative Languages: the Monitor</td>
<td>81</td>
</tr>
<tr>
<td>3.3.1</td>
<td>The Concept of a Monitor</td>
<td>82</td>
</tr>
<tr>
<td>3.3.2</td>
<td>A Rendezvous Monitor Object</td>
<td>83</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Monitors and Predicates</td>
<td>85</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Implementing a Monitor from Semaphores</td>
<td>87</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Monitors for the Readers-Writers Problem</td>
<td>89</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Scheduled Wait Operation</td>
<td>94</td>
</tr>
<tr>
<td>3.4</td>
<td>Declarative Synchronization: Path Expressions</td>
<td>95</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Definition</td>
<td>96</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Using Path Expressions to Solve Synchronization Problems</td>
<td>97</td>
</tr>
<tr>
<td>3.4.3</td>
<td>A Semaphore-Based Implementation of Path Expressions</td>
<td>98</td>
</tr>
<tr>
<td>3.5</td>
<td>Summary</td>
<td>101</td>
</tr>
<tr>
<td>3.6</td>
<td>Bibliographic Notes</td>
<td>102</td>
</tr>
<tr>
<td>3.7</td>
<td>Exercises and Problems</td>
<td>102</td>
</tr>
</tbody>
</table>
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 $\text{store}_\text{collect}()$ 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

9.1 Renaming Objects

9.1.1 The Base Renaming Problem

9.1.2 One-Shot Renaming Object

9.1.3 Adaptive Implementations

9.1.4 A Fundamental Result

9.1.5 Long-Lived Renaming

9.2 Non-triviality of the Renaming Problem

9.3 A Splitter-Based Optimal Time-Adaptive Implementation

9.4 A Snapshot-Based Optimal Size-Adaptive Implementation

9.4.1 A Snapshot-Based Implementation

9.4.2 Proof of the Implementation

9.5 Recursive Store-Collect-Based Size-Adaptive Implementation

9.5.1 A Recursive Renaming Algorithm

9.5.2 An Example

9.5.3 Proof of the Renaming Implementation

9.6 Variant of the Previous Recursion-Based Renaming Algorithm

9.6.1 A Renaming Implementation Based on Immediate Snapshot Objects

9.6.2 An Example of a Renaming Execution

9.7 Long-Lived Perfect Renaming Based on Test&Set Registers

9.7.1 Perfect Adaptive Renaming

9.7.2 Perfect Long-Lived Test&Set-Based Renaming

9.8 Summary

9.9 Bibliographic Notes

9.10 Exercises and Problems

Part IV The Transactional Memory Approach

10 Transactional Memory

10.1 What Are Software Transactional Memories

10.1.1 Transactions = High-Level Synchronization

10.1.2 At the Programming Level

10.2 STM System

10.2.1 Speculative Executions, Commit and Abort of a Transaction

10.2.2 An STM Consistency Condition: Opacity

10.2.3 An STM Interface

10.2.4 Incremental Reads and Deferred Updates
11.4 Three Unbounded Constructions
11.4.1 SWSR Registers:
From Unbounded Regular to Atomic
11.4.2 Atomic Registers:
From Unbounded SWSR to SWMR
11.4.3 Atomic Registers:
From Unbounded SWMR to MWMR
11.5 Summary
11.6 Bibliographic Notes

12 From Safe Bits to Atomic Bits:
Lower Bound and Optimal Construction
12.1 A Lower Bound Theorem
12.1.1 Two Preliminary Lemmas
12.1.2 The Lower Bound Theorem
12.2 A Construction of an Atomic Bit from Three Safe Bits
12.2.1 Base Architecture of the Construction
12.2.2 Underlying Principle and Signaling Scheme
12.2.3 The Algorithm Implementing
the Operation \( R.write() \)
12.2.4 The Algorithm Implementing
the Operation \( R.read() \)
12.2.5 Cost of the Construction
12.3 Proof of the Construction of an Atomic Bit
12.3.1 A Preliminary Theorem
12.3.2 Proof of the Construction
12.4 Summary
12.5 Bibliographic Notes
12.6 Exercise

13 Bounded Constructions of Atomic \( b \)-Valued Registers
13.1 Introduction
13.2 A Collision-Free (Pure Buffers) Construction
13.2.1 Internal Representation of the Atomic \( b \)-Valued Register \( R \)
13.2.2 Underlying Principle: Two-Level Switch
to Ensure Collision-Free Accesses to Buffers
13.2.3 The Algorithms Implementing
the Operations \( R.write() \) and \( R.read() \)
13.2.4 Proof of the Construction: Collision-Freedom
13.2.5 Correctness Proof
13.3 A Construction Based on Impure Buffers
13.3.1 Internal Representation of the Atomic \( b \)-Valued Register \( R \)
| 13.3.2 | An Incremental Construction ..................................... | 358 |
| 13.3.3 | The Algorithms Implementing the Operations \texttt{R.write()} and \texttt{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

15.2.3 Reliable Register When Crash Failures Are Not Responsive: An Unbounded Construction

15.3 Consensus When Crash Failures Are Responsive: A Bounded Construction

15.3.1 The “Parallel Invocation” Approach Does Not Work

15.3.2 A \( t \)-Tolerant Wait-Free Construction

15.3.3 Consensus When Crash Failures Are Not Responsive: An Impossibility

15.4 Omission and Arbitrary Failures

15.4.1 Object Failure Modes

15.4.2 Simple Examples

15.4.3 Graceful Degradation

15.4.4 Fault-Tolerance Versus Graceful Degradation

15.5 Summary

15.6 Bibliographic Notes

15.7 Exercises and Problems

16 Consensus Numbers and the Consensus Hierarchy

16.1 The Consensus Number Notion

16.2 Fundamentals

16.2.1 Schedule, Configuration, and Valence

16.2.2 Bivalent Initial Configuration

16.3 The Weak Wait-Free Power of Atomic Registers

16.3.1 The Consensus Number of Atomic Read/Write Registers Is 1

16.3.2 The Wait-Free Limit of Atomic Registers

16.4 Objects Whose Consensus Number Is 2

16.4.1 Consensus from Test&Set Objects

16.4.2 Consensus from Queue Objects

16.4.3 Consensus from Swap Objects

16.4.4 Other Objects for Wait-Free Consensus in a System of Two Processes

16.4.5 Power and Limit of the Previous Objects

16.5 Objects Whose Consensus Number Is \( +\infty \)

16.5.1 Consensus from Compare&Swap Objects

16.5.2 Consensus from Mem-to-Mem-Swap Objects

16.5.3 Consensus from an Augmented Queue

16.5.4 From a Sticky Bit to Binary Consensus

16.5.5 Impossibility Result
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 \( \Omega \) . . 452
  17.2.1 Definition of \( \Omega \) ................................................. 452
  17.2.2 \( \Omega \)-Based Consensus:
      \( \Omega \) 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 \( \Omega \)-Based Consensus ................................................... 457
  17.4.1 Consensus from Alpha\( _1 \) Objects and \( \Omega \) ............. 457
  17.4.2 Consensus from an Alpha\( _2 \) Object and \( \Omega \) ............ 459
  17.4.3 Consensus from an Alpha\( _3 \) Object and \( \Omega \) ............ 460
  17.4.4 When the Eventual Leader Elected by \( \Omega \) Does Not Participate ......................................................... 463
  17.4.5 The Notion of an Indulgent Algorithm ......................... 464
  17.4.6 Consensus Object Versus \( \Omega \) .................. 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 \( \Omega \) .......................................................... 477
  17.7.1 The Additional Timing Assumption \( EWB \) ................. 478
  17.7.2 An \( EWB \)-Based Implementation of \( \Omega \) .......... 479
  17.7.3 Proof of the Construction ........................................ 481
  17.7.4 Discussion .......................................................... 484
Concurrent Programming: Algorithms, Principles, and Foundations
Raynal, M.
2013, XXXII, 516 p., Hardcover
ISBN: 978-3-642-32026-2