

second edition

# PRINCIPLES <sup>OF</sup> CONCURRENT <sup>AND</sup> DISTRIBUTED PROGRAMMING

M. BEN-ARI

 ADDISON-WESLEY



We work with leading authors to develop the strongest educational materials in computing, bringing cutting-edge thinking and best learning practice to a global market.

Under a range of well-known imprints, including Addison-Wesley, we craft high quality print and electronic publications which help readers to understand and apply their content, whether studying or at work.

To find out more about the complete range of our publishing, please visit us on the World Wide Web at: [www.pearsoned.co.uk](http://www.pearsoned.co.uk)

# Principles of Concurrent and Distributed Programming

## Table of Contents

Front Cover

Contents

Preface

Chapter 1: What is Concurrent Programming?

1.1: Introduction

1.2: Concurrency as abstract parallelism

1.3: Multitasking

1.4: The terminology of concurrency

1.5: Multiple computers

1.6: The challenge of concurrent programming

Transition

Chapter 2: The Concurrent Programming Abstraction

2.1: The role of abstraction

2.2: Concurrent execution as interleaving of atomic statements

States

Scenarios

2.3: Justification of the abstraction

Multitasking systems

Multiprocessor computers

Distributed systems

2.4: Arbitrary interleaving

2.5: Atomic statements

2.6: Correctness

Linear and branching temporal logics

2.7: Fairness

2.8: Machine-code instructions

Register machines

Stack machines

Source statements and machine instructions

2.9: Volatile and non-atomic variables

2.10: The BACI concurrency simulator

2.11: Concurrency in Ada

Volatile and atomic

2.12: Concurrency in Java

Volatile

# Table of Contents

- 2.13: Writing concurrent programs in Promela
- 2.14: Supplement: the state diagram for the frog puzzle
- Transition
- Exercises

## Chapter 3: The Critical Section Problem

- 3.1: Introduction
- 3.2: The definition of the problem
- 3.3: First attempt
- 3.4: Proving correctness with state diagrams
  - States
  - State diagrams
  - Abbreviating the state diagram
- 3.5: Correctness of the first attempt
- 3.6: Second attempt
- 3.7: Third attempt
- 3.8: Fourth attempt
- 3.9: Dekkers algorithm
- 3.10: Complex atomic statements
- Transition
- Exercises

## Chapter 4: Verification of Concurrent Programs

- 4.1: Logical specification of correctness properties
- 4.2: Inductive proofs of invariants
  - Proof of mutual exclusion for the third attempt
- 4.3: Basic concepts of temporal logic
  - Always and eventually
  - Duality
  - Sequences of operators
- 4.4: Advanced concepts of temporal logic
  - Until
  - Next
  - Deduction with temporal operators
  - Specifying overtaking
- 4.5: A deductive proof of Dekkers algorithm
  - Reasoning about progress
  - A proof of freedom from starvation
- 4.6: Model checking
- 4.7: Spin and the Promela modeling language
- 4.8: Correctness specifications in Spin

# Table of Contents

4.9: Choosing a verification technique

Transition

Exercises

## Chapter 5: Advanced Algorithms for the Critical Section Problem

5.1: The bakery algorithm

5.2: The bakery algorithm for N processes

5.3: Less restrictive models of concurrency

5.4: Fast algorithms

Outline of the fast algorithm

Partial proof of the algorithm

Generalization to N processes

5.5: Implementations in Promela

Transition

Exercises

## Chapter 6: Semaphores

6.1: Process states

6.2: Definition of the semaphore type

6.3: The critical section problem for two processes

6.4: Semaphore invariants

6.5: The critical section problem for N processes

6.6: Order of execution problems

6.7: The producerconsumer problem

Infinite buffers

Bounded buffers

Split semaphores

6.8: Definitions of semaphores

Strong semaphores

Busy-wait semaphores

Abstract definitions of semaphores

6.9: The problem of the dining philosophers

6.10: Barz's simulation of general semaphores

6.11: Udding's starvation-free algorithm

6.12: Semaphores in BACI

6.13: Semaphores in Ada

6.14: Semaphores in Java

6.15: Semaphores in Promela

Proving Barz's algorithm in Spin

Transition

# Table of Contents

Exercises

## Chapter 7: Monitors

7.1: Introduction

7.2: Declaring and using monitors

7.3: Condition variables

    Simulating semaphores

    Operations on condition variables

    Correctness of the semaphore simulation

7.4: The producerconsumer problem

7.5: The immediate resumption requirement

7.6: The problem of the readers and writers

7.7: Correctness of the readers and writers algorithm

7.8: A monitor solution for the dining philosophers

7.9: Monitors in BACI

7.10: Protected objects

    Protected objects in Ada

7.11: Monitors in Java

    Synchronized blocks

7.12: Simulating monitors in Promela

Transition

Exercises

## Chapter 8: Channels

8.1: Models for communications

    Synchronous vs. asynchronous communications

    Addressing

    Data flow

    CSP and occam

8.2: Channels

8.3: Parallel matrix multiplication

    Selective input

8.4: The dining philosophers with channels

8.5: Channels in Promela

8.6: Rendezvous

    The rendezvous in Ada

8.7: Remote procedure calls

Transition

Exercises

## Chapter 9: Spaces

# Table of Contents

- 9.1: The Linda model
- 9.2: Expressiveness of the Linda model
- 9.3: Formal parameters
- 9.4: The masterworker paradigm
  - Granularity
- 9.5: Implementations of spaces
  - C-Linda
  - JavaSpaces
  - Java
  - Promela
- Transition
- Exercises

## Chapter 10: Distributed Algorithms

- 10.1: The distributed systems model
  - Communications channels
  - Sending and receiving messages
  - Concurrency within the nodes
  - Studying distributed algorithms
- 10.2: Implementations
- 10.3: Distributed mutual exclusion
  - Initial development of the algorithm
  - The scenario of an example
  - Equal ticket numbers
  - Choosing ticket numbers
  - Quiescent nodes
- 10.4: Correctness of the RicartAgrawala algorithm
- 10.5: The RA algorithm in Promela
- 10.6: Token-passing algorithms
- 10.7: Tokens in virtual trees
- Transition
- Exercises

## Chapter 11: Global Properties

- 11.1: Distributed termination
  - Preliminary algorithm
  - Correctness of the preliminary algorithm
- 11.2: The DijkstraScholten algorithm
  - Correctness of the DijkstraScholten algorithm
  - Performance
- 11.3: Credit-recovery algorithms

# Table of Contents

## 11.4: Snapshots

Correctness of the ChandyLamport algorithm

Transition

Exercises

## Chapter 12: Consensus

### 12.1: Introduction

### 12.2: The problem statement

### 12.3: A one-round algorithm

### 12.4: The Byzantine Generals algorithm

### 12.5: Crash failures

### 12.6: Knowledge trees

### 12.7: Byzantine failures with three generals

### 12.8: Byzantine failures with four generals

Correctness

Complexity

### 12.9: The flooding algorithm

### 12.10: The King algorithm

Correctness

Complexity

### 12.11: Impossibility with three generals

Transition

Exercises

## Chapter 13: Real-Time Systems

### 13.1: Introduction

The Ada Real-Time Annex

### 13.2: Definitions

### 13.3: Reliability and repeatability

The Ariane 5 rocket

The space shuttle

### 13.4: Synchronous systems

### 13.5: Asynchronous systems

Priorities and preemptive scheduling

### 13.6: Interrupt-driven systems

Interrupt overflow in the Apollo 11 lunar module

### 13.7: Priority inversion and priority inheritance

Priority inheritance

Priority inversion from queues

Priority ceiling locking

The Mars Pathfinder



# Table of Contents

13.8: The Mars Pathfinder in Spin

13.9: Simpsons four-slot algorithm

13.10: The Ravenscar profile

Suspension objects

13.11: UPPAAL

13.12: Scheduling algorithms for real-time systems

The rate monotonic algorithm

The earliest deadline first algorithm

Transition

Exercises

## A: The Pseudocode Notation

Structure

Syntax

Semantics

Synchronization constructs

## B: Review of Mathematical Logic

B.1: The propositional calculus

Syntax

Semantics

Logical equivalence

B.2: Induction

B.3: Proof methods

Model checking

Deductive proof

Material implication

B.4: Correctness of sequential programs

Verification in SPARK

## C: Concurrent Programming Problems

## D: Software Tools

D.1: BACI and jBACI

D.2: Spin and jSpin

The jSpin user interface

How Spin verifies properties

D.3: DAJ

## E: Further Reading

Websites

## Bibliography

## Index

# Table of Contents

Back Cover