Book Review of

Synchronization Algorithms and Concurrent Programming

Author of Book: Gadi Taubenfeld

Publisher: Pearson / Prentice Hall, 2006, 433 pages

Reviewed by Danny Hendler, Ben-Gurion University, hendlerd@cs.bgu.ac.il

1 Introduction

Although there are quite a few books that deal with various aspects of the theory and practice of multiprocessor synchronization, only a handful of them fall into the category of ‘textbooks’ ([2, 13, 18] are notable examples); Gadi Taubenfeld’s recent book, “Synchronization Algorithms and Concurrent Programming”, is certainly one of them.

Taubenfeld’s book focuses on inter-process synchronization in shared-memory systems. The book provides an in-depth discussion of algorithms and lower bounds for a wide range of synchronization problems, such as mutual exclusion, barrier synchronization, l-exclusion, deadlock-prevention, the dining philosophers problem, the producer-consumer problem, consensus, and many more.

As stated in its preface, the book is meant to be used as a textbook for upper-level undergraduate or graduate courses and special emphasis has been given to make it accessible to a wide audience. The book’s structure and methodology reflect these goals in several ways:

- The book is mostly self-contained.
- Each section ends with a set of self review questions and their solutions.
- Each chapter ends with a section that contains numerous exercises, categorized according to their estimated difficulty level. For example, the book contains more than 160 mutual-exclusion related exercises. Many of these exercises suggest variants of the algorithms presented in the text and ask the student to analyze their properties. While teaching mutual exclusion algorithms, I found these exercises very constructive in providing intuitions and in deepening students’ understanding of synchronization algorithms.
- There is a Companion Web-site that contains PowerPoint presentations covering most of the book’s chapters and all of the figures that appear in it.

2 Summary of Contents

The first chapter demonstrates the difficulty of inter-process synchronization by two examples that appeal to the reader’s intuition: the Too Much Milk problem (an allegory illustrating the mutual exclusion problem, due to Prof. John Ousterhout) and the Coordinated Attack problem (a.k.a. *the two generals problem* [10]). It then formally introduces the mutual exclusion problem, the complexity measures used to analyze it, and some basic concepts of shared-memory multiprocessors.

Chapters 2 and 3 provide a comprehensive discussion of mutual exclusion using atomic read/write registers. Chapter 2 presents classic mutual exclusion algorithms, such as Peterson and Kessels' 2-process algorithms [14, 21], tournament algorithms, Lamport's fast mutual exclusion algorithm [16], and the bakery algorithm [15]. Correctness proofs for all these algorithms are provided, as well as the linear space lower bound on mutual exclusion using read/write registers due to Burns and Lynch [4]. Some recent results by the author (the Black-Write bakery algorithm [25] and automatic discovery of mutual-exclusion algorithms [3]) are also described. Chapter 3 deals with more advanced read/write mutual exclusion topics, such as local-spin algorithms, adaptive and fault-tolerant mutual exclusion algorithms and impossibility results, and symmetric mutual-exclusion algorithms.

Chapter 4 discusses blocking and non-blocking synchronization using strong read-modify-write primitives. Section 4.1 introduces strong synchronization primitives. Section 4.2 describes techniques for collision avoidance using test-and-set bits. Section 4.3 describes the Ticket algorithm [5]. Section 4.4 presents three queue-based local-spin mutual-exclusion algorithms: Anderson's algorithm [1], Graunke and Thakkar's algorithm [9], and the MCS algorithm [19]. Section 4.5 discusses non-blocking and wait-free implementations of concurrent objects and presents Michael and Scott's queue algorithm [20]. Chapter 4 concludes with a treatise of semaphores, monitors, and notions of concurrent object fairness.

Chapter 5 discusses the issue of barrier synchronization. Several barrier algorithms that use an atomic counter are presented, followed by test-and-set-based barriers, a combining tree barrier, a tree-based barrier [19], the dissemination barrier [11], and the See-Saw barrier (based on [7]).

Chapter 6 deals with the l-exclusion problem. It presents two read/write algorithms due to Peterson [22], followed by a few algorithms that use strong synchronization primitives. The chapter concludes with a discussion of the *l-assignment* problem, which is a stronger version of l-exclusion.

Chapter 7 discusses problems that involve multiple resources. After defining the concept of deadlock in such problems, a few techniques of deadlock-prevention are described, followed by a discussion of the Deadly Embrace problem. The second part of this chapter provides an in-depth discussion of the Dining Philosophers problem.

Chapter 8 presents additional classical synchronization problems, such as the Producer-Consumer problem, the Readers and Writers problem, the Sleeping Barber and the Cigarette Smoker's problems, group mutual exclusion, room synchronization, and more.

Chapter 9 deals with the consensus problem. Apart from the classical results (the impossibility of consensus with one faulty process [6, 17], Herlihy's wait-free hierarchy and the universality of consensus [12]), some less known consensus algorithms are presented, such as an algorithm of Fischer et al. for achieving consensus without memory initialization [8].

The book concludes with Chapter 10, which presents several timing-based mutual-exclusion, fast mutual-exclusion and consensus algorithms, for both the known- and the unknown-delays models.

3 Personal Teaching Experience

I am using "Synchronization Algorithms and Concurrent Programming" for teaching two courses: an undergraduate course on Operating Systems and a graduate course on multiprocessor synchronization algorithms.

The classic Operating Systems textbooks, such as the books by Tanenbaum [24] and Silberschatz et al. [23], do provide some material on shared-memory synchronization. In light of the recent emergence of multi-core computing, however, I maintain that undergraduate students should gain deep understanding of shared-memory synchronization; some important synchronization topics, such as the use of strong read-modify-write primitives and local-spin algorithms, are not sufficiently covered by these classic OS textbooks. I am therefore using Taubenfeld's book, in addition to these classic textbooks, for providing a deeper treatise of mutual exclusion algorithms, semaphores, monitors, and other synchronization problems, such as the dining philosophers problem.

The graduate course I teach deals with both theoretical and practical aspects of multiprocessor synchronization algorithms. For this course, I am using Taubenfeld's book, in addition to other Distributed Computing textbooks [2, 13]. My first presentation for this course uses the Too Much Milk and the Coordinated Attack problems to provide basic intuitions into the difficulty of devising correct synchronization algorithms.

The first part of my graduate course deals with the mutual exclusion problem. Here, I am using many of the slides available in the book's companion web-site for illustrating the ideas underlying classical algorithms, such as Lamport's fast mutual exclusion algorithm [16], and the bakery algorithm [15]. Lamport's Bakery algorithm is presented in Taubenfeld's book and slides by a sequence of less and less naïve "approximations" of the original algorithm. This methodology is also used for several other mutual exclusion algorithms, and both my students and myself find it very useful. The numerous synchronization exercises provided by Taubenfeld's book make preparing home assignments and exams for both these courses much easier.

In summary, "Synchronization Algorithms and Concurrent Programming" is, in my opinion, a fine addition to the Distributed Computing bookshelf. Its structure and methodology make it accessible to a wide audience, which make it useful as a textbook. The rich variety of synchronization research results covered by it make this book valuable also as a reference for researchers in the Distributed Computing community.

References

[1] T. E. Anderson. The performance of spin lock alternatives for shared-memory multiprocessor. *IEEE Trans. on Parallel and Distributed Systems*, 1(1):6-16, 1990.

[2] H. Attiya and J. Welch. Distributed Computing: Fundamentals, Simulations and Advanced Topics, 2'nd Edition. John Wiley and Sons, 2004.

[3] Y. Bar-David and G. Taubenfeld. Automatic discovery of mutual exclusion algorithms. In *Proc. 17th International Symp. on Distributed Computing*, LNCS 2648, Springer-Verlag, 2003, 136-150.

[4] J.E. Burns and N. A. Lynch. Bounds on shared-memory for mutual exclusion. *Information and Computation*, 107(2):171-184, December 1993.

[5] M. J. Fischer, N. A. Lynch, J. E. Burns, and A. Borodin. Distributed FIFO allocation of identical resources using small shared space. *ACM Trans. on Programming Languages and Systems*, 11(1):90-114, 1989.

[6] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374-382, 1985.

[7] M. J. Fischer, S. Moran, S. Rudich, and G. Taubenfeld. The wakeup problem. *SIAM Journal on Computing*, 25(6):1332-1357, 1996.

[8] M. J. Fischer, S. Moran, and G. Taubenfeld. Space-efficient asynchrnous consensus without shared memory initialization. *Infomration Processing Letters*, 45(2):101-105, 1993.

[9] G. Graunke and S. Thakkar. Synchronization algorithms for shared-memory multiprocessors. *IEEE Computers*, 23(6):60-69, 1990.

[10] J. Gray, Notes on database operating systems. *Operating Systems, an Advanced Course*, Lecture Notes in Computer Science, 60:393-481, 1978.

[11] D. Hensgen, R. Finkel, and U. Manber. Two algorithms for barrier synchronization. *International Journal of Parallel Programming*, 17(1):1-17, 1988.

[12] M. P. Herlihy. Wait-free synchronization. *ACM Trans. on Programming Languages and Systems*, 13(1):124-149, 1991.

[13] M. Herlihy and N. Shavit. The Art of Multiprocessor Programming. Morgan Kaufman, 2008.

[14] J. L. W. Kessels. Arbitration without common modifiable variables. *Acta Informatica*, 17(2):135-141, 1982.

[15] L. Lamport. A new solution of Dijkstra's concurrent programming problem. *Communications of the ACM*, 17(8):453-455, 1974.

[16] L. Lamport. A fast mutual exclusion algorithm. *ACM Trans. on Computer Systems*, 5(1):1-11, 1987.

[17] M. C. Loui and H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4:163-183, 1987.

[18] N. Lynch. Distributed Algorithms. Morgan Kaufman, 1996.

[19] J.M Mellor-Crummey and M. L. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. *ACM Trans. on Computer Systems*, 9(1):21-65, 1991.

[20] M. M. Michael and M. L Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *Proc. 15th ACM Symp. on Principles of Distributed Computing*, 267-275, 1996.

[21] G. L. Peterson. Myths about the mutual exclusion problem. *Information Processing Letters*, 12(3):115-116, 1981.

[22] G. L. Peterson. Observations on l-exclusion.In *Proc. 28th Annual Allerton Conference on Communication, Control and Computing*, pages 568-577, October 1990.

[23] A. Silberschatz, P. Galvin and G Gagne. Operating System Concepts, 6'th Edition. John Wiley and Sons, 2003.

[24] A. Tanenbaum. Modern Operating Systems, 3'rd Edition. Pearson / Prentice Hall, 2008.

[25] G. Taubenfeld. The black-white bakery algorithm. In *Proc. 18th International Symp. on Distributed Computing*, LNCS 3274, Springer-Verlag, 2004, 56-70.