The Book Review Column[1]

by William Gasarch
Department of Computer Science
University of Maryland at College Park
College Park, MD, 20742
email: gasarch@cs.umd.edu

In this column we review the following books.

1. Joint review of **Computational Complexity: A Conceptual Perspective** by Oded Goldreich and **Computational Complexity: A Modern Approach** by Sanjeev Arora and Boaz Barak. These are textbooks for a basic graduate course in complexity theory that will define the field for years to come. Review by Daniel Apon.

2. **Algorithmic Game Theory** A collection of articles on Algorithmic Game Theory edited by Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. Review by David Levin.

3. **The** P = NP **Question and Godel's Lost Letter** by Richard J. Lipton. This book is excerpts from Richard Lipton's Blog of the same name (except that the two clauses are reversed). Can a blog be a good book? This may be the wave of the future. Review by Bill Gasarch, a fellow blogger, who thinks YES!

4. **The Pea and the Sun: A Mathematical Paradox** by Leonard Wapner. The Banach-Tarski Paradox says that you can take a sphere (e.g., a pea) and break it into a finite number of pieces, reassemble them, and end up with a sphere of twice the volume. If you keep doing this you will end up with a large sphere (e.g., the size of the sun). You might not like it (I don't) but you should learn more about it so that you can say why you don't like it. This book is an excellent place to find out more about it. Review by Bill Gasarch.

5. **Coding for Data and Computer Communications** by by David Salomon. The title says it all. Review by by David Werden.

6. **Binary Quadratic Forms: An Algorithmic Approach** by by Johannes Buchmann and Ulrich Vollmer. Say you want to find all integer solutions to $aX^2 + bXY + cY^2 = 0$. What would you do? I would read this book. Review by Graham Coleman.

7. **Elliptic Curves** by Lawrence C. Washington. Elliptic Curves are a mathematical object (duh) which have recently gotten much more attention than is common for a mathematical object because of its use in cryptography. Do you want to learn more about them? Then read this book! Review by David Chen.

8. **Concurrent Zero-Knowledge** by Alon Rosen. How many rounds do you need for a Zero Knowledge Protocol? This monograph explores this question and gives matching upper and lower bounds in the black box model. Review by Sarvagya Upadhyay.

9. **Introduction to Cryptography** by Hans Delfs and Helmut Knebl. This is an undergraduate textbook in cryptography. Review by Cillian Murphy.

10. **Introduction to Modern Cryptography** by Jonathan Katz and Yehuda Lindell. This is an undergraduate textbook in cryptography. Review by Ben Fulton. The review is followed by a note from the authors.

11. **An Introduction to Mathematical Cryptography** by Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman. This is an undergraduate textbook in cryptography. Review by Sarah Meiklejohn.

12. **Software Abstractions: Logic, Language and Analysis** by Daniel Jackson. This book is about Alloy, a modeling language which allows programmers to specify their designs and to build software models *incrementally* via the Alloy analyzer. Review by Andrew C. Lee.

BOOKS I NEED REVIEWED FOR SIGACT NEWS COLUMN
Algorithms and Related Topics

1. *Algorithmic Algebraic Combinatorial and Grobner Bases* Edited by Klin, Jones, Jurisic, Muzychuk, Ponomarnko.

2. *Computational Topology* by Edelsbrunner and Harer.

3. *Fast Algorithms for Signal Processing* by Blahut.

4. *Information Retrieval: Implementing and Evaluating Search Engines* by Buttcher, Clarke, and Cormack.

5. *Game of Life Cellular Automata* Edited by Adamatzky (This really is the title.)

Cryptography and Coding Theory

1. *The Cryptoclub: Using Mathematics to Make and Break Secret Codes* by Beissinger and Pless (for middle school students).

2. *Mathematical Ciphers: From Ceaser to RSA* by Anne Young. (For a non-majors course.)

3. *The Mathematics of Coding Theory* by Garrett.

4. *Secure Key Establishment* by Choo.

5. *Insider Threats in Cyber Security* Edited by Probst, Hunker, Gollman, Bishop.

6. *The mathematics of Ciphers: Number Theory and RSA Cryptography* by Coutinho.

7. *Adaptive Cryptographic Access Control* by Kayem, Akl, and Martin.

8. *Spyware and Adware* by Aycock.

Combinatorics and Probability

1. *Erdos on Graphs: His Legacy of Unsolved Problems* Chung and Graham.
2. *Digital Dice: Computational solutions to Practical Probability Problems.* by Nahin.
3. *Elementary Probability for Applications* by Durrett
4. *Integer Partitions* by Andrews and Eriksson.

Semantics and Programming Languages

1. *Drawing Programs: The theory and Practice of Schematic Functional Programming* by Addis and Addis.
2. *From semantics to computer science: Essays in honour of Gilles Kahn* Edited by Bertot, Huet, Levy, Plotkin.
3. *Transitions and Trees: An Introduction to Structural Operational Semantics* by Hans Huttel.

Number Theory

1. *Making Transcendence Transparent: An intuitive approach to classical transcendental number theory* by Burger and Tubbs.
2. *Biscuits of Number Theory* Edited by Author Benjamin and Ezra Brown.
3. *A Guide to Elementary Number Theory* by Underwood Dudley.

Complexity Theory

1. *Deterministic Extraction from Weak Random Sources* by Ariel Gabizon.
2. *Grammatical Inference: Learning Automata and Grammars* by Colin de la Higuera.
3. *P, NP, and NP-completeness: The Basics of Computational Complexity* by Oded Goldreich To quote the book: *The current book is a significant revision of Chapter 2 (and Section 1.2) of the author's book Computational Complexity: A Conceptual Perspective. The revision was aimed at making the book more friendly to the novice. In particular, numerous technical expositions were further detailed and many exercises were added.*

Misc

1. *Polynomia and Related Realms* by Dan Kalman.
2. *Mathematica: A problem centered approach* by Hazrat.
3. *Finite Fields and Applications* by Mullen and Mummert.
4. *Mathematics Everywhere* Edited by Aigner and Behrends.
5. *Mathematical Omnibus: Thirty Lectures on Classic Mathematics* by Fuchs and Tabachnikov.

6. *The Presidential Election Game* by Brams.
7. *Pioneering Women in American Mathematics: The pre-1940 PhD's* By Green and LaDuke.
8. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics* by Platzer.
9. *Roots to Research: A Vertical Development of Mathematical Problems* by Sally and Sally.
10. *The Dots and Boxes Game: Sophisticated Child's play* By Berlekamp.
11. *New Mathematical Diversions* by Martin Gardner.

Joint Review of[2]
Computational Complexity: A Conceptual Perspective
by Oded Goldreich
Published by Cambridge University Press, 2008
606 pages, Hardcover
and
Computational Complexity: A Modern Approach
by Sanjeev Arora and Boaz Barak
Published by Cambridge University Press, 2009
579 pages, Hardcover

Review by
Daniel Apon dapon@uark.edu
University of Arkansas

1 Introduction

Both *Computational Complexity: A Conceptual Perspective* and *Computational Complexity: A Modern Approach* cover the field of computational complexity, which is a (if not *the*) central area in the theoretical foundations of computer science. Both are ultimately concerned with a single, fundamental question: How can we define a reasonable notion of *efficient computation* in the context of perennially limited resources?

The books are designed as textbooks for a course in complexity theory, aimed at early graduate students or even advanced undergraduate students. Every chapter is accompanied by a series of exercises with each book having hundreds of such problems for the reader to use for practice. There is no special prerequisite knowledge necessary to follow either book, aside from a reasonable degree of mathematical maturity, and as such, either could serve well in the role of a self-study text[3]. Additionally, due to their breadth of topics covered, both would be an exceptional reference text for experts in the field.

[2]

[3]Disclaimer: I am originally self-taught in complexity from the Arora/Barak book. On the other hand, I'm not just being polite here: It *is* absolutely possible to use these books in a self-study!

But naturally, the questions arise: How are they any different? How should I pick between the two for my own use? If I already own one, should I consider buying the other? And the answer is *obvious*: Get both! Of course, if this is not an option, then your answer is buried in the sub-title of each – specifically, in the distinction between the authors' choices of *conceptual perspective* and *modern approach*. This will be the driving theme of the review, and I will return to this idea as I review the two books.

2 Summary

Computational Complexity: A Conceptual Perspective

Goldreich's book is divided into ten chapters, beginning with a discussion of models of computation and the history of complexity theory, progressing naturally through the various complexity classes, and ending with more advanced concepts such as probabilistic proof systems, approximation, and average-case complexity. There is also a (quite hefty) set of appendices that comprise nearly a quarter of the pages of the text and that delves into some of the technical details and mathematical background omitted in the regular chapters. There is a strong, chapter-by-chapter, internal consistency to Goldreich's book, and in following his style, this summary of his book will proceed accordingly.

Chapter 1 begins with a high-level survey of the entire field of complexity. Beginning with a definition of Turing machines, this chapter builds up the historical and technical details to firmly place the rest of the book within a meaningful context. Of particular note, the first chapter spends some time to develop the intuitions underlying the theories of computability and complexity, highlighting the Church-Turing Thesis, the Post Correspondence Problem, and the Cobham-Edmonds Thesis. Additionally, there is discussion about uniform vs. non-uniform models of computation, oracle machines, machines that take advice, and a "sanity check" comparison between the Turing machine model of computation and more "real-world" models, such as an abstract RAM machine.

Chapter 2 develops the complexity classes **P**, **NP**, the **P** vs. **NP** question, and the theory of **NP**-completeness. There is a strong emphasis on the concepts of decision problems vs. search problems vs. promise problems. For example, Goldreich defines the search analogs of **P** and **NP**, respectively **PF** and **PC**. Additionally, *Cook-reductions*, *Karp-reductions*, and *Levin-reductions* are each separately developed here.

Chapter 3 is a shorter chapter entitled "Variations on P and NP." This chapter considers variations on the previous chapter's theme. Specifically, the non-uniform complexity class with advice **P/poly** is developed, as well as the Polynomial-time Hierarchy, **PH**. Examples are given of results that are motivated by a collapse of the Polynomial-time Hierarchy, e.g. $\mathbf{NP} \not\subseteq \mathbf{P/poly}$ unless **PH** collapses to its second level, and are related back to the **P** vs. **NP** question.

Chapter 4 formalizes the notion of various *hierarchy theorems*, e.g. there exist languages that can be decided in $O(n^2)$ time that cannot be decided in $O(n)$ time. In particular, the concepts of time- and space-constructable functions are explained, which lead into a discussion of the deterministic time hierarchy, the non-deterministic time hierarchy and the space hierarchy.

Chapter 5 builds an understanding of space complexity from the bottom up. Beginning with a comparison between time and space complexities, the chapter then jumps into a description of logarithmic space **L**, followed by non-deterministic space complexity, and ending with polynomial space **PSPACE** and a "game"-based discussion of the class.

Chapter 6 is devoted to topics in randomness and counting. First, the text considers the formulation of probabilistic complexity classes **BPP**, **RP** and **ZPP**. These classes are then related back to previously discussed classes, such as **P/poly** and the Polynomial-time Hierarchy – specifically **BPP**$\subseteq \Sigma_2$. Then the discussion turns to exact and approximate counting problems, introducing the class **#P** and **#P**-completeness, and relates those concepts to **PH**, **NP** and **NP**-completeness.

Chapter 7, entitled "The Bright Side of Hardness," takes the complexity concepts that have been meticulously developed throughout the book thus far and provides useful applications for some of the more "negative" results. For example, we might not be able to efficiently invert one-way functions, but we can apply that fact in the design of cryptographic systems! Beginning with one-way functions, the chapter proceeds into an exposition of hard-core predicates, followed by average-case hardness arising out of worst-case hardness, Yao's XOR Lemma, list-decoding and hardness amplification.

Chapter 8 picks up from the previous chapter's concepts and develops a theory of pseudorandom generators. In this chapter, the notion of computational indistinguishability is also formalized. A distinction is also made between general-purpose pseudorandom generators (useful for, say, cryptographic applications) and special-purpose pseudorandom generators that are tailored to tackle difficult computational complexity questions. For example, by the existence of a specific, space-bounded pseudorandom generator with exponential stretch in the space bound, we know that the log-space analog of **BPP**, **BPL**, is contained in **DSPACE**$(\log^2)$.

Chapter 9 surveys various types of probabilistic proof systems, specifically covering *interactive proofs* and the complexity hierarchy **IP**, *zero-knowledge proof systems* and the complexity class **ZK**, and *probabilistically checkable proofs* and the complexity hierarchy **PCP**. In particular, Chapter 9 describes the the proofs for **IP**=**PSPACE**, **IP**=**ZK** (under the existence of one-way functions) and the original proof of the PCP Theorem, **NP**=**PCP**$(\texttt{log}, O(1))$, as well as discussing *constraint satisfaction problems* (CSPs) and Dinur's later CSP-based proof of the PCP Theorem.

The final chapter, entitled "Relaxing the Requirements," begins with a short section on approximation and inapproximability. After mentioning the PCP-based inapproximability of a couple **NP**-complete problems and describing the notions of a *polynomial-time approximation scheme* (PTAS) and property testing, the chapter delves deeper into the topic of average-case complexity, where it describes a number of related analogs for traditional complexity classes, e.g. **distNP**, **sampNP** and **tpcBPP**.

As mentioned previously, the appendix section of *Computational Complexity: A Conceptual Perspective* is *very* extensive. Compared to the primary chapters of the text, the appendix comprises a series of largely self-contained discussions covering various topics in mathematics and special topics in complexity. In addition to a survey of necessary mathematical background for the book, subjects covered in the appendices include a survey of the quest for lower bounds, a theoretical discussion of the fundamentals of modern cryptography, advanced topics in probabilistic complexity and randomization, and explicit constructions of error-correcting codes and expander graphs.

Computational Complexity: A Modern Approach

Arora and Barak's book contains 23 chapters divided into three parts. The first 250 pages form "Part One: Basic Complexity Classes," the next 100 pages form "Part Two: Lower Bounds for Concrete Computational Models," and the final 200 pages form "Part Three: Advanced Topics."

In addition, there is a short, 20-page appendix covering necessary mathematical background at the end of the textbook. As a result of this structure, and in contrast to Goldreich's book, the chapters of *Computational Complexity: A Modern Approach* tend to be more self-contained.[4] Therefore, it seems the most reasonable to view a summary of the current text in terms of the various *threads of thought* that bind disjoint chapters together, especially those that join introductory chapters from Part One with advanced chapters from Part Three.

The opening chapter of the book bears the (cheeky) title, "The computational model – and why it doesn't matter," and begins by formally defining Turing machines. Conceding that explicitly programming Turing machines is quite tedious, the chapter proceeds into a discussion of their expressive power (with respect to modern programming languages like Java or the C family), time-constructable functions, the deterministic time hierarchy, the Church-Turing thesis, the class **P** and Edmond's concept of "practical efficiency," i.e. that **P** encapsulates the notion of *efficient computation*.

Chapters 2 through 6 progress from the class **P** through the terrain of basic complexity classes. Chapter 2 includes a discussion of **NP**, **NP**-completeness, exponential-sized classes, Karp-reductions and Levin-reductions. (Cook-reductions appear in a Chapter 2 exercise.) There is a particularly interesting survey section on problems in **NP** that are not known to be **NP**-complete, such as factoring integers, unique games labeling and finding Nash equilibria. Chapter 3 covers diagonalization-based results, including Ladner's proof of the existence of "**NP**-intermediate" languages, and a discussion of oracle machines and relativization. Chapter 4 introduces space-constructable functions, the deterministic and non-deterministic space hierarchies, as well as the space complexity classes **L**, **NL** and **PSPACE**. Chapter 5 continues by introducing the Polynomial-time Hierarchy **PH**, alternating Turing machines (ATMs) and a discussion of **PH** with respect to oracles. Finally, Chapter 6 wraps up the theme with a discussion of Boolean circuits and the class **P/poly**, including the development of the notion of uniform vs. non-uniform computation and ending with a discussion of circuit classes **NC** and **AC** as well as **P**-completeness.

Chapters 7, 8, 9 and 11 complete the introductory material in the book, beginning with the introduction of probabilistic Turing machines and progressing through a survey of the PCP Theorem. Chapter 7 discusses the basics of randomized computation including the classes **RP**, **coRP**, **ZPP** and **BPP**. Additionally, the topics of randomized reductions and randomized logspace algorithms are covered here. Chapter 8 introduces interactive proofs and the corresponding complexity classes **IP**, **MIP** and the Arthur-Merlin hierarchy **AM**[k], as well as a proof of **IP**=**PSPACE**. Chapter 9 is devoted to a survey of cryptography, which includes a discussion of negligible functions, one-way functions, pseudorandomness, pseudorandom generators and zero-knowledge proofs. Chapter 11 introduces the PCP Theorem (the proof is deferred until later) and demonstrates hardness of approximation results for `MAX-3SAT` and other problems. Additionally, there is a brief, introductory discussion of the Hadamard code.

Chapter 10 is worthy of special attention as it surveys quantum computation, which is unique to Arora and Barak's book. Topics covered include a formal definition of a qubit, quantum superposition, the EPR paradox, the complexity class **BQP** (with proofs of **BPP**$\subseteq$**BQP** and **BQP**$\subseteq$**PSPACE**), as well as detailed expositions of three famous quantum algorithms: Grover's search algorithm, Simon's period-finding algorithm and Shor's integer factorization algorithm.

Similar to Chapter 10, Chapters 12 through 16 (the chapters of Part Two) present material that,

[4]This will be elaborated on further in the Opinion segment of this review.

as a whole, is not found in Goldreich's book. Specifically, they cover *concrete models of computation* and the complexity lower bounds that result. Chapter 12 surveys decision trees and decision tree complexity, including Yao's Min-Max Lemma and the Boolean function characterizations of *sensitivity* and *block sensitivity*. Chapter 13 introduces concepts in communication complexity and various techniques for proving communication complexity lower bounds: the fooling set method, the tiling method, the rank method and the discrepancy method. Chapter 14 discusses circuit lower bounds, including Håstad's Switching Lemma, Razborov's Method of Approximations, the complexity class $\mathbf{ACC}0$, monotone circuits and various barriers in circuit lower bounds[5]. Chapter 15 develops proof complexity and includes a discussion of propositional calculus, interpolation theorems, a proof of an exponential Resolution lower bound and further discussion of other such proof systems. Chapter 16 concludes the theme of concrete complexity by considering the restriction the basic operations of previous models, e.g. decision trees and Turing machines, to operations over a field or ring and topological techniques for proving lower bounds in such a setting.

Chapters 17 through 23 constitute Part Three, which focuses on advanced topics in complexity theory, and many of the chapters in this part directly build on concepts previously in the book. Chapter 17, building on the basic complexity classes, covers counting complexity, i.e. the class $\#\mathbf{P}$, including proofs of Valiant's Theorem (PERMANENT is $\#\mathbf{P}$-complete) and Toda's theorem ($\mathbf{PH} \subseteq \mathbf{P}^{\#SAT}$). Chapter 18 surveys Levin's theory of average-case complexity, introducing the notion of average-case reductions and the classes **distP** and **distNP**. Chapter 19, building on Chapter 18 and Chapter 9 (on cryptography), covers two related topics: hardness amplification and error-correcting codes. Topics in Chapter 19 include Yao's XOR Lemma, Impagliazzo's Hardcore Lemma, the Walsh-Hadamard, Reed-Solomon and Reed-Muller codes, and list decoding. Chapter 20, building on Chapter 7 (on randomized computing), discusses the notion of derandomization. Specific topics include derandomization from pseudorandom generators, the Nisan-Wigderson pseudorandom generator construction and the $\mathbf{BPP} \stackrel{?}{=} \mathbf{P}$ question. Chapter 21, building on concepts from at least Chapter 6, 7, 19 and 20, develops explicit constructions of various pseudorandom objects. Beginning with a discussion on random walks, eigenvalues and weakly random sources, the chapter proceeds through a construction of expander graphs and extractors, then develops a space-bounded pseudorandom generator. Chapter 22, entitled "Proofs of PCP theorems and the Fourier transform technique," picks up from Chapter 11 (on the PCP Theorem). Starting with a definition of CSPs, the chapter explicitly lays out Dinur's proof of the PCP Theorem, Raz's Parallel Repetition Theorem, Fourier analysis techniques for PCP proofs, the long code, an improved hardness of approximation result for `MAX-3SAT` and a similar result for `SET-COVER`. The chapter concludes with a discussion of the unique games conjecture and metric space embeddings. Part Three ends with Chapter 23, which builds on Chapter 14 (i.e. circuit lower bounds), by discussing the natural proof barrier to proving lower bounds.

3 Opinion

Let me begin with a relatively uninformative comment: Both books are excellent texts that thoroughly cover both the breadth and depth of computational complexity theory. "But *of course* that's

[5] A meta-comment here: Arora and Barak do an excellent job of drawing connections between different lower bound techniques in concrete complexity. For example, Chapter 14's primary focus is circuit lower bounds but includes a short diversion into circuit lower bound approaches using communication complexity.

the case," you say. "The authors of each are *giants* in theory of computing. I guessed as much with a glance at the cover of each book!" The interesting differences, then, come out of the *angle* from which each book approaches complexity theory. For instance, reading Goldreich's book, I get the unmistakable impression that I am on a step-by-step, start-to-finish journey through the landscape of complexity. Turning to Arora and Barak's book, however, I see a slightly wider array of subjects (*Quantum computation! Communication complexity! The unique games conjecture!*). There is a definite attempt in *A Modern Approach* to include very up-to-date material, while Goldreich focuses more on developing a contextual and historical foundation for each concept presented.

So, perhaps the largest difference in the books is in the style of presentation. Goldreich regularly pauses to offer up self-referential commentary or philosophical musings. There are numerous "teaching notes" embedded in each chapter where he opines on the best manner in which to present the material in a classroom setting. And in fact, it is impossible for a person to read *A Conceptual Perspective* from start to end without stopping periodically to reflect on the recurring question, "So what does this all *mean* anyway?" By contrast, *A Modern Approach* sacrifices just a touch of the story-telling quality in the interest of presenting a wider range of subject material in a timely manner. Arora and Barak's book has a little bit more of a *lemma, lemma, then theorem, and let's get to the point* flavor, which is perhaps more reminiscent of the style of the majority of modern, conference publications. This is not to say Goldreich's book, by any means, lacks rigor (of which there is an abundance!) or that Arora and Barak's book ignores the context or history of various complexity results (each chapter ends with a discussion of the history of the topic in question). Rather, each book tends toward its own style, with Goldreich baking the narrative directly into the text and Arora and Barak pushing through the material and regrouping afterward.

Those intending to use either text in a course on complexity theory should be particularly aware of some of the more technical differences between the two books. Two specific examples from each book can help highlight these trends. The first is the difference in terminology used for the search analogs of **P** and **NP**. In Goldreich's book, they are called **PF** for *polynomial find* and **PC** for *polynomial check* and are presented nearly in tandem with their corresponding classes, whereas in Arora and Barak's book, there is mention of **FP** for *function polynomial time* in a later chapter. A search-**NP** is omitted in the latter, but would presumably be **FNP** for the sake of consistency. Perhaps this is only a minor quibble, but one should take care to ensure that students using either book are not confused if they later encounter different names for the same concepts or classes. The second example concerns the choice of notation used when presenting ideas. For instance, in the chapters presenting the concept of interactive proofs, Goldreich describes the strategy f of each player in terms of $f(x, \gamma)$ for input x and partial transcript γ, while Arora and Barak use a round-by-round notation – $a_1 = f(x), a_2 = g(x, a_1), a_3 = f(x, a_1, a_2)$, etc. Again, if the underlying concept is clearly taught, there should be absolutely no issue here. However, given that there are a litany of such examples between the two books, it may be useful for the instructor of a complexity course to be aware of both methods of explaining the concepts, and then pick and choose which they think fits best.

One final, major difference concerns the choice of topics included in each book. Goldreich is very up front in the introduction to *A Conceptual Perspective* in stating that the book only covers "high-level" ideas, by which he means technical material that, in his words, is clearly connected to "significant conceptual content." As a result, his book omits any mention of *concrete models of computation*, while Arora and Barak have an entire segment of their book (Part Two) devoted to topics in concrete complexity. Similarly, as mentioned before, *A Modern Perspective* has a chapter

on quantum computation, while *A Conceptual Perspective* does not. On the other hand, this gives Goldreich more real estate to devote to more thorough explanations of the philosophy underlying the results, as well as to go slightly more in detail on special topics, such as the complexity foundations of cryptography.

In closing, if I were told to choose one book over the other, it would be an impossible decision. Both books have numerous, unique strengths and very few weaknesses. It really comes down to knowing which type of experience you are trying to find. However, theorists, researchers and instructors of any school of thought will find either book useful. I applaud all three authors for their outstanding contributions.

Review[6] of
Algorithmic Game Theory
Editors: Noam Nisan, Tim Roughgarden, Éva Tardos, Vijay V. Vazirani
Publisher: Cambridge University Press
Price: $45 (free, non-printable version online)
754 pages, 29 chapters
ISBN: 978-0-521-87282-9 hardback
Reviewed by: Dave Levin
http://www.cs.umd.edu/~dml
dml@cs.umd.edu

1 Introduction

Game theory is the standard framework for modeling, analyzing, and developing mechanisms to influence the behavior of self-interested parties. Initiated by John von Neumann and Oscar Morgenstern, game theory has since been applied to myriad settings, including politics, sociology, law, biology, and psychology, to name but a few.

One of the more recent such settings is the Internet. Once a collaborative experiment, the Internet is now comprised of self-interested parties who often require incentives to cooperate with one another. Researchers and system designers are applying game theoretic tools to increasingly many facets of the Internet. Search engines employ auctions to determine which advertisers' links to display; peer-to-peer systems, lacking a centralized coordinator, use economic incentives to motivate users to donate their resources to others; and inter-domain routing, upon which the Internet relies for global connectivity, is largely influenced by service providers' financial motivations. It is difficult to imagine how to maintain and improve the global properties of the Internet without understanding and addressing such economic concerns.

A relatively recent approach in the pursuit of modeling and influencing the behavior of selfish participants is to view game theory through the lens of theoretical computer science. Game theory is inherently inter-disciplinary, and its intersection with algorithms, aptly named *algorithmic game theory*, proves to be a natural one. *Algorithmic Game Theory* is the first book to capture the breadth of work comprising this new field. Forty-five authors, experts from many of algorithmic

[6]

game theory's various sub-areas, contributed to this impressive text. *Algorithmic Game Theory*'s 29 chapters aim to cover the classic results upon which the field is built, as well as new results that demonstrate theoretical computer science's rightful place in the theory of games. The price ($45) is more than reasonable for a textbook of this size (754 pages) and caliber, and the publisher provides a free (albeit non-printable) version online. I offer my review in a top-down manner.

2 What's in the book

Algorithmic Game Theory consists of four parts. They are organized logically, and are independent to the extent that one can easily jump between them (better still, the chapters themselves permit random access).

The crux of game theory is the analysis of the outcomes, or equilibria, of a game. Part I, "Computing in Games," covers the computational complexity of finding Nash equilibria, as well as the algorithms to compute equilibria. It is well known that mixed strategy Nash equilibria exist; this part addresses the question: how difficult are they to compute? This is one of the fundamental questions to arise from the intersection of computer science theory and game theory. The remainder of Part I covers algorithms to compute equilibria in various settings: graphical games, games where players must learn others' strategies, market equilibria, and cryptography.

Mechanism design can be considered the converse of game theory: given a desirable outcome, design a game that achieves it. Part II, "Algorithmic Mechanism Design," covers results in this area, both classic—such as the uniqueness of Vickrey-Clarke-Groves (VCG) auctions, Gibbard and Satterthwaite's impossibility results for voting with more than 2 parties, and the revelation principle—as well as new results from the past several years. Part II's opening chapter, aptly named "Introduction to Mechanism Design (for Computer Scientists)," is the best treatment of introductory auction theory that I have seen (then again, being a computer scientist, it was written for me). In particular, I find Nisan's choice of presenting the revelation principle relatively late in the chapter to be a good one.

While much of classic game theory aims at showing the existence of and means to an equilibrium, Part III, "Quantifying the Inefficiency of Equilibria," addresses the broad question: how bad might that equilibrium be? Much of the focus of Part III is on the problem of routing games, pioneered in recent literature by Tim Roughgarden and Éva Tardos, who open this section of the book with a pleasant introduction to what has become a quickly growing sub-area. Network formation games, load balancing, and resource allocation mechanisms round out Part III. Many of the results in this sub-area of algorithmic game theory make use of the potential function method, which receives focus in Chapter 19.

An appropriate title for the fourth and final part of *Algorithmic Game Theory* would have been "Applications"; it surveys various domains to which some of the previous chapters' results have been applied. The areas covered include peer-to-peer (P2P) systems, prediction markets, social networking, and sponsored search. Since the publication of this book, there have been multiple results that would perhaps better demonstrate the extent to which game theory and mechanism design can be applied in practical settings. That is to say, there are fewer "fundamental" results in this portion of the book, and future editions would in my opinion benefit from choosing new sets of applications as they continue to be explored. Part IV is the only one lacking a unifying introductory chapter. This is a shame; an important addition would have been a broad treatment of why "real world" applications cause us to reevaluate our assumptions, and what it means to

tailor various mechanisms to practical settings.

3 What's in a chapter

A distinct set of authors pen each chapter of *Algorithmic Game Theory*. Perhaps as a result, there is little continuity from one chapter to another; one would expect otherwise had each chapter been written by the same set of authors. Notation changes from one chapter to the next, and there is some material that appears multiple times throughout the book; the Lemke-Howson algorithm, for instance, is presented in Chapters 2 and 3. That said, it also allows for easy jumping to any chapter with the expectation that it will be relatively self-contained. Each chapter begins with a high-level introduction, and whenever material was retread, I found that it served its purpose of maintaining a solid narrative within the given chapter.

Each chapter distills results from several papers on a particular topic into one cohesive narrative. Chapter 14, "Distributed Algorithmic Mechanism Design," for instance, combines some of the main results on implementing VCG routing in BGP (the Border Gateway Protocol, which defines how the Internet routes among autonomous networks), sharing the cost of multicast, BGP route oscillations, and policy-compliant routing, in a way that reads quite well. The goal of *Algorithmic Game Theory* does not appear to be to cover any particular area exhaustively, but rather to present the reader with enough material to understand the problems, key insights, and main findings of the various topics. This, coupled with the ability to jump from one chapter to another, makes the book fitting for use in a course that surveys a broad range of algorithmic game theory, and for researchers who wish to get a feel for what particular sub-area they may be interested in pursuing more deeply. Extensive bibliographic notes at the end of the chapters provide nice pointers for further reading.

That said, no chapter lacks technical meat; the proofs are given when appropriate, both for classic results—like the revelation principle and the incentive compatibility of VCG—as well as new results. Some proofs, such as that of the PPAD-completeness of computing Nash equilibria, would not be appropriate to present in a given chapter, as they would require a significant amount of real estate when the reader could simply refer to the paper for the technical details. In this regard, Papadimitriou may have had one of the most difficult tasks: to distill the complexity results of Nash equilibria in a concise chapter. Yet he succeeds by outlining not just the structure and necessary components of the proofs, but the insights behind them and why this problem is fundamentally combinatoric.

Many chapters also include areas of ongoing work, and suggestions for future work. This is a welcome addition. Given that this is a new field, and that this book seems targeted for readers interested in pursuing research in algorithmic game theory, I believe *all* chapters should have included this.

All but 6 of the 29 chapters also contain exercises. Most sets of exercises consist of roughly 8 questions, and offer a nice mix of straightforward and in-depth questions.

4 Opinion

I recommend *Algorithmic Game Theory*. For researchers new to (or interested in getting into) the field, the ability to jump among chapters makes this a valuable tool for getting acquainted with the problems, methods, and main results quickly. The thorough bibliographic notes serve as

a nice stepping stone to a more rigorous literature review. For those already familiar with the area, the fact that each chapter distills multiple results into a nice overview makes for a good read. Some of the gems in this book—particularly Papadimitriou's treatment of the complexity of computing Nash equilibria, Nisan's wonderful introduction to mechanism design, and Roughgarden and Tardos' introduction to the inefficiency of equilibria—would make for intriguing lectures in a graduate course.

Along with my recommendation of this book comes my recommendation for other background literature. The book's introduction gets off to a bit of a shaky start, but finds its legs halfway through, with some wonderful examples of algorithms-meet-games. Some working knowledge of game theory, for instance from the first few chapters of Osborne and Rubinstein's *A Course in Game Theory*, would be well advised. The intended audience appears to be computer scientists; while *Algorithmic Game Theory* makes relatively minor assumptions on the reader's background in game theory, an advanced undergraduate or graduate level course in algorithms and combinatorics would serve the reader well.

The authors and publisher of *Algorithmic Game Theory* have made available a freely downloadable (but not printable) electronic version.[7] There seems to be a contradiction here: a book about selfish agents, and the authors are giving out their text for free? It is my opinion that this free offer—and the book as a whole—is made with the intent of furthering this new, rapidly growing field, and I applaud this decision.

Ultimately, I do not believe that this will prove to be the definitive text on the connection of algorithms and game theory. Such a tome will, I suspect, come from a smaller core of authors, offering greater continuity between chapters and, hopefully, a retrospective look at how to successfully apply these results to real world problems. But there is time yet for that work; now, *Algorithmic Game Theory* serves as a wonderful text that will introduce and entice its readers to a fascinating new field.

Review of[8] of
The P = NP Question and Godel's Lost Letter
by Richard J. Lipton
Springer, 2010
240 pages, Hardcover

Review by
William Gasarch (gasarch@cs.umd.edu)

1 Introduction

I write this review in the form of a series of fictional blog posts. If there is a comment by, say, Lance F, then that does not mean that someone named Lance F actually thinks that. It may mean that I think that Lance thinks that.

[7] As of the time of this writing, the online version and errata from the first printing are available at Tim Roughgarden's site: `http://theory.stanford.edu/~tim/`

[8]

Do not confuse this with my real blog.

INTRODUCING LIPTON REVIEW BLOG
September 16, 2010

This blog will be dedicated to reviewing Lipton's book **The** P = NP **Question and Godel's Lost Letter**. The book is based on his blog *Godel's Lost Letter and* P = NP. I wonder why he reversed the order going from blog to book?

BLOGS INTO BOOKS? WHY? AND WHAT ABOUT KINDLE?
September 17, 2010

I got a copy of Lipton's book in the mail today from Springer since I am the SIGACT NEWS book review editor.

Why make a blog into a book? Couldn't I just log on to read it? Not really. When I read a post on a blog I skim it and always mean to get back to it later for a careful read but tend not to. Having a book on your desk is calling to you to read it in a way that a blog cannot.

The book is not on Kindle. Would that make sense? Actually yes— I can imagine having a Kindle with me and not having my laptop with me. Especially since I have a Kindle but don't have a laptop.

Comments

Anon 1: On Amazon the book sells for $80.00 new but $120.00 used. What kind of sense is that?

Anon 2: Dear anon 1- this makes plenty of sense. I for one prefer to read a used book since its already broken in. And I am willing to pay more for the privilege.

Anon 3: Isn't $80.00 a lot for a book that you can get on the web anyway?

Jin-Yi C.: I think it is the single most interesting web blog I have seen on related topics. He has great insight and wit and beautiful ways to see things and explain them.

Richard Lipton: The book will be on Kindle soon. Perhaps by the time you read this.

FIRST IMPRESSIONS LAST
September 20, 2010

My first impressions are very good, though since I have already read many of his blog postings, its not clear these are my first impressions.

Lipton has been a professor since 1973 and hence has a long view of the field. He not only has many results but has seen many results and trends. His posts reflect a deep sense of history and where ideas come from. Where do ideas come from? People of course! And Lipton never loses sight of that. Most of his posts begin with a description of story of the people involved.

At the end of the posts he references people who made intelligent comments on that post. Gee, if I had known he would do that then I would have commented on his posts more often. Hopefully intelligently.

DOES HE REALLY THINK P = NP**?**
September 21, 2010

The book is in four parts (I) Introduction, (II) On the P = NP question, (III) On Integer Factoring, and (IV) On Mathematics. The P and NP chapter has 31 posts. His point (and his original motivation for the blog) can be summed up by the following manifesto:

MANIFESTO I: We should have an open mind about the possibility that P = NP.

There are those who disagree. I was one of them; however, in a series of posts he does much to support the manifesto. Here are some of his points.

1. We all thought that NSPACE(n) was not closed under complementation but Immerman and Szelepcsenyi proved us wrong. The post was called *A Proof We All Missed*. He makes the point that not only was the result a surprise, but the proof is easy to follow (though clever to come up with). How easy? He includes a sketch that suffices to explain it.
2. We all thought that Bounded Width Branching Programs were not that powerful. Barrington proved us wrong as explained in the post *Barrington Gets Simple*.
3. What if P = NP but the algorithm is impractical?
4. What if P $\neq$ NP but the proof just shows that SAT does not have $n^{O(\log \log n)}$ sized circuits? It is harder to make the case that this is interesting. Lipton also makes an analogy to the Riemann Hypothesis which, whether it is true or false, will tell us something interesting

Comments:

Lance F: I disagree.with the Manifesto. We should not have an open mind about the possibility that P = NP since clearly $P \neq NP$.

Anon 1: Lance, if its so clear then go ahead and prove it.

Anon 2: If I proved P = NP then I would title the paper *A new barrier to showing* P $\neq$ NP.

Bill G: I disagree about P = NP not being useful. I am sure that if P = NP then whatever technique was used would speed up lots of computations (perhaps not rigorously). There is a wonderful chapter in Lance Fortnow's upcoming book on Complexity Theory for the Layperson on the consequences of P = NP. They are Awesome! The notion of P $\neq$ NP not being enlightening I find more plausible and scary.

Emily Litella The papers that got into SODA are posted. They messed up some of the list! Some of the authors are not listed! Oh, they only list the contact author. Never mind.

Anon 4: Other results that may make us pause when we are so sure that P $\neq$ NP: IP = PSPACE = ZK and PH $\subseteq$ P^{PP}.

Anon 5: When NP was shown to be in PCP($1, \log n$) peoples reaction was *now we can show certain functions hard to approximate*. Why didn't they think *Gee, NP is easier than we thought*.

Dr. Lipton is right- we are too tied to the mindset. Fortunately, since P $\neq$ NP, I don't think its really a problem.

Anon 6: My PhD was on Oracles for Space Classes. While I was proving that proving things about space classes would be hard to prove, Immerman and Szelepcsenyi were proving things about space classes. Oh Well.

Luca T. People forget that at one time the conventional wisdom was that P $\neq$ BPP. And now the conventional wisdom is that P = BPP. Could the same happen to P and NP. NO!

I SHOULD POST ON THAT! OH.
September 22, 2010

Upon reading the post *Ramsey's Theorem and NP* my first thought was *what a great idea! I should post about it on complexity-blog!*. This is of course silly since it was already posted on Lipton's blog. Nevertheless, I will discuss it here, as it is my favorite post in the book (so far). Recall the following subcase of Ramsey's theorem:

For all k there exists an $n = f(k)$ such that, for all 2-colorings of the edges of K_n there is a monochromatic K_k. (It is known that $2^{k/2} \leq f(k) \leq 2^{2k}$.)

Let us turn this around.

For all n there exists an $k = g(n)$ such that, for all 2-colorings of the edges of K_n there is a monochromatic K_k. (It is known that $\frac{1}{2}\lg k \leq g(k) \leq 2\lg(k)$.)

Let G be a graph. We want to know if G has a clique of size m. KEY: If G does have a clique of size m then, if we randomly color the edges of G we will *always* find a monochromatic clique of size $g(m)$. We can use this to obtain the following randomized algorithm for clique

1. Input(G, m).
2. Randomly 2-color the edges of G.
3. By brute force check if there is a monochromatic clique of size $g(m)$.
4. If there is not one then you KNOW that G does not have a clique of size m.
5. If there is one then you do not know anything; however, if you repeat this many times your confidence in G having an m-clique may increase.

It is unlikely that anything can be proven about this algorithm. In fact, a graph with many cliques of size $g(m)$ will fool it (though perhaps we can get around this). But this shows the power of the blog– he can toss out ideas of interest without worrying about referees, editors, journals, or paywalls.

Comments:

Bill G: If you fix m then this looks like a randomized sub-exp algorithm for Clique-of-size-m. Sort of a fixed parameter tractability thing.

DOES HE REALLY THINK FACTORING IS IN P?

Lipton seems to believe the following:

MANIFESTO II: We should have an open mind about the possibility that factoring is in P.

Logically MANIFESTO I implies MANIFESTO II, though he seems to be more emphatic about MANIFESTO I.

There are three postings about factoring. They offer some very interesting ideas. One of them is that if factorial was easy to compute then factoring would be easy.

Comments:
Larry W: If factoring is easy then we will all switch to Elliptic Curve Crypto. That's good news for me since my book on the topic will sell better.

Lance F: If factoring is easy then we'll just all go back to private keys. This won't be a big deal with today's technology.

Anon 1: If factoring is easy then I will take my credit card off of amazon.com and then I can't buy this great book. Oh well.

Bill G: I think its more likely that we show factoring is hard and thus that P $\neq$ NP.

MATHEMATICAL EMBARRASSMENTS, DISEASES, AND SURPRISES

There are nine posts on Mathematics in the chapter called *On Mathematics*. There is more variety here then in the other chapters since mathematics is so large. Of particular interest are the three posts on mathematical embarrassments, diseases, and surprises.

A mathematical embarrassment is a math problem that is open but should really be closed. They just don't sound that hard. Perhaps when we solve them we will see that they aren't that hard. Here is an example from Terry Tao (from his blog which is also a book which I will also review).

The Linear Recurrence Problem: Given integers $c_1, \ldots, c_d$ and numbers $a_0, \ldots, a_{d-1}$ consider the recurrence

$$a_n = c_1 a_{n-1} + \cdots + c_d a_{n-d}.$$

Is there some k such that $a_k = 0$? It is not known if this is decidable.

I agree- this is embarrassing. This should be known. My intuition is that we just solve the recurrence and get good enough approximations (hmmm- that could be the problem) for the roots of the char equation then we can determine if the recurrence ever gets to 0.

A mathematical disease is a problem that obsesses the math community without much success. I give one example due to Ulam.

The Reconstruction Conjecture: Let G be a graph on $n \geq 3$ vertices. Given all of the vertex-deleted subsets of G, can you uniquely determine G. The good money says that you can; however, the conjecture is open.

A mathematical surprise is just what it sounds like- a result that surprised people. We have already mentioned above the closure of NSPACE(n) under complementation and Barrington's result

on Bounded Width Programs. The negative solution to Hilberts' tenth problem was also a surprise. This post also goes into what causes a surprise: connecting two fields, incorrect proofs, and asking a new problem can all cause surprises.

Comments:
Bill G: The following surprised me: Let

$$VC_{17} = \{G \mid G \text{ has a vertex cover of size 17 }\}.$$

I would think this would take roughly $O(n^{17})$ steps. I first learned that by the Graph Minor Theorem you can get it into $O(n^3)$. This algorithm is not practical for a variety of reasons. However, later it was down to $n + O(k^k)$ (which is practical). Its even better now. But this all surprised me. This gave birth to the field of Fixed Parameter Tractability.

Anon 1: I am not surprised that this book is so awesome!

Bill G: So, who should read this book? If you do better reading books than blogs, and you know some (not even that much) theoretical computer science then you should buy this book. Certainly get it for your school's library.

Review of[9] of
The Pea and the Sun: A Mathematical Paradox
by Leonard Wapner
Published by A.K.Peters, 2005
218 pages, Hardcover, $22.00 new, $13.00 used (on Amazon)

Review by
William Gasarch gasarch@cs.umd.edu

1 Introduction

Bill: The Banach-Tarski Theorem states that you can take a sphere of volume 1, break it into a finite number of pieces, then put them together again, and have a sphere of volume 2.

Grad Student: Math is broken!

High School Student: That's just wrong man!
Bill: Would it help if I told you that

- The proof uses the Axiom of Choice.
- The proof is non-constructive. Hence it will not give you a method for carrying out this construction.
- At least one of the pieces that you obtain is not measurable. This means that you cannot assign a volume to it in an intelligent fashion.

[9] ©2010, William Gasarch

Grad Student and High School Student: No.

What Bill refers to as *The Banach-Tarski Theorem* is more often called *The Banach Tarski Paradox*. The title of the book comes from starting the procedure with a pea and doing it many times until you get a sphere the size of the sun. The book is about both the history leading up to the paradox, the paradox itself, and the reaction to it. The book does include enough math so that if you do not know the proof of the paradox ahead of time (I did not) then you can understand it. Note that you may understand the proof (I did) without liking the result (I didn't) or accepting it (I don't).

2 Summary of Contents

Chapter 1 is titled **History: A Cast of Characters**. This chapter has subchapters on Cantor, Banach, Tarski, Godel, and Cohen. There is some personal material but the book mostly concentrates on the history of the mathematics they did. Zermelo and Frankl also make an appearance, as does Hausdorff. This chapter also contains some easy math- mostly material on countability, unaccountability, and the axioms of set theory. The most interesting part of this chapter is the description of Hausdorff's theorem (later known as Hausdorff's paradox): The surface of a ball minus a relatively small number of points can be partitioned into three disjoint sets A, B, C such that $A, B, C, B \cup C$ are all congruent. Results like this, and Vitali's theorem (there exists a non-measurable set) serve to remind us that results like the Banach-Tarski Paradox do not come all-of-a-sudden. They build on prior results.

Chapter 2 is titled **Jigsaw Fallacies and Other Curiosities**. This chapter talks about other odd things in mathematics, including Simpsons Paradox in statistics. These oddities are not directly related to the Banach-Tarski Paradox; however, they serve to remind us that mathematics has other odd things in it. Some can be resolved, but some not.

Chapter 3 is titled **Preliminaries**. This begins to get to the heart of the matter. They talk about set theory, isometries (mappings that preserve distance), scissor congruences (two polygons are congruent if you can take them apart with straight line cuts and rearrange one into the other). They prove that any two polygons of the same area are scissor-congruent. The same statement problem for polyhedra is false (This was Hilbert's third problem.) They also talk about equidecomposablility which is a more careful way of looking at scissor congruences. The most fun part of this chapter is the lyrics to *Hotel Infinity* which is about Hilbert's Hotel. It is to the tune of *Hotel California*. (I hope someone releases it on You-Tube soon!)

Chapter 4 is titled **Baby BT's**. By this they mean mini-versions of the Banach-Tarski Paradox. This chapter has results that are startling, but not as startling as the full blown Banach-Tarski Paradox. One is Vitali's result that there are non-measurable sets. Another is the Sierpinski-Mazurkiewicz Paradox- there exists a set E of the plane that can be partitioned into $E_1 \cup E_2$ such that E, E_1, E_2 are all congruent. This proof does not use the Axiom of Choice.

Chapter 5 is titled **Statement and Proof of the Theorem**. I read Chapters 1,2,3,4 on a 4 hour train ride and understood most of it (I also knew about 1/3 of it ahead of time). Chapter 5 was harder but would have been much harder had I not read Chapters 1,2,3,4. Chapter 5 is understandable and does give the full correct proof. Understanding the proof of the Banach-Tarski Paradox does not make me like the result any better. A layperson may well understand most of Chapters 1,2,3,4 but not 5. That layperson will still have learned a great deal.

Chapter 6 is titled **Resolution**. No they do not resolve the paradox. They do suggest various ways people have understood the paradox. Some toss out the Axiom of Choice, some accept it at face value ("So math is weird. We knew that.") Some try to reinterpret. Nobody in this chapter claims that the result has any interpretation in the real real world.

Chapter 7 is titled **The Real World**. In this chapter there is an attempt to link the paradox to the real world. In particle physics there are cases where a collision creates new paradox. Could this be BT in the real world? There are also attempts to link BT it to Chaos theory. These attempts seem to be done by serious people. Even so, I do not take them seriously.

Chapter 8 is titled **Yesterday, Today, and Tomorrow**. While it is a good read, it could be in a different book. It is about the state of math today and some speculation about its future.

3 Opinion

This was a great book for me. I never liked the Banach-Tarski Paradox but I had never really understood the proof; hence, I was not qualified to have a public opinion. Now I am. I still don't like it.

I recommend the book for anyone who is interested in the paradox and wants to know the history and context behind it. I would recommend reading the entire book (the first four chapters are a quick read) and not jumping to the proof.

I have one criticism which is more of a recommendation for the next version. The Axiom of Determinacy has been suggested as a replacement for the Axiom of Choice. One reason is that it implies that all sets of reals are measurable. This should be mentioned.

Review of[10] of
Coding for Data and Computer Communications
by David Salomon
Springer, 2005
548, HARDCOVER

Review by
David A. Werden dwerden@ieee.org
2115 Shadebrush Court, Augusta, GA 30906

1 Introduction

David Salomon, a professor at California State University Northridge, produced his tenth book as an attempt to explain the three factors that affect the storage and transmission of data: Time (source coding), Reliability (channel coding), and Security). Specifically, this book focuses on the different coding methods used to address these three factors. For the casual reader or new computer science student, this book is NOT about programming source code. The most appropriate target audience for this book is anyone involved in the computer science field, with the caveat that the reader should have some understanding of topics such as finite state machines and binary trees.

[10]

2 Summary

To summarize this book in non-technical terms: this book is very Intensive! The amount of material that Salomon covers is enormous, ranging many specific topics dealing with coding. However, Salomon does an excellent job in that he stays on topic without delving deep into underlying principles.

This book is logically organized into three main sections: Part I Channel Coding (Chapters 1 and 2), Part II: Source Code (Chapters 3 5), and Part III: Secure Codes (Chapters 6 15). In addition to the primary sections of the book, there is a fourth section: Part IV: Essential Resources (four appendices). In addition to having its logical structuring, the book uses more than an adequate number of diagrams, figures, and MatLab code in order to present the reader with strong visual depictions of the related topic.

Part I focus on channel coding. In these two chapters, Salomon discusses the coding methods used for both error control (chapter 1) and error detection (chapter 2). The discussions in this section move relatively quickly as Salomon purposely avoids (throughout the entire text) the use of in-depth mathematical explanations. While these two chapters discuss multiple coding methods, the focus is primarily placed on Hamming codes (including a little mini-biography of Richard Hamming). Any student of computer science should find these two chapters interesting, either as a review, or an introduction to a topic that can be found throughout many disciplines of computer science.

Part II discusses source codes in the form of statistical methods (chapter 3), dictionary methods (chapter 4), and image compression (chapter 5). Source codes deal with the compression of data by the reducing of the number of bits of a data file. Source coding is also referred to as data compression and is most commonly recognized as the algorithms used in graphical data files. This section does tend to move a little deeper into theory, but not so much as to confuse a reader from the targeted audience. Salomon makes good use of trees in this section to explain methods such as the Huffman method and the LZ78 Dictionary Tree.

The quality of an algorithm is equally important as its cost of use. Salomon appears to subscribe to this theory in that he not only explains these methods, but also includes methods used for error metrics in Part II.

Part III is, by far, the most intensive section of the text. This section is comprised of ten chapters that are devoted entirely to secure coding. This section covers in-depth the methods used in securing coding. As with any discussion of secure coding, Salomon begins this section with the Caesar Cipher, a simple rotated alphabet cipher. Moving from Caesar Ciphers to the use of Steganography for data hiding, Salomon takes the time to explain a number of different secure coding methods. His discussions in this section should be of particular interest to anyone interested or involved in information security.

In all three of these parts of the text, Salomon makes an effort to present both the pros and the cons of each method. This allows the reader to have a better understanding of how to choose a particular method to fit a coding need. Additionally, this balanced explanation of methods tends to show that Salomon, who is a known veteran in the computer science arena, does know his subject matter and has made an attempt to pass this wealth of knowledge on to anyone who may be inclined to read this text.

Throughout the first three sections of this book, Salomon presents the reader with many exercise questions that are designed to encourage the reader to not only understand Salomons explanations,

but to go further and do more research on a topic. It is in Part IV that Salomon provides the answers to these exercise questions, relieving a frustration not readily available in a large number of computer science textbooks. Part IV also contains some other vital information in its appendices. Appendix A is a four-page discussion of symmetry groups that is presented well to the reader. Appendix B is a review section, or introduction for some, to finite fields. Finite fields are used in conjunction with a number of methods discussed throughout the book, so this review appendix proved helpful while reading more than one section. Appendix C is a short but accurate discussion on Cyclic Redundancy codes and Appendix D provides, in the form of projects, a little more challenge than the exercise questions found in the text. The projects themselves range from short tasks to more detailed tasks that will require the reader to look further and serve the purpose of reinforcing the methods discussed in the text, how they work, and, most importantly, how to use them these methods.

3 Opinion

It is my opinion that any individual involved with information theory, information systems, or information security will benefit from a study in this book. Salomon does an excellent job in covering the methods used to solve data storage and communication problems, and does so with an understanding that not every method can fit every situation.

The structure of this text, as well as the exercise questions throughout the chapters and the suggested project questions in the appendix easily allow for this book to be utilized as a textbook for a third or fourth year undergraduate student. However, even a seasoned veteran of computer science should find helpful and useful information about coding while reading this text.

Review[11] of
Binary Quadratic Forms: An Algorithmic Approach
by Johannes Buchmann and Ulrich Vollmer
Springer 2007, 318 pages, Hardcover, $87 on Amazon new, $27 used
Author of Review: Graham Coleman, graham.coleman@upf.edu

1 Overview

This book deals exclusively with Binary Quadratic Forms (BQFs), which are quadratic expressions with integer variables X and Y, as in $aX^2 + bXY + cY^2$, where integer coefficients a, b, and c determine the form. When the form appears in an equation, as in $f(a, b, c) = n$, it is a subtype of Diophantine equation, which are polynomial equations with only integer solutions allowed.

In general Diophantine equations are HARD- not decidable. In fact, the problem of, *given a polynomial in many variables does it have an integer solution?* is undecidable (this is Hilbert's tenth problem). On the other hand, the problem *given $n \in N$, does $x^n + y^n = z^n$ have a solution?* is easy, though the proof, which is Wiles proof of Fermat's last theorem, is difficult.

Fortunately, certain subclasses of these equations are both decidable and solvable. Such is the case with BQFs; algorithms are presented herein that classify, solve, and transform (into other

[11]

forms) these forms. Not all of the computations are necessarily easy; many of the algorithms presented run in exponential time.

Why would you be interested in this stuff? Maybe you are a number theorist. In this case, the material here is probably essential. Or maybe you are a cryptographer. Quadratic forms can be used as the basis for both public key and secret key cryptographic systems (described in Ch. 12). Or maybe you are a geometer. Lattices (introduced in Ch. 1 and developed in Ch. 4), a kind of discrete vector field developed out of BQFs, are a tool for solving some discrete geometry problems, such as optimal packing problems.

2 Summary

The first chapter defines the fundamentals. When we have a certain form: $aX^2 + bXY + cY^2$, we want to know if we can get it equal to a certain n by choosing X and Y. That's called the *representation* problem–seeing if there are solutions (X, Y) that *represent* n and enumerating them. Another question given a form (a, b, c) without the n: what is the minimum magnitude value of the form when X and Y aren't both zero? This is aptly called the *minimum* problem.

Next the *discriminant* is defined: $\Delta = b^2 - 4ac$. It's the key to a nice classification theorem. If Δ is negative, you can effectively bound (x, y) to a finite area and exhaustively try all solutions. If Δ is a perfect square, you can factor the form into two linear forms. If neither of these, then there are techniques that can eliminate the possibility of a solution. For example, showing that there can be no solution modulus some prime (in the case of $x^2 - 5y^2 = 2$) can be an easy way to do this sometimes.

In the last part of the chapter, we introduce three applications: the aforementioned lattice packings, factoring so called *ambiguous* forms (of the type $f = (a, ka, c)$ with integers a, k, and c), and how to use BQFs to make good rational approximations $x/y \approx r$.

Chapter 2 shows us how to transform forms, and how to know when forms are equivalent. This chapter deals heavily with the matrix form of the BQF $\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}$ as well as right and left group actions, as defined in the appendix. Chapter 3 gives us algorithms for simplifying forms to *primitive* representations (a, b, c have no common factor) and introduces the Legendre and Kronecker symbols which describe how to compute the number of solutions for a given (Δ, a) pair. Chapter 4 is more geometry based. It develops a link between forms and lattices in the complex plane, so there are more pictures!

Sometimes we want to *reduce* forms, for example, to simplify computation of the minimum, or to easily test equivalence by reducing both forms. Chapter 5 shows us how to reduce *positive definite* forms (a subtype of the "easy" forms with negative Δ) and Chapter 6 shows us how to reduce *indefinite* forms (harder subtype, positive Δ).

Chapter 7 and 8 define the product of lattices and develops the group algebraic structure induced by the product. Chapter 9 connects all of this group theory with exponential-time (in Δ) algorithms for finding roots, orders, discrete logarithms, and group structure for a subgroup. Chapter 10 introduces an algorithm inspired by the work of Shanks and Terr, which helps solve the equivalence problem. Chapter 11 describe probabilistic algorithms that are subexponential (faster than the algorithms in Ch. 9).

Chapter 12 shows the application of BQF theory in cryptography systems. 4 algorithms based on imaginary quadratic (IQ) field cryptography are given for a number of applications: generating

a secret key, IQ-Diffie Hellman; a public-key encryption scheme, IQ-IES; and two authentication schemes, IQ-DSA and IQ-Guillou-Quisquater scheme. The authors thus claim that imaginary quadratic cryptography is ready for practical use, but that real quadratic cryptography is mostly too inefficient to be used in practice. For real quadratic orders, only RQ-Diffie Hellman is presented. If you are a cryptographer, you were probably waiting for this chapter.

3 Opinion

This is an advanced monograph suitable for professional mathematicians or highly motivated students. The writing is clear, if terse and full of proofs at times, and exercises are given for each chapter. The text is organized well. Algorithms are delineated pretty clearly and marked up in pseudocode. Supplemental material such as group theory, lattice, and additional matrix definitions are available in the Appendix and referenced in the main text.

Review[12] of
Elliptic Curves
by Lawrence C. Washington
2008, Hardcover, Chapman and Hall
$65.00 new, $48.00 used on Amazon

Reviewer: David Chen

1 Introduction

In recent decades, the study and use of elliptic curves have risen to prominence in the fields of both pure mathematics and in computer science. They feature in some of the most exciting and open problems and results of our day. As some readers may recall, elliptic curves feature prominently in Andrew Wiles' 1994 proof of Fermat's Last Theorem. The Birch and Swinnerton-Dyer conjecture, one of the "Millennium Problems," is a conjecture about objects defined by elliptic curves. And of course, elliptic curve cryptography has gained a lot of attention recently as an efficient implementation of public key cryptography. Needless to say, they constitute a major area of current research.

In his book, Lawrence Washington gives a broad and elementary treatment of elliptic curves. His book is broad in the sense that he writes for many interests. There are substantial portions of the book dedicated to the development of the theory for beginners, to elliptic curve cryptography for the computer scientists, and to their relation to number theory and curves over complex fields for the mathematically inclined. In addition, his book is elementary in that he assumes relatively little knowledge of more advanced mathematics: some knowledge of modern algebra is all. The level and style of the text is accessible to an advanced undergraduate, and can easily be used as a textbook.

[12]

2 Summary

Washington's book is separated into three main sections: one section for building the basics of the theory, one section for discussing cryptography based on elliptic curves, and one section about the use of elliptic curves in number theory. There are illuminating examples scattered throughout the text, and full sets of exercises after each chapter.

2.1 Background

In the first four chapters of the book, Washington covers the basic definitions, tools, and results that will be used to discuss elliptic curves in later chapters.

Instead of starting off with definitions, the author poses the following problem:

- *Suppose a collection of cannonballs is piled in a square pyramid with one ball on the top layer, four on the second layer, nine on the third layer, etc. If the pile collapses, is it possible to rearrange the balls into a square array?*

This problem is subsequently solved by reducing the conditions to an elliptic curve, and with some algebra, a set of solutions is found. The ensuing Chapter Two is entitled "Basic Theory," and lives up to its name. Here, Washington covers the basics of elliptic curves. He gives us the Weierstrass equation (the equation describing the types of elliptic curves found in the text), and gives the group law over an elliptic curve. Not surprisingly, the "addition" operation of this group is exactly the set of operations used in the previous chapter to solve the cannonball problem. The rest of Chapter Two is a collection of topics and special cases that flesh out the basics of elliptic curves: other equations that describe them, change in coordinate system, and other special cases that might be interesting. There is probably more information here than necessary for most readers, but they are there for completion, and an instructor can easily pick out the subset needed for their course.

The following two chapters continue in building the theory of elliptic curves. In Chapter Three, we study torsion points on elliptic curves, and are introduced to the Weil and Tate-Lichtenbaum pairings that will be used in later chapters. Chapter Four gives a treatment of elliptic curves over finite fields, which will be used later on in applications to cryptography.

2.2 Elliptic Curve Cryptography

Chapters 5-7 cover the application of this theory to cryptography and other procedures. Chapter five, entitled "The Discrete Logarithm Problem," discusses general algorithmic attacks on the discrete logarithm problem (recall that because the points of an elliptic curve constitute an Abelian group, the discrete logarithm problem can be used as a basis for elliptic curve cryptography). The set of such algorithms contain the index calculus, Pollard's ρ and λ methods, and the MOV attack, amongst others.

Chapter six tells the reader how to build cryptosystems using elliptic curves. As hinted by the previous chapter, most of these cryptosystems are based on the hardness of the discrete logarithm problem for elliptic curves. In addition, there is a treatment of digital signatures using elliptic curves, and the description of another cryptosystem based on Weil pairing.

In chapter seven, Washington covers other computational problems that utilize elliptic curves. In particular, he gives relatively efficient algorithms for integer factorization and primality testing in this chapter.

Finally, in chapters 11 and 13, we are given a more detailed treatment of the Weil and Tate-Lichtenbaum pairings mentioned earlier, along with hyperelliptic curves.

2.3 Number Theory

In the later part of the book, Washington turns toward the mathematical audience and focuses more on the study of elliptic curves within a number theory context. In these chapters, we study elliptic curves over over $\mathbb{Q}$ and $\mathbb{C}$. The book goes on to describe complex multiplication, and finishes with respective chapters on zeta functions and Fermat's Last Theorem.

3 Opinion and Comments

As someone with no previous experience with algebraic geometry, I anticipated reading this book with trepidation. I was pleasantly surprised when I finally did read the book. The portion of the book that leads up to and builds elliptic curve cryptosystems are surprisingly accessible, and for this, the author must be given his credit. The techniques used were easy to follow, and Washington gives more than ample examples that both motivate and illuminate his points. The style was definitely easy enough to follow without trouble.

I feel that the level of the book was suitable for both undergraduates and graduate students. As advertised, it assumes a level of algebra knowledge from a first year course, yet still gives a complete treatment of the theory of elliptic curves. In addition, the book itself gives a very nice outline of the progression of chapters recommended for each "track" of study; an instructor could easily further refine these recommendations with the goals of the course in mind. For example, one may want to skip the many special cases given in chapter two, and focus more on the Weierstrass equations that will be more often used. The book has a feeling of being very complete, and would probably serve very well as a reference or companion reading as well. It was a joy for me to read, and I recommend the text to anyone wanting to learn about elliptic curves.

Review of[13] of
Concurrent Zero-Knowledge
by Alon Rosen
Springer-Verlag, 2006
184 pages, HARDCOVER

Review by
Sarvagya Upadhyay supadhya@cs.uwaterloo.ca
200 University Avenue West, University of Waterloo, Waterloo ON N2L 3G1, Canada

[13]

1 Introduction

For nearly 25 years, zero-knowledge proof systems have been a fascinating area to work in from both complexity-theoretic and cryptographic viewpoint. Roughly speaking, zero-knowledge proofs are proofs that convinces a skeptical verifier about the validity of the statement without yielding any other information about the proof. The zero-knowledge property states that the verifier obtains nothing from the proof and is typically formulated by saying that whatever information the verifier obtains from the proof can also be obtained from the statement itself (assuming the statement is valid).

A fundamental question regarding such proofs is whether the property of zero-knowledge is preserved under the various notions of protocol composition. While researchers have investigated the question of sequential and parallel composition of protocols, another notion of protocol composition is the concurrent execution of protocols. It turns out that when the protocols are executed concurrently, building zero-knowledge proofs is much more involved. This book, authored by Alon Rosen, is concerned the study of concurrent zero-knowledge proof systems with *black-box* simulators. These are proofs, where roughly speaking, the zero-knowledge property is preserved under the concurrent execution of protocols. The book presents two main results on the concurrent zero-knowledge proofs:

- For any super constant function $f : \mathbb{N} \rightarrow \mathbb{N}$, there exists an $O(f(n).\log n)$-round black-box concurrent zero-knowledge protocol for NP. This holds under the assumption that there exist statistically hiding commitment schemes.
- If there exists an $o(\frac{\log n}{\log \log n})$-round black-box concurrent zero-knowledge protocol for a language L, then L must be in BPP. This implies that one cannot hope to establish a constant round concurrent zero-knowledge proofs for NP using "black -box simulators".

The two results almost settles the question of round complexity of concurrent zero-knowledge proof systems with black-box simulators. While the issue of round complexity in black-box model is almost settled, Barak [1] showed a constant round zero knowledge argument system for NP that is zero-knowledge under concurrent composition of protocol. In fact, Barak et al![2] also showed that the round complexity with black-box simulators cannot be a constant even if one relaxes the condition of simulator running in *strict* probabilistic polynomial time to *expected* probabilistic polynomial time.

2 Summary

The book contains nine chapters. The first and the last chapters, due to Oded Goldreich, give a brief account of zero-knowledge proof systems and the developments in the past 25 years. The rest of the chapters, written by the author himself, is focused on concurrent zero-knowledge.

Chapter 1: The first chapter focuses on definitional issues pertaining to zero-knowledge proof systems and arguments. The chapter itself is divides into four subsections. The first section is intended to define the preliminaries and starts with defining interactive proof systems and arguments, one-way functions, and lays the groundwork for zero-knowledge proofs. The second section introduces the simulation paradigm and defines various notions of zero-knowledge depending upon the

type of simulation (basic, auxiliary-input, black-box simulation, etc.), the type of verifier (honest or dishonest), the "similarity" of the two probability ensembles (perfect, statistical, and computational indistinguishability), and the notion of strict versus expected probabilistic polynomial time verification procedure or simulation. The third section describes a computational zero-knowledge protocol for NP. The chapter concludes with the notion of composition in zero-knowledge proofs. It describes three types of composition operations on zero-knowledge proofs: sequential, parallel, and concurrent. The last composition is the focus of the book. The chapter is an excellent and almost non-technical survey on zero-knowledge.

Chapter 2: The second chapter pertains to concurrent-zero knowledge proof systems and gives a gentle introduction to the topic. The chapter is divided in to three main sections. The first section starts with a brief motivation for concurrent composition of zero-knowledge protocols, followed by two most relevant questions in the area: the feasibility of concurrent composition of zero-knowledge protocols and the round complexity of such protocols in terms of the number of concurrent executions. The second section begins with illustrating the difference between protocol repetition and protocol composition followed by use of repetition to reduce the error probability of the protocol while preserving the zero-knowledge property. The section ends with the notion of composition of zero-knowledge protocols. The final section of this chapter mentions the problem of scheduling while analyzing concurrent zero-knowledge, introduces the Richardson-Killian protocol to illustrate how to avoid the problem, and briefly mentions a lower bound on the round complexity of such protocols under black-box simulation. This chapter serves as a gentle introduction to the topic and is easy to follow.

Chapter 3: The chapter introduces the notations and conventions used in the rest of the book. It includes a formal albeit brief treatment to interactive proofs, zero-knowledge, and witness distinguishability as well as a formal definition of concurrent zero-knowledge and black-box concurrent zero-knowledge. It also defines various commitment schemes used in the construction of concurrent zero-knowledge protocols and specifies the conventions used throughout the rest of the book.

Chapter 4: This chapter describes the concurrent zero-knowledge protocols due to Richardson-Kilian (RK) and Prabhakaran-Rosen-Sahai (PRS) for NP. The concurrent protocols require a constant round zero-knowledge protocol for NP. However these protocols require different properties to be satisfied by the underlying protocol. The author chooses Blum's protocol that satisfies the required properties and is also simple. The chapter starts with describing Blum's constant round protocol followed by the RK protocol and PRS protocol. The rest of the chapter is devoted to giving an adequate description of the simulation and the analysis of both the protocols in the concurrent setting. At first glance, the protocols look complicated but the chapter contains a good exposition of the protocols. The chapter also contains a brief overview of RK simulator, its main drawback, and the Kilian-Petrank (KP) simulator. The last section gives an overview of the correctness of the KP simulation that forms the basis of the next chapter. Although the author only gives a high level exposition of the ideas involved in the simulation, the last section requires substantial attention and is important in going through the next chapter.

Chapter 5: This chapter contains a thorough analysis of the PRS protocol introduced in the previous chapter. This is the first of the two main results discussed in the book. Roughly speaking,

the chapter is focused on showing that for any super constant function $f : \mathbb{N} \to \mathbb{N}$, there exists an $O(f(n).\log n)$-round black-box concurrent zero-knowledge protocol for NP. The first section introduces the main ideas of the simulation and describes the actual simulator, while the subsequent three sections are focused on three conditions that prove the correctness of the simulation. These three conditions are explained in section 4.5 of the previous chapter. The final section details the extension of the main result, for instance, the applicability to argument systems and resettable zero-knowledge. The hardest and most non-trivial part of the chapter is section 5.4, which shows that the simulator never gets "stuck" with high probability. The proof of this part of the simulation is quite sophisticated and one can easily get lost in the details. A good understanding of section 4.5 of the previous chapter is helpful for this section.

Chapter 6: While the previous chapters give an upper bound on the round complexity of concurrent zero-knowledge protocols for NP, this chapter shows a simple lower bound on the round complexity, at least in the case of black-box simulation. This chapter also serves as an introduction to the main lower bound that is presented in the next chapter. The main result of this chapter is to show that if there exists a 4-message concurrent zero-knowledge proof system for a language L such that the concurrent executions can be efficiently simulated by black-box simulation, then L is necessarily contained in BPP. The rest of the chapter is devoted to proving the result.

Chapter 7: This chapter contains the main lower bound on the round complexity. The main objective of the chapter is to show that if a language L has an $o(\frac{\log n}{\log \log n})$-round black-box concurrent zero-knowledge protocol, then L must be in BPP. This is the second of the two main results in the book and is considerably more complicated and involved than the earlier result. The chapter is divided in to three sections. The first section details the outline of the proof, while the subsequent sections contain the proof of the main result. The proof outline is very important in going through the next two sections. However, one can easily get lost in the details presented in the last section. The technical details presented in the last section seems too complicated for a reader unfamiliar with the topic.

Chapter 8: This chapter provides two possible future research directions in the area of concurrent zero-knowledge proof systems. One research direction is to avoid the lower bounds of the previous chapter by focusing on alternate models and/or look beyond the black-box simulation paradigm. In fact, Barak using non black-box simulators constructed a constant round zero-knowledge arguments for NP that remains zero-knowledge under concurrent composition of protocols. The argument system is based on the assumption of existence of collision-resistant hash functions.

Chapter 9: The last chapter focuses on other developments in the area of zero-knowledge proof systems. The chapter discusses non-interactive and statistical zero-knowledge proof systems and their complexity-theoretic properties such as complete problems for the classes corresponding to such proof systems and limitations on the expressive power of such proof systems. It also contains a brief account of zero-knowledge in multi-prover interactive proof system, which enables the existence of perfect zero-knowledge proofs for NP without any computational assumptions.

3 Opinion

This book is based on the PhD thesis of the author. The focus of the book is on a very special topic in zero-knowledge proof systems. As such it is not suitable for an undergraduate or a beginning graduate course. The book should be useful as a reference for those who would like to investigate the topic or wish to work in the area itself.

Given that the book contains two very complicated results, readers not familiar with the topic (like me) will find it hard to go through the details. Although the author provides a high level overview of the main technical results, I found Chapters 5 and 7 difficult to comprehend. The chapters being lengthy do not help the matter either. However, the author's overview of the main results is one positive aspect of the book. It certainly helped me in going through the technical details.

The author motivates the question of concurrent composition of (computational) zero-knowledge protocols very well. Chapter 2 is well written and forms a very good introduction to concurrent zero-knowledge. However, the author fails to give an adequate treatment to computational zero-knowledge proofs that might be useful to readers unfamiliar with the topic.

Overall, except for the volume of technical details in Chapters 5 and 7, the book is a good read. Being unfamiliar with the topic, I enjoyed reading the book. The first two chapters are well written and are aimed at readers with no prior expertise in the area. Chapters 4 and 6 are certainly helpful in getting a grip on Chapters 5 and 7, respectively.

I must add that the results in this book are very interesting. An interesting aspect of the result presented in Chapter 5 is that it shows an *interactive proof system* for NP that remains zero-knowledge under concurrent composition of protocol, as opposed to Barak's constant round zero-knowledge *argument system* for NP. Moreover, the lower bound in Chapter 7 suggests that one cannot hope for black-box techniques to give a significant savings in the round complexity. The importance of this result cannot be overstated and this book provides a very good treatment to the result.

References

[1] B. Barak. Barak, how to go beyond the black-box simulation barrier. In *Proceedings of the 43nd Annual IEEE Symposium on Foundations of Computer Science,* Vancouver, Canada, 2002.

[2] B. Barak, J. Lindell, and S. Vadhan. Lower bounds for non-black-box zero knowledge, 2004. See ECCC TR06-127, at eccc.hpi-web.de/eccc-reports/20064TR04-11/index.html. Earlier version in FOCS 2003.

Review of [14]
Introduction to Cryptography
Authors: Hans Delfs and Helmut Knebl

Publisher: Springer, 2007
ISBN: 978-3-540-49243-6

Reviewer: Cillian Murphy (cillian.k.murphy@gmail.com)

1 Overview

Introduction to Cryptography by Hans Delfs and Helmut Knebl is, as the title suggests, an introductory guide to the area of cryptography. The book is aimed primarily at an undergraduate audience, with some of the later sections covering more advanced material. Some mathematical experience is assumed as many of the key concepts are initially outlined, and later precisely defined, in mathematical terms.

The book is split into two main sections. The first section, comprising of chapters 1-4, discusses the main points of symmetric and asymmetric encryption, digital signatures, RSA, electronic elections, and other areas. These chapters focus on asymmetric cryptography, and do not deal with probability theory. The second section, chapters 5-10, introduces probability theory and uses this to precisely prove the security of cryptographic systems and describe the assumptions that underlie the safety of public-key cryptographic systems.

2 Summary of Contents

Chapter 1: Introduction. A gentle introduction to the subject of cryptography. This chapter contains an explanation of the goals of cryptography, the basics of encryption and decryption, and some cryptographic protocols. Basic attacks against cryptographic schemes are outlined and the concept of provable security is introduced.

Chapter 2: Symmetric-Key Encryption. Symmetric key encryption is described, and block and stream ciphers are introduced. The data encryption standard algorithm is described and its strengths and weaknesses are outlined. The Rijindael algorithm is then introduced, presented in high-level terms, and broken down into its constituent parts. The chapter ends by outlining block cipher modes of operation for dealing with messages over the permitted size.

Chapter 3: Public-Key Cryptography. The concept of public key cryptography is outlined briefly, and an overview of necessary modular arithmetic is provided. The RSA cryptographic system is described, with key generation, encryption, and decryption methods outlined. The concept of digital signatures is also introduced. This is followed by a discussion of the potential attacks against RSA and similar systems including the small message space attack, Bleichenbacher's 1 million chosen ciphertext attack, and the common modulus attack. Cryptographic hash functions are outlined with their security requirements, details of their construction, and how they may be used for message authentication. The chapter ends with discussions of ElGamel's encryption and signature scheme, and Rabin's encryption and signature scheme.

Chapter 4: Cryptographic Protocols. The key factors and uses of cryptographic protocols are outlined at the beginning of this chapter. Key exchange and entity authentication are introduced and the Kerberos authentication service and the Diffie-Hellman key agreement are described. The concepts of an identification scheme and zero knowledge are introduced, and the Fiat-Shamir identification and signature schemes are discussed. The chapter finishes with detailed descriptions of the key factors of electronic elections and digital cash transactions

Chapter 5: Probabilistic Algorithms. This chapter represents the beginning of the second part of the book – from here on the book is considerably more mathematical. The basics of probability theory are outlined, with descriptions of coin tossing algorithms and Monte Carlo algorithms.

Chapter 6: One Way Functions and the Basic Assumptions. This chapter builds upon some of the concepts outlined in earlier chapters, with the aim of providing more precise definitions.

Discrete exponential functions and uniform sampling algorithms are described in detail, along with discussions of modular powers and modular squaring. There is a section dealing with the formal definition of one-way functions, and the chapter closes with a discussion of hard core predicates. Compared to the first half of the book, this chapter requires more effort on behalf of the reader.

Chapter 7: Bit Security of One-Way Functions. The concept of bit security is outlined in the introduction to this chapter, and this is followed by an in depth discussion of the bit security of several algorithmic families. The aim of this chapter is to demonstrate the computation of an encoded message from the Exp, RSA, and Square families of algorithms when given the most, or least, secure bit of the message. The methods outlined are detailed well, and the exercises at the end of this chapter are especially helpful given the nature of the material.

Chapter 8: One-Way Functions and Pseudorandomness. The relation between cryptography and randomness is expanded in this chapter. It begins with a discussion of pseudorandom bit generators, including a description and proof of how random and pseudorandom sequences cannot be differentiated in certain circumstances. The chapter finishes by outlining Yao's theorem.

Chapter 9: Provably Secure Encryption. The notion of perfect secrecy underlies this chapter, with much reference to Shannon's work. This is one of the most interesting chapters in the book, as many of the main concepts previously discussed are brought together. The main topics covered are probabilistic attacks, ciphertext-only and chosen-ciphertext attacks, unconditional security, and the noisy channel model which ends the chapter.

Chapter 10: Provably Secure Digital Systems. The final chapter deals with the security of cryptographic schemes in real-world scenarios. Various attacks are outlined, and these are followed by descriptions of claw free pairs, collision resistant hash functions, and authentication-tree based signatures. Attacks on signatures schemes are described at the end of the chapter, with two case studies considering different attackers.

3 Opinion

Introduction to Cryptography is well written and outlines many of the key areas of modern cryptography, and I enjoyed it. The material flows well and is easy to follow, chapters are laid out logically, and the appendices summarize much of the mathematics required for understanding early descriptions of cryptographic systems. One of the book's advantages is that authors have included a set of exercises at the end of each chapter, and these enable the reader to practice the material covered. The answers to the exercises are available on the authors' website, which is helpful if the reader is not using the book as part of a course.

The authors mention in the introduction that they have omitted the elliptic curve cryptography in order to keep the level of mathematics easy. This is an understandable decision and, in my opinion, the book does not suffer greatly as a result. All of the main areas covered are outlined and described well, and easy to understand upon finishing the book. The reader will come away with a good introductory understanding of many of the key factors of cryptography, and it would make a good companion to an introductory course on the subject.

Review[15] of
Introduction to Modern Cryptography

[15]

Author: Jonathan Katz and Yehuda Lindell
Publisher: Chapman & Hall-CRC 2008
1-58488-551-3

Reviewer: Ben Fulton ben@benfulton.net

1 Overview

Cryptography is a subject that has been intriguing people for millennia. Show a group of ten-year-olds a simple substitution cipher and they will immediately understand the utility and practice of sending the messages. Follow up by demonstrating the mechanical tools available to create a simple shift cipher and the students will remember the techniques for life. And these methods, with greater or lesser sophistication, have been used to conceal military messages, business information, and other communications for almost as long as people have been able to write.

But prior to the 19th century, the making and breaking of codes was more an art than a science. Some codebreakers began to apply more systematic analysis to their codes in the 1800's, but there was still much disagreement as to exactly what comprised a *good* code over the next 150 years. In the computer age, though, more rigorous definitions of terms like *secret* and *secure* came to be applied, ushering in the age of modern cryptography, the subject of this book and which the authors define as *the scientific study of techniques for securing digital information, transactions, and distributed computations.* The book covers the principles underlying all modern security today, including https, digital certificates, and digital file signing.

2 Summary of Contents

The book is divided into three main sections: the first section introduces the topic and gives an overview of classical cryptography; Chapters three through six discuss private-key cryptography, and the remainder of the book is devoted to public-key schemes.

Chapter 1 discusses several historical ciphers in terms of modern analysis, along with methods of breaking them. Some basic syntax is defined, and the background of private-key encryption is defined and discussed. Chapter 1 also lays out three principles of rigor that separate modern cryptography from the classical form: exact definitions, precise assumptions, and rigorous proofs.

In Chapter 2, perfect secrecy is defined, and the one-time pad and Shannon's theorem are discussed. It is shown that perfect secrecy requires a key as long as the message it encodes. (Demonstrating that the key must also be used just once is left as an exercise.)

Chapter 3, at 59 pages the most extensive in the book, begins the discussion of private-key encryption. Since one-time pads for any given message are not practical, a definition for practical security is given, based on the probability of an attacker breaking the code in a given time, and several ramifications of the definition are discussed. Calculating this probability requires some background in computational complexity, so this is given here.

In Section 3.4, stream ciphers are introduced, along with the required pseudorandomness. Fixed-length, variable-length, and multiple encryptions are discussed, and the section concludes with by pointing out that no unquestionably secure stream cipher is known today. Sections 3.5 and 3.6 discuss chosen-plaintext attacks and securing schemes against them, and block ciphers along with modes of operation are introduced for the first time. Section 3.7 covers chosen-ciphertext attacks,

and discussion of some practical applications of chosen-plaintext and chosen-ciphertext attacks clarify the need for defending against them.

Chapter 4 is split into two parts. The first part is on message authentication codes; a formal model is given, and some peculiarities are discussed, including replay attacks and the inability of encryption to provide adequate authentication. Finally, a way of constructing codes based on pseudorandom functions is given, and carefully proved to be correct. The second part of Chapter 4 defines and discusses collision-resistant hash functions, such as MD5 and SHA-1, and then defines some authentication schemes that are based on them. The Chapter concludes with some schemes that are secure against chosen-ciphertext attacks, and discusses some of the implications of combining encryption and authentication together.

In Chapter 5, block ciphers are formally introduced. The basic principles behind DES and AES are explained, along with the properties that make them secure. The authors also demonstrate various attacks on DES and AES, and show how certain attacks can be successful if the algorithms are not fully implemented.

While Chapter 5 is a practical discussion of block ciphers and pseudorandom objects, Chapter 6 delves into the underlying theory. The authors show that all private-key cryptography is based on the existence of one-way functions, and construct some provably secure pseudorandom permutations. The difference between the practical and theoretical is brought to the fore, as the provably secure permutations turn out to be far less efficient and useful than the permutations discussed in the previous Chapter. Dealing as it does with some of the theoretical underpinnings of private-key cryptography, Chapter 6 serves as an appendix to this section of the book.

The theoretical vein continues in Chapter 7, which begins the section on public-key cryptography. The concept of what it means for a problem to be *hard* is rigorously defined and the examples of integer factorization and computing of discrete logarithms are given. The authors then show formally the existence of one-way functions and collision-resistant hash functions. This gives us the required proofs we need to show the difficulty of breaking appropriately created schemes. Chapter 8 demonstrates several methods for solving these hard problems in slightly faster fashion than a simple brute-force search.

Chapter 9 opens the discussion of public-key cryptography. The authors discuss some of the reasons why private keys are not always sufficient, and also comment on key distribution centers as one possible solution. But then the authors discuss the seminal paper of Diffie and Hellman in 1976 in detail, and formalize the Diffie-Hellman key-exchange protocol.

In Chapter 10, attacks on public-key encryption are covered. For example, the advantage an adversary has of being able to encrypt messages at will using a public key are covered (chosen-plaintext attack), and it is demonstrated that the scheme can still be secure. The authors then cover the case of using a public key to encrypt a private key, and sending that key along with a message. RSA and El Gamal encryption are covered. The authors show RSA encryption without padding, and warn against using it; and then show some padded RSA schemes. Finally, several examples of chosen-ciphertext attacks are given. Some optional material follows: a discussion of trap-door permutations ends Chapter 10, and Chapter 11 shows a few alternate public-key schemes, and proves them secure.

Chapter 12 introduces digital signatures. RSA and DSS signature algorithms are covered, along with the potential weaknesses of a simple RSA scheme, along with a one-time signature scheme. A section on certificate authorities and some strategies for invalidating certificates finish up the Chapter.

The final Chapter discusses the Random Oracle methodology, and the difference between formally proving a scheme's security and proving it through a random oracle. The authors believe that the Random Oracle methodology is significantly better than having no proof of security at all.

3 Opinion

Although the book is nicely balanced between formal mathematical proofs and discussions of conceptual principles, it does not lend itself to light reading. The authors do an elegant job of laying out each problem, giving an intuitive description of the problem, and formally proving the solution. With some study, a student without a detailed background in number theory should be able to grasp the general principles of cryptography from this book, while graduate students looking for a reference manual for formal proofs of security will be amply served by the theorems outlined.

A Note from the Authors

We wrote our book to address what we saw as a glaring deficiency among all other cryptography textbooks that were available at the time. Specifically, our aim was to write a textbook covering *modern* cryptography from a computer scientist's (rather than a mathematician's) perspective, at a level accessible to undergraduate students as well as practitioners. To the best of our knowledge, our book remains the only one on the market with this focus.

The book is not intended to be 'light reading'. It is, however, intended to be approachable, and to present the material in a self-contained way requiring minimal prerequisites. While it is impossible to know for sure whether we have succeeded in this regard, we are pleased to note that the book has been used successfully in undergraduate courses at our institutions as well as several other schools. We have been heartened to receive much positive feedback from instructors and their students, as well as from several people in government and industry who have used our book for self-study and reference.

Further information about the book, including a table of contents and the first chapter, is available at http://www.cs.umd.edu/~jkatz/imc.html

Review of[16] of
An Introduction to Mathematical Cryptography
by Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman
Springer-Verlag, 2008, 523 pages, Hardcover

Review by Sarah Meiklejohn smeiklej@cs.ucsd.edu

1 Introduction

The mathematics underlying public-key cryptography has been growing steadily more complex over the past thirty years. Historically, the math involved in cryptography (public-key cryptography in

[16]

particular) has been focused primarily on basic number theory and algebra, but with the introduction of lattice-based, elliptic-curve, and most recently pairing-based, cryptography, people in the field are dealing with complex mathematical objects that go far beyond the scope of an introductory algebra course. As a result, it has been quite difficult (in fact, I would say impossible) to find an explanation of all of these different areas of mathematics in the same book.

In this book, the authors finally provide us with a clear and approachable explanation of a wide variety of cryptographically relevant areas such as finite fields, polynomial rings, lattices, and elliptic curves. In addition, the authors assume no mathematical background (albeit a fairly high level of mathematical sophistication, especially in later chapters), yet still manage to provide rigorous proofs for almost all of the theorems in the book. Finally, adding to these already significant accomplishments, the authors manage to appeal to the more "computer science" side of things as well. For almost every algorithm they introduce, the authors provide pseudocode, runtime analysis, and multiple sample runs of the algorithm to make sure the reader really understands what is going on.

Because of its mathematical nature, the book focuses almost entirely on public-key cryptography, and even more specifically on encryption schemes and, at the end, digital signatures. There are brief sections at the end on other topics such as pseudorandomness, hash functions, quantum computing, etc.; these sections are not meant to be rigorous, however, but to serve as a jumping-off point for any further research into the topic.

2 Summary

From the basic Caesar cipher to Babai's algorithm for solving the approximate closest vector problem for lattices, the book really covers the gamut of cryptography. It assumes no background in algebra or number theory (although it does assume knowledge of some very basic facts from linear algebra) and is thus suitable for almost any reader (or at least any reader interested in the mathematics of cryptography!). This is reflected in the exercises; some of them begin by asking for a "2 to 5 page paper on one of the following topics," while others ask the reader to implement Pollard's ρ algorithm for solving the discrete log problem or suggest that they "might also try to prove that the Weil pairing is bilinear" (although they do mention that this last attempt is quite difficult).

The book is generally organized as follows: in each chapter, a particular problem (i.e., discrete log or factoring) or branch of cryptography is introduced, often accompanied by a fair amount of mathematical background. Constructions based on the hardness of the given problem are then introduced, followed by known algorithms for solving the problem. I'll also summarize each chapter individually below:

Chapter 1 provides an introduction to some of the basic mathematical equipment necessary for the rest of the book: modular arithmetic, groups, Fermat's Little Theorem, etc. It also provides a few brief sections on symmetric encryption; as mentioned before, the book is almost entirely devoted to public-key cryptography.

Chapter 2 introduces the discrete log problem. It first describes some cryptographic protocols based on the DLP, Diffie-Hellman key exchange and El-Gamal encryption, and then describes some of the best-known algorithms for solving the DLP. The chapter ends with some more mathematical background: most notably, polynomial rings and finite fields.

In Chapter 3 we see another computational problem, this time the hardness of factoring an integer of the form $N = pq$ for p and q prime. Again, the authors begin by describing protocols

based on the hardness of factoring (in this case RSA) before continuing on to describe factorization algorithms. In the latter category, the authors include Pollard's $p-1$ method, the linear algebra method using difference of squares, and finally both the quadratic and the number field sieves. For each method, the authors provide a fair amount of intuition behind how the algorithm works before going into the math behind it, as well as multiple examples and a runtime analysis. Because of its similarity to the sieving methods, the authors also describe in this chapter the index calculus method for solving the DLP. Finally, the authors introduce the notion of quadratic reciprocity and end the chapter with an outline of the Goldwasser-Micali cryptosystem.

Chapter 4 is mainly devoted to mathematical background and various cryptanalytic techniques. It provides definitions for basic combinatorics and probability. It also switches gears slightly to discuss cryptanalysis on the Vigenére cipher. To make use of the probability theory, the authors then discuss collision algorithms, a discussion which culminates in a detailed description (again, complete with proofs, analysis of both time and space complexity, and examples) of Pollard's ρ algorithm. The chapter then ends with a section on information theory and complexity.

In Chapter 5, the authors begin to venture into what is perhaps less familiar territory. This chapter is devoted to elliptic curve cryptography, which requires much of the necessary background behind elliptic curves; notably, however, the authors manage to pull this off without introducing or requiring any algebraic geometry (at the cost of having to skip some proofs). Once again, the authors do not limit themselves to one particular side of cryptography and instead show both constructions based on the security of problems on elliptic curves (i.e., Joux's tripartite Diffie-Hellman key exchange) and algorithms for breaking hard problems (i.e., Lenstra's factorization algorithm). The authors also develop the mathematics necessary for the Weil pairing (and pairings in general) and describe algorithms for efficiently computing these pairings, as well as their applications.

In Chapter 6, the authors switch from elliptic curve cryptography to lattice-based cryptography. This also requires a fair amount of mathematics, although again the authors do not assume any background (they even have a review of the properties of vector spaces). They then give problems that are considered to be hard for lattices, such as the standard shortest/closest vector problems, as well as the shortest basis problem and the approximate shortest/closest vector problems. As usual, the authors first describe cryptosystems based on the hardness of these problems (GGH and their own NTRU) as well as attacks, such as the LLL algorithm and its generalizations.

Chapter 7 is perhaps the only chapter in the book devoted solely to constructions. In this chapter, the authors motivate and describe digital signatures. They start by describing signatures in the standard setting (i.e., RSA and DSA) and then describe lattice-based signatures such as the GGH and NTRU signature schemes.

In Chapter 8, cryptographers working in computer science finally get to see many of the topics to which they are more accustomed: hash functions, pseudo-random generators, provable security, zero knowledge, etc. There are no proofs provided (and in many cases no constructions), however, and each section is quite short. Instead, the descriptions seem as though they are mainly there to pique the reader's interest in the topic (or, as stated by the authors, to possibly provide inspiration for a final presentation or research project in a course) and not actually provide a rigorous exposition.

3 Opinion

As is probably already clear, I highly recommend this book to anyone, mathematicians and computer scientists alike. As mentioned in the introduction, this book succeeds in the amazing task of

bringing together a diverse arrangement of mathematical topics, while still managing to be clear and rigorous in the treatment of each, which means that someone interested solely in the mathematics of cryptography will find it to be a good resource. In addition, a computer scientist interested in actually implementing any of these cryptosystems or algorithms will find clear pseudocode for every algorithm, runtime (and in certain cases space complexity) analysis, as well as multiple examples that demonstrate exactly how it should be run. What I did not mention in the introduction, but is perhaps of equal importance, is that the book is actually fun to read! The authors include many historical points of interest and often provide intuitive arguments (for example, why a certain problem should be considered 'hard') before getting into the mathematical justification; these interludes combined with their overall conversational style make it a really pleasant read.

Aside from being a wonderful introductory text, the book also serves as a good reference. Many textbooks on elliptic curves, for example, require a fair amount of background in topics such as algebraic geometry, which may make these books impossible for readers without a rigorous mathematical background to understand. This book, in contrast, provides definitions and theorems for elliptic curves and pairings that I actually found to be very accessible. In addition, as mentioned earlier, the authors do not go into much detail on the more traditionally "computer science" topics such as symmetric encryption, the random oracle model, or hash functions. The book, therefore, has a very different feel from other introductory cryptography textbooks (such as those of Goldreich or Katz and Lindell) and in fact serves as a nice complement.

Finally, in terms of being used in a course, I can say (from personal experience) that the book works quite well for courses designed to serve as introductions to mathematical thinking, in which the students learn how to write proofs while covering the earlier material (groups, rings, etc.) and then move on to the more advanced material of either elliptic curves or lattices, or both if there is sufficient time. If this seems too challenging, I could also imagine just covering the earlier chapters and then saving the later chapters for a more advanced 'topics' course. It is also worth mentioning that in the introduction the authors helpfully outline which sections could be left out (in the sense that if abstract algebra is a prerequisite for the course there is no need to cover groups and rings in detail, or if the course has a certain focus it might seem unnecessary to cover the sieving methods for factorization or the Pohlig-Hellman algorithm for discrete log) as well as various possible syllabi.

Review of[17]
Software Abstractions: Logic, Language and Analysis
Daniel Jackson
M.I.T. Press, 2006, 366 pages, Hardcover
Review by Andrew C. Lee alee18@syr.edu

1 Introduction

This book is about Alloy and its analyzer. Alloy is a modeling language which allows programmers to specify their designs and to build software models *incrementally* via the Alloy analyzer. Apart from the book website [1], the authors' research group maintains a website at [2]. The software and many of its related materials can be downloaded from the site. There is also an entire community

[17]

of users that interacts with each other regularly via online forums etc. hosted within the website. This review focuses on the book only.

So, what can Alloy do ? As explained by the author, Alloy is neither a model checker nor a theorem prover. It is a software tool that can check all instances of a prescribed small size automatically. During the design process, software developers may use it to help discover small instances (if any) that violate the desired properties being specified. In other words, it helps to expose flaws in a design by checking small examples. As "Most bugs have small counterexamples" (*small scope hypothesis*, page 141), the non-existence of small counterexamples may boost the developer's confidence regarding a software design.

2 Summary

Motivations The author describes two approaches in formal methods that were popular in the early 90s. They're:

1. The use of modeling checking tools (e.g. the SMV model checker): They are full automatic tools that can expose flaws in (hardware) design automatically.
2. The use of specification languages (e.g. the Z language): These languages are simple, elegant and can help capture the essence of many software abstractions.

In addition, there seems to be a lack of software tool support for formal specification. The wish to fill these gaps and to combine the benefits of the above two approaches (i.e. automatic checking capability; simple but expressive specification language) motivates the author. This leads to the development of Alloy.

The Author also compares and contrasts this approach with others briefly. They include extreme programming (design that evolves with code development) and theorem proving (a different approach used by the formal methods community).

Another motivation, stated in the preface, is to " ... capture the attention of software developers, who are mired in the tar pit of implementation technologies, and to bring them back to thinking deeply about underlying concepts".

Whirlwind Tour Using Alloy differs from writing a formal proof of correctness and to test it using a theorem prover (or using pencils and paper). The idea is to create software models incrementally. With the help of the Alloy analyzer, immediate visual feedbacks are provided to support this *continual* review process. The short tour presented in Chapter 2 focuses on the development of an email client's address book. It probably gives a reader the first hand experience on what this development style is like.

Core Content The core elements behind Alloy are addressed in Ch.3 - Ch.5.

1. Chapter 3 Logic: Alloy makes use of three different logic styles (predicate logic, relational calculus and navigation expression). This chapter gives examples on properties that they can express.

2. Chapter 4 Language: This chapter introduces what software models and meta-models are. The language used to specify them as well as some of its unique features (eg. Scope, Signatures etc.), are presented.
3. Chapter 5 Analysis: This chapter presents typical forms of analysis behind Alloy. It explains some key parameters (choice of the size of scope) within the analysis, and the power and limitations of these approaches.

The overall treatment is quite informal (No theorems and proofs). There are plenty of examples. Instead of elaborating on the details, interested readers should run the examples via Alloy while reading the text. The examples are already packaged together with the Alloy analyzer. Both conceptual and implementation related issues are covered through them.

Examples Chapter 6 contains four well elaborated examples. Each covers a different kind of applications. They are:

1. Leader Election in a Ring: An example on analyzing a distributive algorithm.
2. Hotel Room Locking: The focus is to demonstrate two different style of modeling.
3. Media Asset Management: A modeling problem that many software engineers are familiar with.
4. Memory Abstractions: A pedagogical example on justifying the correctness of a cache memory management system.

In Appendix A, there are more examples that are assigned as exercises.

Appendix The book has a long list of appendices (more than 100 pages). They include exercises; references on the Alloy language and its kernel semantics; diagrammatic notations used in the text and a short description about other modeling languages.

3 Opinion

The intended audience of the book are software designers. A major objective, as stated by the author, is to "bring them back to thinking deeply about the underlying concepts" (See page XIV). One popular perception about Formal methods is that "it is difficult because it requires a mature mathematical ability to understand and apply them properly". With all these points in mind, I find the book quite well written. The author has made serious efforts to communicate the technical ideas to those with less mathematical background. Concepts are explained by examples. All code examples are packaged together with the current version of Alloy and are ready to use. In almost every chapter it comes with a discussion section. Many questions seem to be collected from practitioners. The answers are presented informally but professionally.

In my opinion, readers from a variety of background would benefit from this book. Let me elaborate as follows:

1. Software designers:

(a) Those who have witnessed the success of using model checkers in hardware verification: This book will introduce to them a similar tool from the formal method community that may help to improve software design. They are the likely candidates to study this book and try out Alloy.

(b) Software engineers that have some exposure in traditional formal methods such as program verification via Hoare's Logic: They understand the mathematics rigor underlying formal methods. They may have encountered difficulties or in-feasibility of using formal techniques in their work. They are the likely candidates to study this book and appreciate Alloy, a tool for *light-weight* formal method.

(c) Those who have a real concern about software reliability: A recent article [3] reported that [4] "software has been exhibiting declining levels of dependability". Professionals who recognize the concerns will see the importance of thinking deeply about the underlying concepts in their software design projects. They are the likely candidates to invest their time to study the materials in the book.

2. Students:

(a) Advanced Undergraduates: Unlike textbooks such as [5] this book is not a standard textbook for formal methods. However, students who wish to do a research project (eg. A substantial term project, undergraduate thesis etc.) may find this book helpful. The examples and exercises therein provide plenty of good project ideas.

(b) Graduate Students (Master-level): Graduate students often have exposure to several software tools (eg. Model Checkers, Theorem Provers etc.). It will be instructive for them to compare and contrast the capabilities and differences, say, between Alloy and other tools such as SMV (See [6] for an example of using the model checker in verification of software systems). This book will definitely be helpful to those students.

3. College Professors:

(a) Those who teach programming and introductory software design courses: To them, a software tool that can generate small but adequate examples (or counterexamples) automatically is of great value. The ability to present the examples (or counterexamples) visually is a big plus to demonstrate basic ideas in these courses. As a side note, teachers of formal methods may find more teaching related materials directly from the Alloy website.

(b) Academics with an interest in both logic and computer science: It is often a great idea to see how researchers from other related disciplines tackle their problems. The small scope hypothesis is the main development principle behind Alloy. This hypothesis has a similar flavor as the compactness theorems and the Lowenheim Skolem theorems in classical model theory (See [7]). As a logician, I find the idea of using this hypothesis to navigate a software design process quite intriguing. Of course, those who like to see the technicalities behind Alloy should dig into the literature instead, as this book is not written for that purpose.

Finally, a couple of points should be noted. The book was based on Alloy 3 and the current version (May 2009) is Alloy 4.1.10. As Alloy is being revised constantly and there's an entire website to provide various support for Alloy, the author may like to re-think what materials regarding Alloy should be kept in printed form. What principles, techniques, lessons or tricks are relatively "long lasting" ? As a reader, I find the examples in the text (Chapter 6) most helpful. If there is going to be a second edition, I believe that the insights, the "gems" and the "Aha" moments (while using Alloy) are the best candidates to be added.

Another suggestion is about the tools offered from the formal methods community in general. Theorem provers, model checkers and instance finders are three different types of tools. What are their roles in the analysis of software abstractions ? Readers with less background in formal methods will probably need more explanations earlier on before studying the details of Alloy presented in the text.

References

[1] http://softwareabstractions.org.

[2] http://alloy.mit.edu/alloy4/.

[3] M. Hinchey, M. Jackson, P. Coursot, B. Cook, J. P. Bowen and T. Margaria *Software Engineering and Formal Methods,. Journal of A.C.M.*, 51:9:54–59, 2008.

[4] J. Gray *Dependability in the Internet Era. Proceedings of the High Dependability Computing Consortium Conference.* Santa Cruz, CA. May 2001.

[5] M. Huth and M. Ryan *Logic in Computer Science: Modeling and Reasoning about Systems, Second Education.* Cambridge University Press, 2004.

[6] J. Wing and M. Vaziri-Farahani *A case study in model checking software systems. Science of Computer Programming*, 28:273–299, 1997.

[7] C C. Chang and H J. Keisler *Model Theory, Third Edition.* Elsevier, 1990.

[8] A. Wigderson *Depth through Breadth (or or, why should we go to talks in other areas). Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, page 579 2004, Chicago, IL, USA