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. **Excellence Without a Soul: How a Great University Forgot Education** by Harry Lewis. Review by William Gasarch. This book is about how Harvard had declined; however, it has lessons for us all.

2. The following three books are jointly reviewed by Ronald de Wolf. This review originally appeared in *Quantum Information and Computation*, Vol 3, No. 1, pages 93–96, 2003. It is reprinted here with permission. The books are on Quantum computing.

   - **An Introduction to Quantum Computing Algorithms** by Arthur O. Pittenger.
   - **Quantum Computing** by Mika Hirvensalo.
   - **Classical and Quantum Computation** by A. Yu. Kitaev, A. Shen, and M. N. Vyalyi.

3. **Introduction to Lattices and Order** by B.A. Davey, H.A. Priestley. Reviewed by Jonathan Cohen. This is a book on lattices that can be used by undergraduates- it has exercises, diagrams, and good exposition.

4. **Computational Techniques for the Summation of Series** by Anthony Sofo. Review by Vladik Kreinovich. This is about summing series. It uses complex math (literally- complex analysis) and is a research book, not a textbook.

Books I want Reviewed

If you want a FREE copy of one of these books in exchange for a review, then email me at gasarchcs.umd.edu

Reviews need to be in LaTeX, LaTeX2e, or Plaintext.

Books on Algorithms and Data Structures

1. *A Programmers Companion to Algorithm Analysis* by Ernst Leiss.

2. *Combinatorial Optimization: Packing and Covering* by Cornuejols.

3. *Distributes Systems: An Algorithmic Approach* by Ghosh.

4. *An Introduction to Data Structures and Algorithms* by Storer.

5. *Nonlinear Integer Programming* by Li and Sun.

6. *Biologically Inspired Algorithms for Financial Modelling* Brabazon and O'Neill.

7. *Planning Algorithms* LaValle.

Books on Cryptography

1. *Concurrent Zero-Knowledge* by Alon Rosen.
2. *An Introduction to Cryptography (second edition)* by Richard Mollin.
3. *Algebraic Aspects of the Advanced Encryption Standard* Cid, Murphy, Robshaw.
4. *Cryptographic Applications of Analytic Number Theory: Complexity Lower Bounds and Pseudorandomness* by Shparlinski.
5. *Cryptography and Computational Number Theory* edited by Lam, Shparlinski, Wang, Xing.
6. *Coding, Cryptography, and Combinatorics* edited by Feng, Niederreiter, Xing.

Books on Security

1. *Computer Viruses and Malware* by Aycock.
2. *Network Security Policies and Procedures* by David Frye.
3. *Data Warehousing and Data Mining Techniques for Cyper Security* by Anoop Singhal.
4. *Electronic Postal Systems: Technology, Security, Economics* by Gerrit Bleumer
5. *Cryptographic Algorithms on Reconfigurable Hardware* by Rodriguez-Henriques, Saqib, Perez, Koc.
6. *Preserving Privacy in On-Line Analytical Processing (OLAP)* by Wang, Jajodia, Wijesekera.

Books on Coding Theory

1. *Coding for Data and Computer Communications* by David Salomon.
2. *Block Error-Correcting Codes: A Computational Primer* by Xambo-Descamps.
3. *Authentication Codes and Combinatorial Designs* by Dingyi Pei.
4. *Algebraic Coding Theory and Information Theory: DIMACS Workshop* Edited by Ashikhmin and Barg.

Combinatorics Books

1. *Graphs and Discovery: A DIMACS Workshop* Edited by Fajilowicz, Fowler, Hansen, Janowitz, Roberts. (About using software to find conjectures in graph theory.)
2. *Combinatorial Designs: Constructions and Analysis* by Stinson.
3. *Computationally Oriented Matroids* by Bokowski

Logic and Verification Books

1. *Elements of Finite Model Theory* by Libkin.
2. *Mathematical Approaches to Software Quality* by O'Regan.
3. *Software Abstractions: Logic, Language, and Analysis* by Jackson.

Misc Books

1. *Automata Theory with Modern Applications* by James Anderson.
2. *An Introduction to Game-Theoretic Modelling* by Mestertson-Gibbons.
3. *An Introduction to Difference Equations* by Elaydi.
4. *Difference Equations: From Rabbits to Chaos* by Cull, Flahive, Robson.

Review of[1]
Excellence Without a Soul: How a Great University Forgot Education
Author of Book: Harry Lewis
Published by Public Affairs, 290 pages
Reviewed by William Gasarch (gasarch@cs.umd.edu)
University of Maryland

Disclaimer: I was a Harvard graduate student and Harry Lewis, the author of the book under review, was my adviser.

1 Introduction

This book is about what is wrong with Harvard. Why should we care? Because much of what is wrong with Harvard is also wrong with other universities and other institutions.

The book discusses the usual problems one hears about: Grade Inflation, College Athletics, The Canon, Date Rape (and more generally how schools teach morals and responsibility), and others. However, the book gives such scholarly historical background on these topics that it makes the usual discussion of these issues seems trite. The books is at its best when it deals with factual information that is interesting and most people do not know. The book is at its weakest when it offers strong opinions; however, this is not often.

The book does not offer that many opinions or solutions, but that is all to the good.

In this review I will only discuss three of the topics raised. Other reviews are posted at Harry Lewis's website:

`http://www.eecs.harvard.edu/$\tilde{}$lewis/`

It is to his credit that he posts both positive and negative reviews.

[1]

2 Grade Inflation

Every once in a while some statistic comes out which makes people pay attention to grade inflation. Statistics like "93% of all Harvard Students graduate with Honors". The book contains information about how grades were first assigned, the first time the charge of 'grade inflation' was made (about 8 years after they began using A/B/C/D/F grades), and some thoughts on why we give grades at all. This was all very interesting. Then he concludes that grade inflation is not a problem compared to other issues in education. While this is probably true, one wonders both how he reached that conclusion (it seems disconnected from the prior discussion) and why he spend so many pages talking about it.

The notion that grade inflation is not a problem seems to be at odds with other parts of the book which complain that schools try to give students what they want rather than what they need. However, the final conclusion does not detract from the intelligent discussion that proceeds it.

3 College Athletics

When I think of College Athletics I think of (1) Len Bias overdosing on cocaine my first semester at University of Maryland, (2) Maryland students becoming destructive if their team wins or loses (Celebrations and Disappointments seem to have the same effect), (3) various recruiting scandals I've heard about, (4) pressure to give athletes passing grades (this has not happened to me personally).

This book has another take on college athletics. There is a vast prejudice against college athletes. If a student had virtually any other extra curricular activity then a professor would not assume they were bad students. But if they are an athlete than clearly they are not going to work hard at academics.

The book is pro-college-sports. The book has the history behind the notion of amateurism (originally designed to keep the lower classes out of sports) and points out that problems with the current system. No real answers are offered, but it does make you think and rethink your own prejudices.

4 The Canon

Consider the following logical argument:
Premises:

1. The University wants the students to have a well rounded education.
2. Professors teach students.
3. Professors' research tends to be in rather narrow topics. Therefore
   (a) Professors do not know what a 'general education' should be.
   (b) Professors do not like to teach general courses, preferring instead to teach their narrow specialty.

Conclusion: It is impossible to have a well defined and meaningful core.

The book seems to make this argument but does not conclude that a meaningful core is impossible. The book gives a history of the various types of core requirements Harvard had. What comes through is that this has always been a problem. I half-expected him to end with "not having a core is not a problem" but he does not. Since grade inflation and a core have always been a problem one wonders what is the Soul Harvard used to have? When it was a divinity school?

The book offers some mild suggestions about the core late in the book, but the real contribution is pointing out the problems inherent in having a core. As such its far more though-provoking than the usual "is the core too full of dead white males" debate.

Dr. Lewis was wise to not suggest a particular cannon since that would misdirect the discussion. However, the recent Harvard report on this issue references the book many times and seems to have been influenced by it.

5 Larry Summers

Harry Lewis was a Dean when Larry Summers became president of Harvard. Later, Larry Summers fired Harry Lewis. (The book *Harvard Rules* has an account of this and other aspects of the Summers presidency.) When I told one of my colleagues that Harry Lewis wrote a book about Harvard he inquired if the book was a hatchet job on Larry Summers. It is not. In fact, when I was 3/4 through the book my thought was "Summers is hardly mentioned, and that's probably just as well." However, in the Conclusion sections there is a discussion of Larry Summers. Much like the rest of the book, the discussion is intelligent. To illustrate, here are various things I have heard about Larry Summers firing prior to reading this book.

1. He was fired because he said politically incorrect things, thus showing that academia is a radical liberal enclave.
2. He was fired because he said rude and incorrect things about women in the sciences, thus showing that academia is an enlightened place.
3. He was fired because he actually wanted professors to teach and do relevant research.

The truth seems to serve nobody's agenda and hence is under reported. The truth is that while he may have raised good issues (e.g., the core) he was a terrible manager and leader. In short, he was incompetent. The book under review and also *Harvard Rules* documents this meticulously.

6 Opinion

I recommend this book for giving thought provoking discussion and background on some of the issues facing all colleges today. Computer science faces the problem of "what is core" since its a new field and hence we do not quite know what is core yet. For this topic the book may be a cautionary tale.

Joint Review[2] **of**
An Introduction to Quantum Computing Algorithms
by Arthur O. Pittenger
Birkhäuser, December 1999
Hardcover $44.95 (138 pages) ISBN: 0817641270

and

Quantum Computing
by Mika Hirvensalo
Springer, May 2001
Hardcover $44.95 (190 pages) ISBN: 3540667830

and

Classical and Quantum Computation
by A. Yu. Kitaev, A. Shen, and M. N. Vyalyi
American Mathematical Society, July 2002
Hardcover $59.00 (272 pages) ISBN: 082182161X (softcover should be considerably cheaper)

Reviewed by
Ronald de Wolf[3]

A quick survey on `amazon.com` shows that the number of books on quantum computing ($\gg 20$) is more than 10 times as high as the number of quantum algorithms that we have today (2: Shor's and Grover's). [4] Many people in the field, including this reviewer, feel that Nielsen and Chuang's *Quantum Computation and Quantum Information* is probably the best and most comprehensive of these. Nevertheless, there is room for some other books with different perspectives. Here we review and compare three books that focus mainly on the algorithmic side of the field, and hence can afford to be significantly shorter than Nielsen and Chuang's 675-page volume.

1 Pittenger

Arthur Pittenger's *An Introduction to Quantum Computing Algorithms* reflects its author's own experience in learning the mathematics and theoretical physics required for the subject, as he writes in the acknowledgments. It is generally written in a pleasant and informal style, with much motivation in between the mathematics. Of the three books reviewed here it is probably the most readable.

It consists of three parts of about 40 pages each. The first part (Chapters 1 and 2) covers the computational model: states and operations, some quantum mechanical background, and the circuit model of quantum gates, generalizing classical reversible circuits. The chapter does not go into the issue of approximating arbitrary circuits by a finite set of basis gates. The second part (Chapter 3) reflects the title of the book. It explains the main quantum algorithms: Deutsch-Jozsa, Simon, Grover, Shor, and the generalization of the latter to the Abelian hidden subgroup

[3]CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. `rdewolf@cwi.nl`
[4]Note that this review was written in 2003 so this statement is no longer true.

problem. The chapter is self-contained, except for some of the number theory needed for the classical post-processing and analysis of Shor's algorithm. The final part (Chapter 4) discusses quantum error-correction in detail. It covers the 9-qubit, 7-qubit, and 5-qubit codes, as well as the general framework of stabilizer codes and Calderbank-Shor-Steane (CSS) codes. Fault-tolerant implementation of gates is not dealt with. Still, in just 120 pages this book manages to explain much of the core of quantum computing, and to explain it well.

2 Hirvensalo

Mika Hirvensalo's *Quantum Computing* is more recent but covers quite similar ground. It too is much shorter than Nielsen-Chuang and has a clear computer science focus; one wonders whether Hirvensalo was aware of the earlier Pittenger book, which he does not reference. His style of writing is more formal and mathematical than Pittenger's.

The six chapters of his book explain the circuit model (like Pittenger, without a discussion of universal gate sets), Shor's algorithm, the hidden subgroup problem, Grover's algorithm, and the polynomial method for query complexity lower bounds. As a bonus, the last 80 pages of the book consist of two very extensive appendices. The first covers the mathematical foundations of quantum mechanics. This includes quite a few advanced topics that are not used in the main text, such as an entropic version of the uncertainty relations, Gleason's theorem that every well-behaved assignment of probabilities comes from a density matrix, and a characterization of completely positive maps on density matrices. The second appendix explains some mathematical background: group theory, the discrete Fourier transform, linear algebra, number theory including the continued fraction expansion used in Shor's classical post-processing, and some information theory.

3 Kitaev, Shen, and Vyalyi

The very recent *Classical and Quantum Computation* by Kitaev, Shen, and Vyalyi (KSV) grew out of a course on classical and quantum computing given by Kitaev and Shen in Moscow in 1999. It is a translated and expanded version of an earlier Russian book, which is still available for free at `http://www.mccme.ru/free-books` for those who can read Russian [5]. From a researcher's perspective it is by far the most interesting of the three books, and I will correspondingly be more detailed in discussing it.

The book has three parts: classical computing, quantum computing, and solutions to exercises. The classical part is quite excellent. In a clear and intuitive style of writing it describes the essentials of the theory of classical algorithms and complexity. This covers Turing machines, circuits, reversible computing, NP-completeness, randomized algorithms, and the polynomial hierarchy. This 30-page exposition also includes some more advanced results like $BPP \subseteq P/poly$ and $BPP \subseteq \Sigma_2$.

The quantum part of the book (Chapters 6 to 15) devotes about half of its 120 pages to a thorough exposition of the quantum circuit model, including representing or approximating arbitrary unitaries by means of elementary gates, quantum computation with mixed states, and a detailed account of measurement. After all the details of the circuit model are in place, the book continues in Chapter 13 to describe the phase estimation technique (originally due to Kitaev) and the way

[5] The translation no doubt caused the occasional absence of the definite article in the English text.

it can be used to solve the Abelian hidden subgroup problem, including factoring and discrete logarithms. Chapter 14 deals with a quantum version of the complexity class NP and a complete promise problem for this class. Finally, Chapter 15 describes quantum error-correcting codes and ends with a brief description of Kitaev's work on toric codes and anyons, where error correction would be a natural property of the underlying physical system itself.

Apart from its conciseness and rigor, one of the main strengths of this book is the attention it gives to Kitaev's contributions to quantum computing. These include a detailed analysis of efficient approximation of arbitrary circuits using only gates from a specific finite basis, the Abelian hidden subgroup problem, quantum NP-completeness, and toric codes. These topics are explained in much detail and with many subtleties and insights that are often glossed over in other presentations—and for things like quantum NP-completeness there *is* no other presentation. A good understanding of the quantum part of this book (including the exercises and their solutions) will provide the researcher with invaluable insights and tools for new research.

At the same time, this bias towards Kitaev's quantum work may also be viewed as a weakness of this book. Even if one restricts attention to computer science aspects of quantum computing, various things are missing. The work of a number of key researchers is completely ignored, including that of Andris Ambainis, Harry Buhrman, Daniel Gottesman, Peter Høyer, Richard Jozsa, Michele Mosca, and Umesh Vazirani. Between them, these people have contributed a large fraction of the main results on quantum algorithms and complexity, yet none of their papers is even cited. The Deutsch-Jozsa algorithm is absent; there is nothing about the applications of Grover's algorithm in counting, collision-finding etc. The result that Grover's algorithm is optimal for quantum search is only mentioned in passing and the paper that first proved this (Bennett, Bernstein, Brassard, and Vazirani "Strengths and weaknesses of quantum computing") is not cited. There is nothing else on lower bounds, virtually nothing about quantum communication or communication complexity, no quantum cryptography, etc. The book's cover suggests using it as a textbook for a graduate course on quantum computing, but I fear that such a course would give a somewhat biased view of the field.

A second problem with using this book for a course is the disparity between its classical and quantum parts. These are apparently written predominantly by different authors. The classical part is generally very well and intuitively written, and does not presuppose much. On the other hand, the quantum part is a much less smooth read. It is significantly more demanding and not quite self-contained. For instance, the proof that the standard basis of elementary gates can efficiently approximate circuits over other bases (Section 8.3) assumes some knowledge of Lie groups and Lie algebras, and the last sections about error-correcting codes assume some acquaintance with homology of manifolds. Again, this may be problematic when using KSV as a textbook for a course, since most students will not be very familiar with this material. As another example, the use of quantum phase estimation in quantum algorithms is originally due to Kitaev, but the exposition of his method in the subsequent paper "Quantum algorithms revisited" by Cleve, Ekert, Macchiavello, and Mosca is much more clear than the exposition given here.

The above points notwithstanding, a lot can be learned from this book; much more than from the other two, but it requires a greater effort by the reader. This is fine when that reader is a researcher—and that is probably where the book will be used the most: as a valuable resource for people who want to look up or learn the intricacies of things like the circuit model, quantum NP-completeness, etc.

4 Comparison and conclusion

The overlap between these three books is quite large. All three are explicitly oriented towards computer science, devoting most of their pages to the quantum circuit model and the main quantum algorithms. The contents of the Pittenger and Hirvensalo books in particular are very close. The main differences are that Pittenger has a chapter on error-correction and his style of writing is somewhat more informal and intuitive, while Hirvensalo has a chapter on lower bounds and some more mathematical background (such as continued fractions).

The KSV book offers more than the other two books. This includes a succinct but very nice introduction to a lot of classical complexity theory, a more in-depth discussion of the quantum circuit model, and topics like quantum NP-completeness and toric codes that are not readily available in any other books. On the downside, I found its quantum part more demanding and harder to read at times than the other two books.

All three books are precise and reasonably succinct introductions to the algorithmic aspects of the field of quantum computation. As such they will be most useful to computer scientists and mathematicians who want to learn about the algorithms without being bothered too much by the physics. The books are suitable for a 1-semester course on quantum algorithms (omitting some of the more advanced sections in the case of KSV). All three books have many exercises, but KSV is the only one that gives the solutions as well. Hirvensalo discusses some lower bounds but Pittenger and KSV do not, and none of the books treat communication-based topics like quantum cryptography, channel capacities, or communication complexity, nor the various applications of Grover's algorithm.

So, which book to choose? If you want a very accessible first introduction to quantum algorithms, I would recommend Pittenger's book. If you prefer more formal mathematics, Hirvensalo also gives a good first introduction to roughly the same topics. If you have sufficient mathematical maturity and/or are prepared to do some work while reading, then go for the KSV book. For everyone with a broader interest in quantum computing (including quantum information theory), I would still recommend Nielsen and Chuang over these books. It contains *much* more material, is very readable, and not significantly more expensive than the three books discussed here.

Finally, to all those currently working on yet another book about quantum computing: what this field needs most is more algorithms, not more books.

Review[6]
Introduction to Lattices and Order
by B.A. Davey, H.A. Priestley
Second Edition, Cambridge University Press
Review by Jonathan Cohen[7]

1 Introduction

The idea that a set may come equipped with a natural ordering on its elements is so basic as to pass by unnoticed by most. However, this belies a wonderful opportunity for unifying the study

[7]Computer Sciences Laboratory, The Australian National University, jonathan.cohen@anu.edu.au

of many disparate examples of these creatures. Historically, the study of order has led to a great unification of results in algebra and logic. More recently, it has infused into theoretical computer science, particularly into programming language semantics.

Given two elements, x and y, in some partially ordered set, or poset, P, it is often the case that they have both a least upper bound, $\text{lub}\{x, y\}$, and a greatest lower bound, $\text{glb}\{x, y\}$. An example of such a poset is provided by the natural numbers under their usual order. However, this property certainly does not hold for every poset. For instance, suppose that the underlying set of P consists of those finite subsets of $\mathbf{N}$ that have even cardinality. This becomes a poset by saying that x is less than or equal to y just in case x is a subset of y. Now, suppose that $x = \{1, 2\}$ and $y = \{2, 3\}$. Any upper bound of both x and y has to contain $x \cup y = \{1, 2, 3\}$. Necessarily, a least upper bound would have cardinality 4. However, there is no *least* such set containing $\{1, 2, 3\}$.

If we are in the happy situation that any two elements of our poset P have both a least upper bound and a greatest lower bound, then we say that P is a *lattice*. If P is a lattice and $x, y \in P$, then let us save on effort by writing $\text{glb}\{x, y\}$ as $x \vee y$ and $\text{lub}\{x, y\}$ as $x \wedge y$. The operator $\vee$ is called *join* and the operator $\wedge$ is called *meet*. Playing around with these operators a bit, one may notice some identities starting to crop up. Amongst those identities that hold for elements $a, b, c \in P$, we find the following:

$$\begin{array}{ll} a \wedge a = a & a \vee a = a \\ a \wedge b = b \wedge a & a \vee b = b \vee a \\ a \wedge (b \wedge c) = (a \wedge b) \wedge c & a \vee (b \vee c) = (a \vee b) \vee c \\ a \wedge (a \vee b) = a & a \vee (a \wedge b) = a \end{array}$$

In fact, the above equations are sufficient to define P by making the observation that $a \leq b$ if and only if $a = a \wedge b$. So, we have two ways of looking at a lattice. That is, we can see it as either a special sort of poset or as a set with two binary operators satisfying the above equalities.

This brings us to the book under review, which sets out to provide a thorough introduction to both points of view. Examples of lattices abound in theoretical computer science and, indeed, the book makes no bones about dipping into computational applications on a regular basis. Let's see what's between the covers, shall we?

2 What's Inside?

Chapter 1: Ordered Sets. At a very abstract level, one may define a deterministic program to be a function from a set of input states to a set of output states. Of course, one only really wants to consider those functions that are computable, but let us sweep this concern under the rug. Now, it may very well happen that a given program does not terminate when presented with certain inputs. We can build this possibility into our model by only requiring a program to correspond to a *partial* map.

For sets I and O, let us use $(I \Rightarrow O)$ to denote the collection of all partial functions from I to O. This set carries a partial ordering in a natural way. To wit, take two partial functions $f, g \in (I \Rightarrow O)$. The partial ordering arises by declaring that $f \sqsubseteq g$ if and only if $\text{dom}(f) \subseteq \text{dom}(g)$ and $f(x) = g(x)$ for every $x \in \text{dom}(f)$. Viewing f and g as deterministic programs, we have $f \sqsubseteq g$ if and only if whenever f terminates from some input state, so does g and in the same output state as f. Additionally, g may terminate from some input states that f does not.

The above is one of many examples of ordered sets that kick off this chapter and crops up repeatedly throughout the book when developing aspects of the semantics of programming languages.

Following on from the examples, several important constructions on posets are introduced. Of these, we only mention two that are of particular importance for lattices.

Let P be a poset. We say that P has a *bottom* if there is some $\bot \in P$ such that $\bot \leq x$ for all $x \in P$. Dually, P has a top if there is some $\top \in P$ such that $x \leq \top$ for all $x \in P$. A poset with both a top and a bottom is said to be *bounded*. In the deterministic program setting, $\bot$ corresponds to the program that fails on every input and $\top$ corresponds to an ill defined program. At this point, I cannot resist quoting the authors' view on this situation:

> Accordingly, computer scientists often choose models which have bottoms, but prefer them topless.

Suppose now that we are handed a subset $Q \subseteq P$. The *down set* of Q is the set $\downarrow Q$ consisting of all those elements "below" Q. Specifically, $\downarrow Q := \{x \in P \mid (\exists y \in Q)\ x \leq y\}$.The collection of all down sets of P is denoted by $\mathcal{O}(P)$ and itself forms a poset under the set inclusion ordering. This poset is very important for the representation theory of lattices developed later in the book.

Chapter 2: Lattices and Complete Lattices. In the definition of lattices given in the introduction to this review, it was only required that the join and meet of any *two* elements exists. By induction, we can form the join and meet of any finite number of elements of a lattice. But what if the lattice contains infinitely many elements? Can we always form the join and meet of an arbitrary subset? Sadly, the answer is no. A counterexample is provided by the rational numbers under their usual ordering.

A lattice, L, is called *complete* if the join and meet of any subset exists. Every complete lattice is necessarily bounded by taking $\bot = \bigwedge L$ and $\top = \bigvee L$. By the above remarks, every finite lattice is complete. An example of an infinite complete lattice is provided by the powerset of any infinite set, ordered by inclusion. An important observation is that, for any lattice L, the poset $\mathcal{O}(L)$ forms a complete lattice by taking $\wedge := \cap$ and $\vee := \cup$.

After introducing lattices and complete lattices, the chapter goes on to develop some of the basic constructions on lattices. These include homomorphisms, sublattices and products of lattices. Following this, some basic results concerning complete lattices are derived. The most important result along the way for computational purposes is the Knaster-Tarski Fixpoint Theorem. Briefly, if L and K are posets, a map $f : L \to K$ is called *order preserving* if whenever $x \leq_L y$ in L, we have that $f(x) \leq_K f(y)$ in K. The theorem states that if L is a complete lattice, then any order preserving map, f, from L to itself has both a greatest and a least fixpoint, where a fixpoint is defined to be an $x \in L$ such that $F(x) = x$.

Chapter 3: Formal Concept Analysis. One way to describe an abstract object is by the collection of properties it satisfies. If we agree on a collection of properties of interest, then this provides the basis for a rough comparison of equivalence of objects. This is roughly the starting observation for the young field of Formal Concept Analysis. Here, "object" is usually taken to be an arbitrary physical object, but one could equally well consider objects in some object oriented programming language. Getting slightly more technical, we say that a *context* is a triple (O, A, I), where O is a set of *objects*, A is a set of *attributes* and $I \subseteq O \times A$. Given some $o \in O$, we can look at the set, o', of all attributes that it satisfies. Similarly, we can look at the set, a', of objects that possess a given attribute $a \in A$. There is nothing really stopping us from generalizing this to

arbitrary subsets $X \subseteq O$ and $Y \subseteq A$ in the following way:

$$\begin{aligned} X' &:= \{a \in A \mid (\forall x \in X)\ xIa\} \\ Y' &:= \{o \in O \mid (\forall y \in Y)\ oIy\} \end{aligned}$$

Given some context (O, A, I), a *concept* is a pair (X, Y), where $X \subseteq O$, $Y \subseteq A$, $X' = Y$ and $Y' = X$. The collection of all concepts, denoted $\mathcal{B}(O, A, I)$, can be made into a poset by defining

$$(X_1, Y_1) \leq (X_2, Y_2) \iff X_1 \subseteq Y_1 \iff Y_2 \subseteq X_2.$$

In fact, $\mathcal{B}(O, A, I)$ becomes a complete lattice under this ordering.

The main results of this chapter obtain the completeness of $\mathcal{B}(O, A, I)$ and show, essentially, that any complete lattice L is isomorphic to a concept lattice, namely to $\mathcal{B}(L, L, \leq)$. In the special case where we take L to be $\mathbf{R}$ with its usual ordering, this result says that $\mathbf{R}$ is isomorphic, as a complete lattice, to the set of Dedekind Cuts. This is no accident, a proper explanation comes later in the book when completions are discussed.

Chapter 4: Modular, distributive and Boolean lattices. Let L be a lattice. We say that L is *modular* if for any $a, b, c \in L$ such that $c \leq a$, we have $a \wedge (b \vee c) = (a \wedge b) \vee c$. The seemingly arbitrary name for this property comes from the fact that it is satisfied by modules over a ring, where $\wedge$ is taken to be set intersection and $\vee$ is taken to be direct sum. Many other algebraic examples of lattices, such as the lattice of normal subgroups of a group or the lattice of subspaces of a vector space, form modular lattices.

A stronger property to impose on L is to require that it be *distributive*. That is, for any $a, b, c \in L$, we have $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$. Necessarily, any distributive lattice is modular, but the converse need not hold.

If L is a bounded lattice, then one may wonder whether there is any significant relation between the lattice operations and $\top$ and $\bot$. Given $a \in L$, we say that $b \in L$ is a *complement* of a if $a \wedge b = \bot$ and $a \vee b = \top$. If L happens to be distributive then complements, if they exist, are necessarily unique. Armed with these definitions, we may now call a lattice *Boolean* if it is bounded, distributive and every element has a complement. Of course, this name arises from the fact that there is effectively no difference between them and Boolean algebras.

After defining modular and distributive lattices, the chapter goes on to give several examples of each before discussing some methods for establishing that a given lattice is *not* modular or distributive. Following that, Boolean lattices and Boolean algebras are introduced and several basic results on truth tables, normal forms and digital circuits are derived.

Chapter 5: Representation: the Finite Case. Lattices were defined in such a way that we can see them equally as posets or sets with special binary operators. Given this correspondence, can we find classes of ordered sets corresponding to the lattices introduced in Chapter 4, which were defined only in terms of meet and join? In many cases, it turns out that we can. In the finite case, things work out particularly nicely.

Let L be a lattice and let $x \in L$. We say that x is *join irreducible* if whenever $x = a \vee b$ for some $a, b \in L$, either $x = a$ or $x = b$. If L has a bottom, then we further require that $x \neq \bot$. The set of all join irreducible elements of L is denoted by $\mathcal{J}(L)$.

After a brief introduction to the general strategy behind obtaining representation theorems, the representation theorem for finite Boolean lattices is proved. The most important result stemming from this representation is that every Boolean lattice is isomorphic to $\mathbf{2}^n$ for some $n \geq 0$, where $\mathbf{2}$ is the unique Boolean lattice on a two element set.

Of much greater interest for later chapters is the representation theorem given for finite distributive lattices. Here, it turns out that any finite distributive lattice L is isomorphic to the poset $\mathcal{O}(\mathcal{J}(L))$. As a corollary, it follows that every finite distributive lattice is isomorphic to a sublattice of $\mathbf{2}^n$ for some $n \geq 0$. The representation theorem for finite distributive lattices is deceptively straightforward. In essence, it goes through by the fact that any finite lattice is complete, since we saw previously that $\mathcal{O}(L)$ is complete for *any* lattice L. Since infinite lattices need not be complete, significantly more machinery needs to be developed before considering *infinite* distributive lattices in a later chapter.

Chapter 6: Congruences. This is the shortest chapter of the book and provides a glimpse of one of the ways that lattices arise in modern algebra. It can be safely skipped without losing the continuity of the book.

Roughly speaking, a congruence is an equivalence relation on an algebra that is "compatible with the operations". Congruences correspond, for instance, to normal subgroups of a group and ideals of a ring. Given a lattice L, an equivalence relation $\equiv$ on L is called a *congruence* if whenever $a \equiv b$ and $c \equiv d$ then $a \vee c \equiv b \vee d$ and $a \wedge c \equiv b \wedge d$, where $a, b, c, d \in L$.

After defining congruences on lattices and characterizing the special properties of the blocks of the underlying equivalence relation, the chapter goes on to show that the set of all congruences on a lattice itself forms a distributive lattice.

Chapter 7: Complete Lattices and Galois Connections. The collection of all subspaces of a vector space, V, can be given a lattice structure by first defining meet to be set intersection. But how are we to define join? Clearly we cannot just define it to be set union, since the union of two subspaces of V need not be a subspace. The trick is to define, for subspaces S_1 and S_2 of V, their join to be $\text{Span}(S_1 \cup S_2)$.

The above example works because Span() is a *closure operator*. For a poset P, a map $c : P \to P$ is called a *closure operator* if, for any $x, y \in P$, we have:

$$x \leq C(x), \quad C \leq y \Rightarrow C(x) \leq C(y), \quad C(C(x)) = C(x).$$

The first half of the chapter is concerned with a certain class of closure operators, which are called "algebraic". As the name suggests, these are of great importance in modern algebra. In a nutshell, algebraic closure operators are intimately related to "algebraic lattices" and it can be shown that every algebraic lattice is isomorphic to the lattice of subalgebras of some abstract algebra. This last result is certainly beyond the scope of the book under review. Nevertheless, some of the most important features of algebraic lattices and algebraic closure operators are developed, hinting at the development of domain theory later in the book.

The second half of the chapter develops some aspects of the theory of Galois connections, which were hinted at in Chapter 3. Let P and Q be posets. Suppose that $f : P \to Q$ and $g : Q \to P$ are order preserving. Then, we say that f and g form a *Galois connection* if for any $p \in P$ and $q \in Q$, we have $f(p) \leq q \iff p \leq g(q)$. Given that f and g form a Galois connected pair, it is not hard to show that both $f \circ g$ and $g \circ f$ form closure operators. Conversely, one can show that any closure operator arises in this way.

The chapter concludes with a section on completions. Showcased is the Dedekind-Macneille completion, which is a generalization of the Dedekind Cut method of completing the rationals to an arbitrary poset.

Chapter 8: CPOs and Fixpoint Theorems. After an extended theoretical development, the book now begins to edge back towards computational applications.

Suppose that P is a poset and that $D \subseteq P$. We say that D is *directed* if for any $x, y \in D$, the pair $\{x, y\}$ has an upper bound $z \in D$. Now, consider a map $f : \mathbf{N} \to \mathbf{N}$. If $g \in (N \Rightarrow N)$ is a partial map having finite domain such that $g \sqsubseteq f$, then we may regard g as providing a finite approximation to f. The collection of all such partial maps forms a directed subset of $(N \Rightarrow N)$ whose join is f.

Motivated by the above example, we say that a poset P is a *Complete Partial Order*, or CPO, if the join of any directed subset of P exists. Of course, any complete lattice is also a CPO. An example of a CPO that is not a complete lattice is provided by the set of all binary strings under the prefix order.

After defining CPOs, the chapter goes on to define some basic constructions on CPOs. The heart of the chapter follows this and develops three fundamental least fixpoint theorems for CPOs in the spirit of the Knaster-Tarski fixpoint theorem for complete lattices, which are put to good use in the following chapter. The chapter concludes with some useful results on computing least fixpoints for compositions of Galois connected order preserving maps on complete lattices.

Chapter 9: Domains and Information Systems. This is the computational climax of the book. Suppose that P is a CPO and let $p \in P$. we say that p is *finite* if for every directed subset $D \subseteq P$, if $k \leq \bigvee D$, then $k \leq d$ for some $d \in D$. The collection of all finite elements of a CPO P is denoted $F(P)$. For the CPO consisting of all binary strings under the prefix order, the finite elements are precisely the finite strings.

If the meet of any nonempty subset of a CPO P exists, then we say that P is a *complete semilattice*. A *domain* is a complete semilattice P, such that for every $p \in P$, we have $p = \bigvee(\downarrow \{p\} \cap F(P))$. It can be shown that $\downarrow\{p\} \cap F(P)$ is necessarily a directed set. Examples of domains include $(S \Rightarrow S)$ for any $S \subset \mathbf{R}$, as well as the collection of all binary strings.

After introducing domains, the chapter goes on to define the notion of an information system. Roughly, an information system is a triple $\langle A, Con, \vdash \rangle$ such that A is a set of "tokens", Con is a nonempty set of finite subsets of A and $\vdash$ is an "entailment relation" between members of Con and members of A. The set Con and the relation $\vdash$ are subject to some constraints.

One of the main results of the chapter is to show how to set up a bijective correspondence between domains and informations systems, so that results about one may be transferred to the other. Following this, a very brief introduction to denotational semantics and the role that domains play in them is provided.

Chapter 10: Maximality Principles. This chapter mainly collects together a number of tools that will be needed in the final chapter. Several equivalents to the Axiom of Choice are introduced and their implications for distributive lattices and Boolean algebras are discussed. The most important consequences for the following chapter are that every distributive lattice is isomorphic to a lattice of sets and that this embedding carries over to Boolean algebras.

Chapter 11: Representation: the General Case. This is the mathematical climax of the book. Indeed, the mathematical sophistication of the material jumps up a notch, relying on some point set topology, which is briefly introduced in an appendix. The main goal is to provide a concrete description of the images of the embeddings of distributive lattices and Boolean algebras introduced in the last chapter. The first stop is Stone's Representation Theorem, which shows that every Boolean algebra is isomorphic to the Boolean algebra of clopen subsets of some compact totally disconnected topological space. After a very brief foray back into logic, the chapter returns to the question of what the image of a bounded distributive lattice looks like under the embedding. This is the content of Priestley's Representation Theorem. The main problem is that, since we no

longer have complements as in the Boolean case, we need to find a way to exploit the underlying order more. A *Priestley Space* is a set, X, which caries both a partial order and a topology such that the topology is *totally order-disconnected*. What this means is that for any $x, y \in X$ such that $y \not\leq x$, there is a clopen down set U such that $x \in U$ and $y \notin U$. An example of a Priestley Space is provided by the Cantor Set, with the order inherited from the interval $[0, 1]$. Priestley's Representation Theorem says that every bounded distributive lattice is isomorphic to the lattice of clopen down sets of some Priestley Space. Following on from the proof of this very important theorem, the chapter concludes with a discussion of duality.

3 Opinion

The book is written in a very engaging and fluid style. The understanding of the content is aided tremendously by the very large number of beautiful lattice diagrams. **TeX**nicians may be interested to note that there are descriptions of the algorithms used for drawing various classes of lattices dotted throughout the book.

An interesting pedagogical development is the inclusion of "Exercises from the text" at the end of many chapters. These ask that the reader fill in the details of proofs, substantiate claims made in the chapter etc. In short, it points out things that the reader should have been checking as she read the chapter. This is an excellent idea for an introductory text such as this, as it helps students make the transition to reading monographs and research papers. In addition, there are copious exercises, most of which extend the theory and give a glimpse of more advanced topics. As such, the book is particularly well suited to self study.

While there are technically no prerequisites for the book, as everything is developed from scratch, some prior exposure to abstract algebra would be very useful. In particular, a reader without such a background is likely to find the final few chapters very heavy going.

There are at least two qualitatively different courses that could be taught from this book. For a very computationally oriented course, one could use Chapters 1,2,4,7,8 and 9 as a basis for an introduction to programming language semantics, possibly supplemented with some extra references on domain theory and denotational semantics. For a mathematically oriented course, one could use Chapters 1,2,4,5,7,10 and 11 as an introduction to lattice theory, perhaps supplemented with some additional references on universal algebra.

One minor quibble is that the decision to split the discussion of complete lattices into two chapters, which appear far apart from each other in the book, is slightly strange. While it is clear that the authors' intention is to present the minimal amount of material necessary to introduce a particular application, this does place a strain on some of the discussion. In particular, the constructions in Chapter 3 would be more perspicuous had some material on abstract Galois connections already been presented.

While the authors note in the preface that coverage of universal algebra and category theory is beyond the scope of the book, the inclusion of an entire chapter on congruences seems rather arbitrary. The space used to describe parochial properties of congruences may well have been better used providing an introduction to basic categorical language, which would have helped to unify much of the discussion of duality.

These concerns aside, the book provides a wonderful and accessible introduction to lattice theory, of equal interest to both computer scientists and mathematicians.

Review of[8]
Computational Techniques for the Summation of Series
Author: Anthony Sofo
Kluwer Academic Publishers, New York, 2003
206 pages, $105.00

Review by Vladik Kreinovich, vladik@utep.edu

Anyone who browsed through D. Knuth's *Art of Computer Programming* books knows that in many cases, the computational complexity of an algorithm – be it worst-case or average-case complexity – can be described in terms of recurrence relations or series. If we simply describe this complexity as, say, a sum of the series, we do not gain much: computing this sum is often not much faster than simply running the original algorithm, and analyzing how this complexity grows with the size of a problem is very difficult. If we can have an *analytical expression* for the corresponding sum, then the situation changes drastically: we can compute its value fast, and we can efficiently analyze its asymptotic properties.

In his analysis of algorithms, Knuth encountered a few series for which the analytical expressions for the sum were already known, a lot of series for which he himself succeeded in deriving the corresponding analytical expressions (and quite a few cases in which he did not succeed). Many of his new cases were somewhat similar, so he conjectured that there may be a general algorithm for computing such sums.

Since this 1970s hypothesis, researchers have indeed found algorithms that cover most of Knuth's (and related) problems. These algorithms have been implemented in efficient code, and there is a very nice and very accessible book "$A = B$" by Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger – the leading researchers who contributed to this breakthrough – a book that covers the problem, the algorithms, the mathematics behind the algorithms, and the main ideas behind the efficient computer implementations of these algorithms.

The resulting algorithms are very general, so general that in the $A = B$ book, the only mention of computational complexity is in the motivations part (and in the names of examples). And indeed summation of series is useful outside computational complexity as well. Let me explain this in more detail. In computer science, everything is discrete: time t is discrete, the size of the input data is discrete. As a result, for example, a natural way to describe the dynamics of a computer system (be it a finite automaton or a Turing machine or a state of the algorithm) is to describe how the state $s(t+1)$ at the next moment of time $t+1$ depends on the state $s(t)$ at the previous moment of time t. As a result, in computer science, difference equations of the type $s(t+1) = F(s(t))$ (and the corresponding series) naturally appear.

In the mathematical analysis of an algorithm, we normally use the integer-valued clock time t; in terms of physical time, in which each clock tick takes time Δt, the above relation takes the form $s(t+\Delta t) = F(s(t))$. The smaller Δt, the closer the "next" moment of time $t+\Delta t$ to the original moment t. In nature, the time t is continuous. Therefore, to describe the dynamics of a physical system, we have to take $\Delta t \to 0$. In this limit, a natural description of the change in $s(t)$ is the derivative. As a result, a natural way to describe the dynamics of a physical system is to describe how the derivative $ds(t)/dt$ depends on its state $s(t)$.

This was Newton's original idea, an idea that led to the blossoming field of differential equations $ds(t)/dt = F(s(t))$. There exist many algorithms for solving differential equations (and, of course,

many interesting open problems as well). However, it is known that in many practical situations, differential equations provide only an approximate description of the situation. Indeed, Newton-type differential equations implicitly assume that the change in the state at a moment of time t is uniquely determined by state of the system at the exact same moment of time; in other words, we assume that all the interactions are instantaneous. In real life, interactions take some time, so for some components i of the state s, the change $ds(t)/dt$ at moment t depends on the values of these components not at the same moment t, but rather in some previous moments of time $t - t_i$.

For example, in relativistic mechanics, when we describe the dynamics of an electrically charged particle A in an electromagnetic field generated by a charged particle B, then the corresponding acceleration of A at time t is determined not by the position of B at the same moment of time t, but rather by the position of B at the time $t - d/c$, where d in the distance and c is the speed of light (the speed with which electromagnetic interaction travels). Similarly, in biology, the number of newly born rabbits is determined not only by the total number of rabbits at a given moment of time, but also on the number of rabbits of different ages, i.e., in effect, on the number of rabbits born at different moments of time in the past.

In such situations, instead of a differential equation, we have either a differential-difference equation

$$ds(t)/dt = F(s(t - t_1), \ldots, s(t - t_n))$$

(when we have finitely many different delays $t_1, \ldots, t_n$), or, more generally, a differential-integral equation, e.g., equation of the type

$$ds(t)/dt = \int F(s(t - u)) \cdot A(t, u)\, \mathrm{d}u.$$

How can we solve such equations? Good news is that in physics, dependencies are usually smooth – even analytical. As a result, solutions are also normally analytical, i.e., they can be represented as Taylor series $s(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + \ldots$ (and the dependence on the parameters is also often analytical). If we substitute this analytical expression into the original equation, we can often recurrently determine the coefficients a_0, a_1, etc., one by one. This is, crudely speaking, how physicists often work. This is, e.g., how astronomers determine the trajectories of the celestial bodies in real-life situations, in which the analytical expressions are not known. Strictly speaking, the above description is, to some extent, an oversimplification. In addition to Taylor series, we have trigonometric (Fourier) series that describe periodic processes, Laplace transforms – with terms of the type $\exp(-p \cdot t)$ that describe fast decrease such as radioactive decay, etc. In all these cases, however, the resulting solution *is* a sum of the series.

There are few cases when we know an analytical solution – like a 2-body gravitational system of Newton's mechanics. In such cases, we can predict the system's behavior faster, we can describe its asymptotics easily – exactly the same advantages as with the analytical expressions corresponding to computational complexity.

It is therefore desirable to search for series that allow an analytical expression for the sum. This is what the book is about.

Why cannot we use methods from the $A = B$ book? In some cases, we can; however, these methods are mainly related to series of "hypergeometric" (crudely speaking, rationally recursive) type most frequently encountered in computational complexity applications, while series in physics and biology are often of more general type.

For computer science-type series, a general algorithm is known. In contrast to computer science-type series, series occurring in natural problems can be of drastically different types. It is therefore difficult to expect that we would be able to find a general algorithm for summing such series. Instead, the author presents several techniques and shows that they can be very helpful in different applied problems. The main idea behind these new techniques is the idea of residue of a complex analytical function. It is known that in the complex plane, the integral $\int\limits_\Gamma f(z)\,\mathrm{d}z$ of an analytical function $f(z)$ over a contour Γ is 0 – provided that this function $f(z)$ is well defined everywhere inside Γ. If, inside Γ, a function $f(z)$ has a few points ("poles") in which it takes infinite values (e.g., $1/z$ is infinite for $z=0$), then we can reduce the computation of $\int\limits_\Gamma f(z)\,\mathrm{d}z$ to the computation of integrals of $f(z)$ over small circles in the vicinity of each such pole. The limits of these integrals are called *residues*.

Residues are a known tool in the theory of analytical functions, they are useful to find asymptotics. The author extends their use to providing explicit expressions for many interesting series, including many series coming from practical physical problems.

At first glance, complex analytical functions may sound like a weird concept, typical of complex numbers, with no real-number analogy. However, it is worth mentioning that many of the original Euler's ideas of dealing with complex numbers and analytical functions come from the fact that an analytical function can be viewed as a natural analogue of a polynomial – albeit a polynomial with infinitely many terms. This is exactly how Euler came up with his formula for an explicit analytical expression for $\sum\limits_{n=1}^{\infty} n^{-2}$ – the first example given in the book. It is known that an n-degree polynomial $P(x)$ with roots $x_1,\ldots,x_n$ can be described as a product $C\cdot(x-x_1)\cdot\ldots\cdot(x-x_n)$, for some constant C. We can view $\sin(x)$ as a polynomial of an infinite degree; its roots are $0,\pm\pi,\pm 2\pi,\ldots$ Therefore, in the limit, we can represent $\sin(x)$ as $C\cdot x\cdot(x-\pi)\cdot(x+\pi)\cdot(x-2\pi)\cdot(x+2\pi)\cdot\ldots$ Combing terms $x\pm k\cdot\pi$, we get $\sin(x)\sim C\cdot x\cdot(x^2-\pi^2)\cdot(x^2-4\pi^2)\cdot\ldots$ For the infinite product to make precise mathematical sense, we must have terms that tend to 1; so we should rescale the terms from $x^2-k^2\cdot\pi^2$ to $1-x^2/(k^2\cdot\pi^2)$. As a result, we get:

$$\frac{\sin(x)}{x}=C\cdot\left(1-\frac{x^2}{\pi^2}\right)\cdot\left(1-\frac{x^2}{4\pi^2}\right)\cdot\left(1-\frac{x^2}{9\pi^2}\right)\cdot\ldots$$

When $x\to 0$, the left-hand side tends to 1, while the right-hand side tends to C; hence $C=1$. If we expand both sides of this equation into Taylor series, and consider the next term (proportional to x^2), then in the right-hand side, we get x^2 with the coefficient

$$-\frac{1}{\pi^2}\cdot\left(1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots\right),$$

while in the left-hand side, the Taylor series formula gives us $-1/6$. Thus, $\sum\limits_{n=1}^{\infty}=\pi^2/6$.

This Euler's derivation is heuristic. One way to make it precise is to go from roots to poles and use contour integrals. Transition from roots to poles is easy: a function $f(z)$ has a root if and only if $1/f(z)$ has a pole; so, we must consider contour integrals of the function $1/\sin(z)$ and related functions.

So far, I have been praising the book, but let me give you some fair warnings. In short, the main warning is that it is a research book, not a textbook. It assumes that the reader knows a lot

about complex functions, way beyond what even most mathematicians know. For example, p. 3 starts with a formula

$$\sum_{n=-\infty}^{\infty} (-1)^n f(n) = -\sum_j \mathrm{Res}_j(\pi \ \mathrm{csc} \ \pi z \ f(z)),$$

which is given without a derivation or a reference, kind of assuming that the reader must be able to understand why such formulas are true. The ability of the reader to understand is not helped by the fact that the author follows a mathematical tradition (unfortunate tradition, in my viewpoint) of not placing the arguments of trig functions in parentheses; for example, the expression in the above formula means $\csc(\pi z) \cdot f(z)$. (We mortals who have to program such formulas know better and usually place parentheses to avoid confusion.)

Although the author proves theorems and provides definitions – the activity that is usually presented in a formal definition-theorem-proof style – the author's style is very informal, which may be confusing to some readers. For example, the very first definition (on p. 11) not only defines the notion, but also places some results about this new notion as a part of this same definition. Formulations of some theorems (e.g., Theorem 5.9 on p. 90) not only describe the theorem itself, but also includes the ideas behind their proofs within the same formulation. Both things are very un-orthodox from the mathematical viewpoint.

The author's emphasis is on proofs and examples, not on algorithms. This comment refers not only to his own work, but also to his survey of alternative techniques in Chapter 2 – this survey does not give us any idea how exactly the surveyed methods work. What is also lacking is comparison with other methods. Yes, there are cases when other methods do not work, but in many examples, both the $A = B$ methods and the author's techniques can be applied. In such situations, it would be nice to know which method is better in some reasonable sense (faster? easier to understand?).

In short, this is a good book, but it is *not* a pedagogically nice exposure of a field in which almost all problems are solved. It is rather a raw and exciting glimpse of a field in action, a field that is definitely worth looking at.