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. **Algorithmic Geometry** by Jean-Daniel Boissonat and Mariette Yvinec. Reviewed by E. W. Čenek. This is a textbook on Computational Geometry. It is big on principles, but not implementation.

2. **Branching Programs and Binary Decision Diagrams: Theory and Applications** by Ingo Wegener. reviewed by Lance Fortnow. This book is a monograph on Branching Programs that is quite comprehensive.

3. **Networks in Distributed Computing (DIMACS proceedings)** Edited by Marios Mavronicolas, Michael Merritt, and Nir Shavit. Reviewed by Ivelin Ivanov. This is a collection of articles on Networks from a conference in 1997.

4. **Graph Theory and Its Applications** by Jonathan Gross and Jay Yellen. Reviewed by David Marcus. This is a text on graph theory which seems to really present applications as opposed to lip service.

5. **Graph Theory** by William Tutte. Reviewed by Christopher G. Jennings. This is an introduction to Graph Theory which is highly mathematical.

6. **Data Refinement: Model-Oriented Proof Methods and Their Comparison** by Willem-Paul de Roever and Kai Engelhardt. Reviewed by Reviewer: Dan Hestand. This is about the process by which an abstract model of a program gets transformed into a real program.

7. In the last column William Gasarch reviewed **Proofs and Refutations** by Lakatos. Included in that review were some remarks about the relevancey of Philosophy of Math for theoretical computer scientists. David Molnar disagrees with some of what was said and was invited to write a response. This is included, as well as a response to the response.

I am looking for reviewers for the following books

If you want a FREE copy of one of these books in exchange for a review, then email me at gasarchcs.umd.edu If you want more information about any of these books, again, feel free to email me. Reviewing a book is a great way to learn a field. I have personally reviewed books and then went on to use what I learned in my research.

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

Books on Algorithms, Combinatorics, and Related Fields

1. *Randomized Algorithms: Approximation, Generation, and Counting* by Bubley.

2. *Algorithm Design: Foundations, Analysis, and Internet Examples* by Goodrich and Tamassia.

3. *An Introduction to Data Structures and Algorithms* by Storer.
4. *Graph Colouring and the Probabilistic Method* by Molloy and Reed.
5. *Random Graphs* by Bela Bollobas.
6. *Geometric Computing and Perception Action Systems* by Corrochano.
7. *Structured Matrices and Polynomials: Unified Superfast Algorithms* by Pan.
8. *Computational Line Geometry* by Pottmann and Wallner.
9. *Bioinformatics: The Machine Learning approach* by Baldi and Brunak.
10. *Computational Commutative Algebra* by Kreuzer and Robbiano.
11. *Algorithms Sequential and Parallel* by Miller and Boxer.
12. *Computer Algorithms: Introduction to Design and Analysis* by Baase and Van Gelder.
13. *Linear Optimization and Extensions: Problems and Solutions* by Alevras and Padberg.
14. *Number Theory for Computing* by Yan.
15. *An Introduction to Quantum Computing Algorithms* by Pittenger.

Books on Cryptography

1. *Foundations of Cryptography: Basic Tools* by Goldreich.
2. *Modern Cryptography, Probabilistic Proofs and Psuedo-randomness* by Goldreich.
3. *Introduction to Crytopgraphy* by Buchmann.
4. *Secure Communicating Systems: Design, Analysis, and Implementation* by Huth.
5. *Elliptic Curves in Crytography* by Blake, Seroussi, and Smart.
6. *Coding Theory and Cryptograph: The Essentials* by Hankerson, Hoffman, Lenoard, Linder, Phelps, Rodger, and Wall. *Introduction to Cryptography with Coding Theory* by Washington and Trappe.

Books on Complexity and Logic

1. *Models of Computation: Exploring the Power of Computing* by John Savage.
2. *Logic and Language Models for Computer Science* by Hamburger and Richards.
3. *Complexity and Information* by Traub and Werschulz.
4. *Logic for Applications* by Nerode and Shore.
5. *The Classical Decision Problem* by Borger, Gradel, and Gurevich.
6. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods* by Roever, De Boer, Hannemann, Hooman, Lakhnech, Poel, and Zwiers.

7. *Set Theory for Computing* by Cantone, Omodeo, and Policriti. *Domains and Lambada-Calculus* by Amadio and Curien.

8. *Derivation and Computation* by Simmons.

DIMACS workshop books

The following are DIMACS workshop books which are collections of articles on the topic in the title.

1. Randomization Methods in Algorithm Design.

2. Multichannel Optical Networks: Theory and Practice.

3. Advances in Switching Networks.

4. Mobile Networks and Computing.

5. Robust Communication Networks: Interconnection and Survivability.

Review of: **Algorithmic Geometry** [2]
by Authors: Jean-Daniel Boissonat and Mariette Yvinec
Translated by: Hervé Bronnimann
Publisher: Cambridge University Press

Reviewed by E W Čenek, University of Waterloo

1 Overview

Consider a set of points in the plane; is it possible to divide the plane into regions so that each region is dominated by one of the points? Or, given a set of line segments, is it possible to enumerate all the intersections? Given a set of points in d dimensions, is it possible to calculate a polytope which has minimal volume but includes all the points? Can we determine the influence of a set of objects on a give space?

The ability to build and manipulate geometric objects efficiently is required in many different disciplines, including robotics, computer graphics, and medical imaging. While the first cases of constructive geometry can be traced back to Euclid, the field truly came into its own in the mid-1970s, emerging eventually as a fully-fledged scientific discipline. The authors present this book as an introduction to the field, and cover many of the fundamental methods.

2 Summary of Contents

The book is divided into five parts, each consisting of multiple chapters.

The first part is devoted to the algorithmic tools which will be used throughout the book. These tools include notions of complexity and basic data structures, and a discussion of both the deterministic and random methods used in geometry.

In two dimensions, a convex hull of a set S of points is a convex polygon which contains all points in S inside the polygon. Similarly, in three dimensions, a convex hull is the convex polyhedron which contains all points in S. This notion can easily be extended to convex hulls

in d dimensions, also known as polytopes. Every set of points has many possible convex hulls; the interesting convex hulls is the one with minimum volume. Part II - Convex Hulls consists of one chapter which carefully and concisely defines polytopes and convex hulls mathematically, followed by several chapters which discuss methods for finding convex hulls in the general case of d dimensions, and the more restricted 2 and 3 dimensional cases. The last chapter in this section is devoted to linear programming which is offered both as an application of, and a solution for, finding the polytope of a set of d-dimensional points.

Fundamental objects of geometry, including polygons and polyhedrons, are sometimes described in terms of elementary objects with a bounded complexity. Triangulating an n-sided polygon, for instance, results in a set of non-overlapping polygons, each of which has exactly 3 sides, so that their union is exactly the original polygon. Part III - Triangulations devotes Chapter 11 to the basic definitions and combinatorics of triangulations in the general case. Chapters 12 and 13 discuss the specific cases of 2 and 3 dimensional triangulations, respectively.

The arrangement of a finite set of curves or arcs in the plane is the decomposition of the plane into regions induced by these curves and arcs, so that no arc passes through the interior of a region. Part IV - Arrangements discusses these arrangements, first by describing them for the general d dimensional space (Chapter 14), and then considering the cases of line segments in the plane (Chapter 15), and triangles in 3 dimensions.

Voronoi diagrams are used to model objects and the distance between them. The simplest case is the case where the objects are points in the plane and the distance is the usual Euclidean metric. Then the Voronoi diagram divides the plane into regions so that each region contains one of the objects, and so that for any other point in that region, the object inside the region is closer than any of the other objects. We can use this, for instance, to model growth processes and in particular the places where growth from two different origins begin to compete with one another. Part V - Voronoi Diagrams discusses these diagrams using both Euclidean (Chapter 17) and non-Euclidean (Chapter 18) metrics of distance. Lastly Chapter 19 discusses the specifics of computing Voronoi diagram in the plane, including a simple sweep algorithm for points, as well algorithms for line segments in the plane and points in two planes.

3 Opinion

The emphasis in this book is on providing a broad overview by at all times choosing to present the most general methods, and to describe those methods in general terms. The particular topics chosen are fundamental to the field, and pointers to extensions are included with the bibliographical notes at the end of each chapter.

The authors have chosen to concentrate on the principles underlying each method rather than detailing the implementation. Thus the actual implementation of the algorithms is left as an exercise to the reader. The amount of implementation detail offered in Part I, Chapter 2 (Tools - Basic Data Structures) is somewhat greater than in the following parts, but the amount of detail presupposes that the reader either already knows how to build these data structures, in which case the chapter can be browsed for a quick review, or that the reader is a competent programmer who can build the data structures fairly straightforwardly. For the reader who has no such background, the chapter does not contain enough detail to actually implement the data structures. However, the reader who can implement the data structures is well on her way to implementing the methods described throughout the rest of the book.

Review of

Branching Programs and Binary Decision Diagrams: Theory and Applications[3]
Author of Book: Ingo Wegener

Series: SIAM Monographs on Discrete Mathematics and Applications
Published by SIAM in 2000
$92.50

Author of Review:Lance Fortnow, NEC Research Institute

Note: This review was originally prepared for publication in SIAM Review.

When Wegener wrote his 1986 monograph *The Complexity of Boolean Functions* (Wiley-Tuebner Series in Computer Science, Wiley) he devoted his last chapter to Branching Programs, a computation model where a path is taken on an acyclic graph where each node queries an input bit and that determines which edges one can take. Over the intervening decade and a half research in circuit complexity, which formed the bulk of the 1986 book, showed limited progress but we have seen considerable work in branching programs. So it comes as no surprise that Wegener focuses his current text on branching programs. Binary decision diagrams are just another name for branching programs that came out of the program verification community.

Branching programs give a very simple view of a computational process. Though general branching programs can model any computation, some restricted models have a nice combination of some reasonable computational power combined with the ability to formally understand and manipulate them. In particular Oblivious Binary Decision Diagrams (OBDDs), where each input bit is queried once in some previously fixed order, share many nice attributes with finite automata, like quick algorithms for minimality and equivalence. As the models increase in complexity, we have a corresponding increase in their computational power and a decrease in what we can understand about them.

Wegener's new book gives quite an extensive coverage of branching programs, covering the various theoretical models and essentially all of the major and minor results about them. For example, the book has several chapters on OBDDs giving many details of the algorithmic techniques used for them. Wegener devotes an entire chapter to the problem of finding a good order of the input bits to minimize the complexity of the OBDD.

I find the proofs understandable though I wish they leaned a little more toward the intuition than the technicalities. The book is quite comprehensive in the proofs that it gives. I was a little disappointed that the proof of my favorite recent work in branching programs, Ajtai's lower bounds for depth-restricted BPs, was only given as a citation to Ajtai's hard to read paper but perhaps the result was too recent to be fully included in this book.

The greatest strength of this book, its comprehensiveness, is also its greatest weakness. If one is searching for a specific result, I found the index of little use, and one had to try and guess which section the result occurred. Once found, one has to read several other sections of the book to understand the notation to really understand the proofs. Also the book seems to give equal weight to all results, it is hard to get from the book an idea of the most important results in branching programs.

For a serious researcher who deals with branching programs, I do recommend this book as a reference book. If you use this book extensively you can follow most proofs and find what you need quickly. However, if you plan to just occasionally need a specific result in branching programs it might be easier to just track down the original paper.

I would only recommend this book as a textbook to complement a lecturer who already had familiarity of the subject and give some intuitive understanding to the definitions and proofs.

A practitioner who wants to, say, manipulate OBDDs for program verification, may find this book useful to get a theoretical understanding of the power of OBDDs and some of the general tools one can use to create algorithms for them. But this book is clearly written by a theoretician and significant work is necessary to write code to implement the algorithmic descriptions given.

Overall, for a researcher interested in branching programs/binary decision tree, this books gives a comprehensive view of the subject not really available in any other form. To search, learn or implement from this book requires some effort but you will likely find the information you need from it.

Review of

Networks in Distributed Computing

DIMACS Series in Discrete Mathematics and Theoretical Computer Science volume 45 [4]

Published by AMS, $39.00, 1998

Edited by Marios Mavronicolas, Michael Merritt, and Nir Shavit.

American Math Society

Reviewer: Ivelin Ivanov

1 Overview

The DIMACS Workshop on Networks in Distributed Computing was held October 27-29, 1997, at Rutgers University, in Piscaway, New Jersey. The primary goal of the workshop has been to bring together researchers from both industry and academia insterested in issues related to networks as they appear in the context of distributed computing. Examples of particular problems which have been addresses are ATM networking technology, routing and flow control in modern communication networks, service pricing, security, optical networking, management of Internet, mobile computing and formal verification of communication protocols.

2 Content Summary

This book includes 9 of the papers presented at the workshop. Short summary of each one follows.

"Scalable group membership services for novel applications" This paper presents a new network flow control scheme called "virtual credit", which builds on and improves two successfully used schemes in the ATM industry - the credit based and the rate based schemes. The main advantage of the credit based scheme is its very low cell loss rate, while its main disatvantage is its implementation complexity. One the other side, the main advantage of the rate based scheme is its simple switch architecture, while its main disadvantage is the higher cell loss rate. The proposed virtual credit scheme allocates resources more efficiently than the original credit based scheme and it is also enables easier policing of session than the rate based scheme.

"Scalable Group Memebership Services for Novel Applications" Group communication is essential for the modular design of gropware and other multi-user applications in wide area networks. This paper proposes a new architecture for a scalable group membership service for such environments. Its client-server architecture provides two levels of services and their semantics, each geared to different applications with different needs. It allows lightweight clients to benefit from advanced membership services, while allowing such thin clients to coexistance with full-fledged clients.

"Implementing an Eventually-Serializable Data Service as a Distributed System Building Block" ESDS deals with replicated objects that allow the users of a service to relax consistency requirements in return for improved responsiveness, while providing guarantee for of eventual consistency of the replicated data. This paper offers an optimized implementation of ESDS. The proposed algorithm is given in terms of I/O automata.

"Frequency Assignment in Mobile and Radio Networks" This paper uses graph theoretic and optimization techniques to offer several on-line algorithms for solving the NP-complete problem of frequency assignment in mobile networks. It studies these algorithms efficiency and lower bounds.

"On Limited Wavelength Conversion in Optical Networks" Optical wavelength routing networks are based on wavelength division multiplexing (WDM) transmission technology and on optical switching. WDM allows for transmission of multiple streams on a single fiber by sending each stream using a laser emitting light at a different frequency. This paper surveys recent results on the benefits of limited length conversion in WDM ring networks. It focuses on one of the most fundamental network-level problems: design and allocation of resources, or more specifically, the allocation of wavelenghts to lightpaths.

"UNITE - An Architecture for Lightweight Signaling in ATM Networks" Modern communication networks need to support a wide range of applications with diverse requirements. Network technologies that were traditionally designed and optimized for specific communication services are evolving towards accomodating this diversity on a single integrated communication infrastructure. One difficulty with ATM signalig is the inextricable connection between basic connectivity and Quality of Service management. This paper proposes a new lightweight architecture which separates connectivity from QoS control.

"Timing Based Connection Management" Connection management protocols coordinate the establishment and release of a connection between two hosts acress a wide-area network to allow reliable message delivery. This papers surveys recent work that studies the precise impact of the level of sunchrony provided by the processors' clocks on the performance of connection management protocols, under common assumptions on the pattern of failures of the network and the host nodes.

"Cryptographic Authentication for Real-Time Network Protocols" This paper describes a new security model and authentication scheme for distributed, real-time network protocols used in time synchronization and event scheduling applications. It outlines the design requirements of these protocols and why these requirements cannot be met using conventional cryptography and algorithms. It proposes a new design called autokey, which uses a combination of public-key cryptography and a pseudo-random sequence of one-way hash functions. Autokey has been implemented for the Network Time Protocol.

"Path layout in ATM Networks - A Survey" This paper presents a model for the theoretical study of virtual path layout in ATM networks. The model amoounts to covering the network with simple paths, under various constraints.

3 Conclusion

This book is a compilation of the most interesting papers presented at the DIMACS workshop on Networks in Distributed Computing held in 1997, at Rutgers University. The papers use a rigourous style, while combining theoretical work and experimantal results targeted to practical applications.

It is targeted to the academic and industry professionals with advanced knowledge on distributed computing who seek to update their expertise with the latest research work.

Review [5] of
Graph Theory and Its Applications
Authors: Jonathan Gross and Jay Yellen
Publisher: CRC Press, 1999
$75.00, 592 pages
Reviewed by: David J. Marcus, CEO, M5 Consulting, Inc. djmarcus@m5inc.com

1 Overview

With the advent of computers, graph theory is well on its way to taking center stage along side computer science. It has evolved from a collection of disparate topics into a cohesive framework.

The authors set out to collect the associated techniques, analytical tools, and applications, and transform them into a unified mathematical methodology. The book covers a wide range of topics ranging from basic principles to applications such as: Printed Circuit Boards, Parallel Computing, Register Allocation for Compilers, and many more. Emphasis is on the conceptual rather than the rigorous. Algorithms are presented in pseudo-code which is programming language independent and high-level. This approach makes it easy to follow the essence of the algorithm without getting bogged down in programming details. It also provides the programming devotee an opportunity to excel in translating the pseudo-code into real programs.

The book covers graph theory in two groupings. The first six chapters concentrate on general topics such as basic graph properties, modeling, and representation. This material is common to most graph theory contexts and thus serves as a foundation for the remaining sections of the book. While most of the material assumes no prerequisites, some familiarity with discrete mathematics is helpful. The latter chapters (7 - 15) branch out and focus on more specific topics of interests such as graph coloring, networks, etc.

2 Summary of Contents

2.1 Ch. 1, 2: Models, Structure and Representation

These preliminary chapters introduce graphs and their properties. They are overwhelmingly full of terminology definitions. The reader becomes familiar with edges (directed and undirected), vertices (degree, adjacency, etc), families of graphs (complete, bipartite, regular, etc), components/subgraphs, paths/cycles (Eulerian, Hamiltonian, etc), and so on. In addition, the reader is introduced to graph operations such as insertion/deletion of edges and vertices, detecting graph isomorphism, and others.

After the first two chapters the reader is well versed in the basics of graphs. The exercises at the end of each section provide ample opportunity for the reader to gain the "hands-on" experience in order to build the essential intuition which is so necessary for the later chapters.

2.2 Ch. 3, 4: Trees

These chapters introduce the fundamental concept of a tree - a connected graph with no cycles. The book defines and derives the key properties of a tree T with n vertices:

1. T is connected.
2. T contains no cycles.
3. Given any two vertices u and v of T, there is a unique u-v path.
4. Every edge in T is a cut-edge, and its deletion results in a subgraph having exactly two components.
5. T contains exactly $n - 1$ edges.
6. T contains at least two vertices of degree 1 if $n \geq 2$.
7. Adding an edge between two vertices of T yields a graph with exactly one cycle.

The reader becomes familiar with the commonly accepted tree terminology: depth/level of a vertex, tree height, parent/child vertices, siblings, leaf vertices, etc. The topic of binary trees is correctly given special consideration considering its general usefulness. Tree counting, binary tree traversal, binary-search trees, and priority trees are introduced in chapter 3, while chapter 4 covers in great detail the concept of spanning trees and their associated operations.

2.3 Ch. 5, 6: Connectivity and Traversal

These chapters cover the important topics of vertex and edge connectivity, constructing reliable networks, max-min duality (and Menger's theorems), block decompositions, Eulerian trails/tours, DeBruijn sequences, Hamiltonian paths/cycles [Hamiltonian-type problems are classified as NP-hard], and related topics such as Gray codes and the infamous traveling salesman problems. The material is fast paced and is necessarily more difficult than the earlier chapters, however, that burden is eased somewhat due to the many good illustrations accompanying the text.

2.4 Ch. 7 - 15: Applications of Graph Theory

While the earlier chapters covered the broad spectrum of graph theory basics, within these chapters, the book shifts gears into more specialized applications of graph theory. The reader is introduced (take a deep breath as you read the following list) to: graph operations/mappings (binary operations, linear mappings, network modeling, graph transformations, etc), graph topologies (projections, imbeddings, surfaces, sphere maps, etc), planarity of graphs (Kuratowski's theorem), graph coloring (vertex, edge, map), network flows, graphical enumeration, and algebraic specification of graphs.

As is evident, these chapters cover *a lot* of material. Their inclusion is significant of the enormous value and applicability of graph theory. For the reader interested in greater depth on any of these topics, a rich bibliography is presented for each chapter. The citation references range from "ancient" references to fairly modern.

Here is a taste of the contents. Consider the book's treatment of one of the landmarks of graph theory: Kuratowski's (circa 1930) characterization of planarity in terms of two forbidden subgraphs, K_5 and $K_{3,3}$ (Section 9.3, pages 314 - 320).

Definition: Any graph homeomorphic to either K_5 or $K_{3,3}$ is called a Kuratowski subgraph.

Theorem: *A graph is planar if and only if it contains no subgraph homeomorphic to* K_5 *and* $K_{3,3}$.

The proof outline is as follows:

- Establish that K_5 and $K_{3,3}$ are both nonplanar. Thus containing no Kuratowski subgraph is a necessary condition for planarity. It then remains necessary to prove the "sufficiency" requirement.

- Demonstrate by contradiction. Assume a counter example and select the one with the minimum number of edges among all counter examples. The strategy becomes one of deriving some properties that this counter example would have to have which ultimately exposes the contradiction.

The authors present 3 elegant steps which lead to the contradictions:

1. Show that the minimum counter example would have to be *simple* and *3-connected*.

2. Show that the minimum counter example would have to contain a cycle with three mutually overlapping appendages.

3. Show that any configuration comprising a cycle and three mutually overlapping appendages must contain a Kuratowski's subgraph.

The details are well organized, replete with visualizations. Even if the reader is not able to follow every detail of every step, the essence of the proof is easy to follow.

3 Opinion

The book is generally well-written. I believe the authors have done a good job of meeting their stated goals. It is an excellent book if the reader is looking for a comprehensive fast-paced book on graph theory that is not heavy into rigorous proofs. Some of the theorem proofs are actually single line proofs while others are a hand-waving "Outline of Proof". The book can serve as both a teaching tool as well as becoming a well-worn reference book for the practitioner. Nevertheless, some of the well-earned kudos are:

- The text is easy to read. It allows the user to focus on content rather than deciphering verbiage.

- The figures/diagrams are well thought out and equally well presented.

- The progression of material is logical and well thought out. The authors do an excellent job of covering the prerequisites before reaching any topic. This allows the book to be used for both an undergraduate course in graph theory as well as a graduate level course.

- The exercises at the end of each section of each chapter are well organized in terms of level of difficulty as well as progression. Solving an earlier problem often suggest the approach to a later problem.
- While the coverage of any single topic is not exhaustive, given the constraints of the book (in overall size), they cover a remarkable amount of material.

Having extolled the virtues of the book, it is only proper to point out some of the deficiencies.

I would have liked to see the book contain solutions to the even (or odd) numbered problems at the end of each section. From the title of the book, it is clear that the authors intend the reader to have a hands-on experience. Being able to solve problems is key to mastering the material. As noted in the kudos above, the progression of the problems is well done, however, if the reader stumbles on any of them, the remaining problems may be out of reach. Had the authors provided answers to alternating problems (or at least to selected problems), the user would use the answers as feedback. It would greatly enhance the absorption of the material.

The book goes overboard in defining terms (seems like thousands of them), many of which seem of little value. For example, on page 57 the authors offer as a "definition" the words "Adding a vertex". From the explanation it is clear that they are referring to the act of adding a vertex to a graph. This hardly merits a formal "definition". In fact, the book is cluttered with terms and definitions that are used rarely (typically only in the definition itself).

In some cases the connection between derivation steps are not at all obvious and could benefit from some more explanatory text. For example, in section 3.4 "Counting Binary Trees: Catalan Recursion" (pages 104-106), there is a very clever derivation that *The number b_n of different binary trees on n vertices is given by*

$$b_n = \frac{1}{n+1}\binom{2n}{n}$$

Step (a) of the derivation defines a generating function, $g(x) = \sum_{k=0}^{\infty} b_k x^k$, for the number b_n of different binary trees. It states that

$$1 + x[g(x)]^2 = g(x)$$

It turns out that this innocent little formula can be derived through a tortuous number of steps involving:

- An unnumbered formula near the top of the page (105)

$$b_n = b_0 b_{n-1} + b_1 b_{n-2} + ... + b_{n-1} b_0$$

- Squaring the expression for $g(x)$ and grouping terms
- Recognizing that the $(n-1)^{th}$ term matches the unnumbered equation near the top of the page (hence the label *generating function*)
- Multiplying each term by x and adding 1 to make the n^{th} term match
- Comparing against the definition of $g(x)$ to see that the generating equation holds.

While experienced readers may think that all these steps are obvious and readily apparent, I suspect that many readers will stumble here. More likely, they will take the authors' word for the equation and miss out on the beauty and elegance of its derivation and technique used.

The book is full of unnumbered equations. In most cases the lack of numbering is not a detriment, but in others it can interfere.

A glossary is provided at the end of each chapter throughout the book. The glossary provides a very brief definition of the terms introduced in the chapter. While useful, the glossary items lack a reference to the page where the term was first introduced. As it stands, the reader is forced to locate the page via the index of the book (which is very complete) which tends to minimize the value of the glossary. Furthermore, the glossary section itself within each chapter does not show up in the table of contents. These unfortunate shortcomings greatly diminish the value of the glossary sections.

Review of **Graph Theory**[6]
Author of Book: William T. Tutte
Publisher: Cambridge University Press
Year: 2001, Price: $31.95, Pages: 333
ISBN: 0-521-79489-7
Author or Review:Christopher G. Jennings McMaster University cjennings@acm.org

This title forms volume 21 in the *Encyclopedia of Mathematics and Its Applications*, edited by Gian-Carlo Rota of MIT. It was originally published in hardback in 1984 by Addison-Wesley; this is a softcover edition of the same content. Tutte was an excellent choice as author: as one of the pioneers of graph theory, his unique perspective of what was then 45 years of experience results in a coherent organization of what might otherwise be a disparate collection of theorems.

According to Rota, the encyclopedia series is aimed for the nonspecialist with the hope that it will encourage a wider use of mathematics in problem solving. This statement reminds me of something Donald Knuth once said of *The Art of Computer Programming* in an interview. Knuth explained that he had tried to write the series so that it could be understood by nonspecialists, which had resulted in a work that was understandable by specialists. If he had not done it that way, he said, no one would have been able to understand it. Given that the scope of this series is all of mathematics, one wonders how the term nonspecialist ought to be defined. In the case of the present volume, at least, a fairly strong mathematical background is required, although no previous knowledge of graph theory is needed. It should be accessible to an advanced undergraduate who is willing to put in an effort.

The graph theory groundwork is laid down in the context of connection and edge deletion and contraction in electrical circuits in the first two chapters (including Menger's Theorem). The next two chapters deal with 2-connection and 3-connection. The fifth chapter deals with the reconstruction problem, that is, the properties that can be inferred about a graph if we delete one of its vertices. The sixth chapter delves into digraphs (including the Matrix-Tree Theorem) and Kirchhoff's Laws. The next three chapters discuss cursality and alternating paths, cycles and coboundaries in the context of algebraic duality, and recursive graph functions (map-colourings). The remaining two chapters build a theory of graph planarity derived from combinatorial axioms. The important technique of using bridges to simplify the construction of proofs, which was first introduced by Tutte, is put to good use in these chapters.

The breadth and depth of coverage makes *Graph Theory* a self-contained book that builds a solid foundation in the subject. Starting from first principles, it builds to a point where a dogged

[6]

reader is well prepared to tackle research papers. It includes a good sprinkling of exercises (though no solutions) and extensive references. One could hardly ask more of such a book.

The only complaint I might lodge is that, for a volume in an encyclopedia that emphasizes applications, this title keeps those applications in the background. While the organization of the material tends to be around applications in an abstract sense, the text itself generally does not make these applications explicit. More background on those applications might have made this book more accessible to the nonspecialist audience it is aimed at.

Certainly it is not a book on directly applying graph theory to computer science. The focus is on properties and theorems rather than algorithms. While worthy of study in its own right, this is not the place to start if you're looking for a graph algorithm to solve a problem with a programming project. Instead you should seek the help of a book such as *Graph Theory and Its Applications* (also reviewed in this issue), to which this book would make an ideal complement.

Indeed, this is a mathematics text in the classical style: densely packed with important details that must be dug out by carefully working your way through the theorems and their proofs. Such an approach is not for everyone; many people prefer a style with more hand-holding. However, this gives it the advantage of serving well as both a learning tool and a reference. And in the long run, the extra work it requires pays off with a deeper and more permanent understanding.

Graph Theory is a cohesive introduction to a field whose importance has blossomed in tandem with the rise of computing. Presented with passion and vigor through the eyes of one of its most respected practitioners, it brings together all of the most important results in a way that gives them a meaning and relevance that no one else could have provided. Its reputation as the definitive volume in the field is well-deserved, and it is worth the effort required to work through it.

Review of

Data Refinement: Model-Oriented Proof Methods and Their Comparison [7]
Cambridge Tracts in Theoretical Computer Science #47
Willem-Paul de Roever and Kai Engelhardt
Cambridge University Press, 1998, $80.00

Reviewer: Dan Hestand

1 Overview

The creation of executable software products is a stepwise refinement of an abstract model into a more concrete model with each step adding more detail. At each step we start with an abstract operation and end with a concrete operation. Examples of this include generation of requirements from a Use Case, source code generation from a model, and the generation of executable code from source code. In all cases, the refinement step involves some transformation of the model. If we are concerned with the correctness aspects of our product then we must ensure correctness at each step during the refinement. In practice, this may be nearly impossible for all but the simplest cases and in some steps we may have no adequate theory of formal correctness. Other factors such as resource, schedule, and budget constraints may limit the amount of verification activities applied during a project. Furthermore, simply proving correctness at each step is not sufficient since the transformation may not generate a new concrete model that is an accurate representation of the prior abstract model.

There are, however, techniques for decreasing the amount of effort required to show correctness of a model after some transformation. One such technique is to show correctness at the abstract level (generally an easier task), show that the transformation step preserves correctness, then correctness of the concrete level is ensured by construction. This is the the subject of *Data Refinement: Model-Oriented Proof Methods and Their Comparison*. This monograph deals with the concepts of data refinement and proof of data refinement through simulation in two parts. Part 1 is an exposition of the theory of data refinement and simulation. Part 2 shows how the most common methods for formal verification of software (such as VDM and Z) embody the concepts exhibited in Part 1.

The topics covered in Part 1 lay the groundwork for understanding data refinement and simulation from a detailed theory viewpoint. The central result of this part is the expression of simulation verification conditions as Hoare triples. This result applies whenever the abstract operation is a specification statement. The topics discussed in this part are

- *Introduction to Data Refinement* - what data refinement is and what it seeks to achieve. A very brief history of the development of the data refinement concepts is also presented indicating some of the key players.

- *Simulation as a Proof Method for Data Refinement* - definition of simulation as an effective proof method for data refinement. Abstraction relations are introduced as the means of moving from abstract operation to concrete operation and back. Four types of simulation are also defined: L, L^{-1}, U, and U^{-1}. Soundness and incompleteness of L are shown. The chapter ends with an introduction to VDM and Reynolds' Method in the context of simulation as just defined.

- *Relations and Recursion* - definition of some technical machinery for the use of binary relations required for later chapters. Recursion is discussed in the context of the μ-calculus of Scott, deBakker, and Park.

- *Properties of Simulation* - study of the differences between the four types of simulation. Of particular importance are the differences that affect the concatenation of simulation diagrams. Data invariants are shown to be crucial for converting the partial abstraction relations to total relations. This result is required to show soundness of simulation.

- *Notation and Semantics* - notation for expressing data refinement is introduced. Interpretation functions are defined for predicates, abstraction relations, Hoare-style assertions, programs, relational terms, and correctness formulae are defined and discussed. Additionally, a toy programming language is defined along with its semantics for use in the later discussions on Hoare logic.

- *A Hoare Logic* - introduction and discussion of a Hoare logic with a minimal number of axioms for simplified soundness and completeness proofs. This is preparatory material for showing that data refinement proof obligations for a pair of operations (one abstract, one concrete) can be reduced to a Hoare triple.

- *Simulation and Hoare Logic* - core chapter of the first part of the book. For the four types of simulation, the authors show how two operations and an abstraction relation can be reduced to a single Hoare triple. The statement and proof of a normal form theorem for specifications is given for L-simulations.

- *An Extension to Total Correctness* - extension of the Hoare logic to total correctness. Reformulation of the theory is required since the prior partial correctness interpretation ignores non-terminating states (including blocked states). Complete partial orders are introduced to assist in dealing with the non-terminating states and the semantic interpretations given in chapter 5 are recast with the new framework.

- *Simulation and Total Correctness* - complications in using simulation arising from the introduction of the total correctness semantic model are dealt with by extending the soundness and completeness results to total correctness.

- *Refinement Calculus* - the essence of simulation, mapping state sets to other state sets is recast from the binary relations used in the prior chapters to predicate transformers. The connection between binary relations and predicate transformers is made and the soundness and correctness of predicate transformers is shown in both partial and total correctness settings.

Part 2 shows how the theory in Part 1 is embodied by commonly used methods of verifying software and includes

- *Reynolds' Method* - connection between Reynolds' four-point recipe for constructing concrete programs from abstract ones and simulation is shown using an extended directed graph example. Refinement in Reynolds' method is program refinement. The main result is that Reynolds' recipe is essentially L-simulation but with changes required to adapt his verification conditions for expressions and operations. The authors admit that they are not rigorous in proving their claim because there is some subjectivity required to interpret Reynolds' Method in the context of simulation, but this does not seem to affect the overall result.

- *VDM* - Vienna Data Method (VDM) is introduced through specification of a dictionary. The authors show that the data refinement steps in VDM imply L-simulation with total correctness. Refinement in the VDM is refinement between data types.

- *Z, Hehner's Method, and Back's Refinement Calculus* - three methods are presented in survey form to show the correspondence between them and data refinement. Z is shown to be equivalent to VDM when considered in the context of data refinement. Hehner's Method introduces the notion of data transformers which corresponds to a total L-simulation relation. Back's Refinement Calculus is based on predicate transformer semantics and when confined to conjunctive operations is shown to be equivalent to L-simulation under total correctness.

- *Refinement Methods Due to Abadi and Lamport and to Lynch* - refinement mappings are defined as total abstraction functions combined with the use of auxiliary variables such as history variables and prophecy variables. These refinement mappings are used by Abadi and Lamport to prove refinement between concurrent programs. The authors restrict themselves to non-deterministic sequential programs for showing the correspondence between the Abadi and Lamport methods and data refinement. Lynch's possibility mappings combine simulation with prophecy variables and these are shown to be equivalent to L-simulation after proving isomorphism between functions mapping concrete state spaces to abstract state spaces and relations mapping the same.

2 Opinion

The authors indicate that they use this book as a classroom text for advanced undergraduates and beginning graduate students and that a minimal knowledge of Hoare logic is required. My feeling is that the advanced undergraduate should have a high level of mathematical maturity to properly appreciate the material in the book. The exercises are numerous and of sufficient difficulty to provide a challenge to most students.

The pages separating Part 1 and Part 2 of the book contained photographs of some of the luminaries in the field of data refinement and simulation. The inclusion of these photographs with some historical text was a nice touch. When I am lost in the depths of details, it is easy to lose sight of the fact that the tools, techniques, and concepts used to simplify our understanding are not just sterile data, but the result of the hard-work and insight of gifted and tenacious people. Having an image to associate with a name in the bibliography makes the material come alive for me.

I truly enjoyed reading this book. One gets the sense of enthusiasm the authors have for the subject matter despite the density of theorems and proofs. For me, the proofs were very enlightening as they provided guidelines for how to go about conducting proofs in this subject. The heavy use of examples and the whole of Part 2 of the book provide the often neglected connection of theory with the real world. As a commercial software consultant, I can use the lessons provided in this book in my practice, either personally or directly for the benefit of my customers. I heartily recommend this book for anyone interested in any aspect of formal verification of software.

Response to Review of "Proofs and Refuations by Lakatos"
Original review by William Gasarch
Response by David Molnar

I agree with the "Opinion" section of the December 2001 review of Proofs and Refutations that philosophy of mathematics is important for theoretical computer science, but I do not believe that the reasons given are the strongest possible. The author [8] states that every computer scientist should have a "nodding acquaintaince" with the philosophy of mathematics, because "it is helpful to realize we have assumptions and what they are." Then the author cites as examples questions concerning alternative definitions for NP-completeness and zero-knowledge proofs. The implication seems to be that these are research questions that might be raised by exposure to philosophy of mathematics, therefore philosophy of mathematics is valuable to theoretical computer scientists as a potential source of research questions, especially concerning definitions. While philosophy of mathematics may be sufficient to raise such questions, it is not clear to me that it is necessary. We should look for contributions philosophy might make which would not come from computer science alone.

Both of the questions mentioned in the review have been investigated in some depth. I will address the second question.

The second question concerns parallel composition and zero-knowledge proofs, specifically the question of whether the definition of "zero-knowledge" should be amended to account for the fact that the parallel graph isomorphism protocol is neither known to "leak knowledge" nor is proved to be "zero-knowledge." This seems like a natural question, with an immediate practical motivation – executing all rounds in parallel increases the efficiency of the overall protocol, yet how do we know that this more efficient protocol will still be "secure?" Even without exposure to philosophy

[8] here and following, "the author" refers to the author of the review, William Gasarch, not to the author of the book, Imre Lakatos.

of mathematics, this is not a happy situation. Therefore I believe this question would have arisen even without the desired "nodding acquaintance."

Cryptographers responded in at least two ways to this problem (I apologize for omissions here). Feige and Shamir introduced a new definition of "witness-indistinguishable" protocols and showed that the parallel graph isomorphism protocol met this definition[1]. In fact they went further and showed that the class of witness-indistinguishable protocols was closed under parallel composition. Goldreich and Krawczyk, on the other hand, showed that the 3-round parallel graph isomorphism protocol could not be black-box zero knowledge, unless graph isomorphism is in BPP (which it is believed not to be)[2]. Both approaches test the limits of the definition of "zero-knowledge." Indeed, much recent work in cryptography concerns new definitions and testing the limits of old definitions. Were these results motivated by philosophy of mathematics or instead by concerns more "internal" to cryptography? Would they exist even if **Proofs and Refutations** had never been published?

Philosophy of mathematics might have more force in helping us interpret the results of existing research. **Proofs and Refutations** discusses "monster" counterexamples to theorems and the status of a theorem after such counterexamples are found. For another example from cryptography, Canetti, Goldreich, and Halevi showed that the "random oracle model" for proving protocols secure in cryptography has a disturbing feature[3]. There are cryptographic protocols which can be proved "secure" in the random oracle model yet provably have no secure implementation in the "real" world. At the same time, these protocols can be considered "monsters," very far removed from anything we would ever expect in practice. Then the question is "given this result, what does a proof in the model mean?"

In addition, philosophy of mathematics might help explain the phenomenon of equivalent definitions. The class of regular languages can be characterized by regular expressions, DFAs, logic, Boolean circuits, or by monoids (for a start). All these definitions seem to capture the same object. Why should this be so? We might also look to philosophy of mathematics to help us ajudicate between two incomparable definitions for the same intuitive concept. Each of these questions seems both of interest to theoretical computer science and closely connected to philosophy.

One final note: what is the place for "foundationalist" philosophy of mathematics in the "nodding acquaintance" every computer scientist should have? The review restricts itself only to the philosophy connected with the book under review, as is proper. I would be interested to know, however, the author's views on how widely this acquaintance should range.

[1] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In 22nd STOC, pages 416–426, 1990.

[2] O. Goldreich and H. Krawczyk On the Composition of Zero-Knowledge Proof Systems. SIAM Journal on Computing vol 25 number 1

[3] O. Goldreich and R. Canetti and S. Halevi. The Random Oracle Model, Revisited. In STOC '98 ACM Press 1998

Response to the Response
by William Gasarch

In David Molnar's response to my review he writes:

"While philosophy of mathematics may be sufficient to raise such questions, it is not clear to me that it is necessary."

I agree (so much for getting a controversial discussion going). However, I still maintain that Philosophy of Math is helpful and that Lakatos's book is a good place to learn some. The reason

why Lakatos's book in particular is a good place is that it focuses on one theorem in math (Euler's formula for polyhedra) which is hard enough to be interesting, easy enough to be understood, and has enough historical value to be enlightening.

David Molnar also writes:

"I would be interested to know, however, the author's views on how widely this acquaintance (with foundationlist issues in math) should range."

I think computer scientists (and mathematicians) should know enough about foundationalist issues to realize that the way math concepts are defined is not sacrosanct. For this purpose, a discussion of how a function should be defined (culminating in Dirchlet's notion which is now standard) would be enlightenting.