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

A website you should know about: the website www.bestbookbuys.com will, given a book, list LOTS of websites that are selling that book, in decreasing order of price.

Welcome to the Book Reviews Column. We hope to bring you at least two reviews of books every month. In this column nine books are reviewed.

1. The following four books are all reviewed together by William Gasarch. **Fair Division: From Cake Cutting to Dispute Resolution)** , by Brams and Taylor, **Cake Cutting: Be Fair if You Can** by Robertson and Webb, **The Win-Win Solution** by Brams and Taylor, and **Fair Allocation (Proceedings of Symposia in Applied Mathematics, Vol 33)** Edited by H. Peyton Young. These books all deal with the problem of fair division. Given a good or set of goods that you want to divide fairly, what is the best procedure? What is fair? What is best? These questions can be made rigorous and, so some extent, answered.

2. **Computational Geometry: Algorithms and Applications (2nd Ed)** by Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. reviewed by Hassan Masum. This book can be used as both a textbook for a graduate course or as a refresher for someone who does not work in the area.

3. **Parameterized Complexity** , by Downey and Fellows. and is reviewed by Gabriel Istrate. Many NP-complete problems have as input a graph G and a parameter k. If k is fixed one can often easily obtain an algorithm of complexity $n^{O(k)}$. There are some problems where where you can obtain an algorithm of complexity $f(k)n^a$ where a is some constant and f is some function. Such problems are 'easy' even though they are NP-complete. This book is mostly about such problems, but also has some material on a formalism to show that some problems are not of this type.

4. **Modern Graph Theory** by Bollobás. Reviewed by E. W. Čenek. This is a somewhat advanced book on graph theory that requires some mathematical maturity. It is on pure graph theory with few applications.

5. **A=B** by Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger. Reviewed by Vladik Kreinovich. Recall that Exercise 1.2.6.63 of Knuth Volume 1 is *Develop computer programs for simplifying sums that involve binomial coefficients.* This book solves the problem completely. The math is stunning, the algorithms actually work, and code is available in MAPLE or MATHEMATICA.

6. **Communicating and mobile systems: the π-calculus** by Robin Milner. Reviewed by Riccardo Pucella. This book introduces a formalization of semantics for concurrent systems.

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 can be a great way to learn a field. I have personally reviewed books and then went on to use what I learned in my research.

1. *Computational and Geometric Aspects of Modern Algebra* Edited by Atkinson, Gilbert, Howie, Linton, and Robertson. (Proceedings of a workshop)
2. *The Discrepancy Method* by Chazelle.
3. *Control Flow Semantics* by Jaco de Bakker and Erik de Vink
4. *The Clausal Theory of Types* by Wolfram.
5. *Model Checking* by Clarke, Grumberg, and Peled.
6. *Introduction to Process Algebra* by Fokkink.
7. *Automata and Languages: Theory and Application* by Alexander Meduna.

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. Networks in Distributed Computing.
4. Advances in Switching Networks.
5. External Memory Algorithms.
6. Mobile Networks and Computing.
7. Robust Communication Networks: Interconnection and Survivability.

Reviews [2] of THREE books on Fair Division of Resources

Fair Division (From Cake Cutting to Dispute Resolution)
by Steven Brams and Alan Taylor
Published by Cambridge University Press, 272 pages, 1996
Softcover, $21.95
ISBN number 0-521 R5539-0

AND

Cake-Cutting Algorithms(Be Fair if You Can)
by Jack Robertson and William Webb
Published by A.K. Peters, 177 pages, 1998
Hardcover, $38.00
ISBN number 1-5688-10766

AND

[2] © William Gasarch 2000

The Win-Win Solution
by Steven Brams and Alan Taylor
Published by Norton, 177 pages, 1999
Softcover, $13.95
ISBN number 0-393-04729-6

AND

Fair Allocation
Proceedings of Symposia in Applied Mathematics, Vol 33
Edited by H. Peyton Young
Published by AMS, 1986,170 pages
Softcover, $31.00
ISBN number 0-8218-0094-9

Reviews by

William Gasarch
University of Maryland At College Park
gasarch@cs.umd.edu

1 Overview

The following problem arises in many real world situations: How do you divide a quantity among n people in a fair manner? To make this question rigorous we have to know more about the item, the people, and the notion of fairness. The following questions need to be raised.

1. Is the quantity continuous (like a cake) or discrete (like a set of items in an inheritance)?
2. What is each players valuation? (For example Bill values the chocolate part of the cake.)
3. What do the players know about the other players valuations?
4. If a player thinks (in his measure of value) that he got $\geq \frac{1}{n}$ of the total value, but that someone else got more than he got, is that fair? (Say Bob, Carol, Ted, and Alice split a cake such that Bob thinks he got $\frac{1}{3}$, but he also thinks that Carol got $\frac{1}{12}$, Ted got $\frac{1}{12}$, and Alice got $\frac{1}{2}$. All the other players think that they each got $\frac{1}{4}$. Is this fair?)
5. Is there a trusted referee?
6. If the item is continuous then can we use continuous methods? For example a knife could be passed over the cake until someone yells STOP, and whoever yells STOP gets the piece.
7. If the item is actually a set of discrete items then what methods are allowed? Auctions? Voting?
8. If the method being used is discrete then how many operations are allowed? For example, we might want to minimize the number of cuts when dividing a cake.
9. Is the item something people want like cake or something people do not want like chores?

10. Is an approximate solution good enough? How approximate?

11. What if the players want unequal shares? For example both Alice and Bob can agree that they want Alice to get $\frac{2}{3}$ and Bob to get $\frac{1}{3}$.

12. Given a way to divide the item(s) is there another division where at least one player is better off and nobody else is worse off?

13. Assuming that all parties know each others valuations, do all parties think that they got the higher (or tied) fraction of goods then the other parties. For example, if after a divorce Alice thinks she god 60% of the estate, Bob thinks he got 90% of the estate, and Alice knows how Bob's feels, this might not be considered fair.

14. Is the protocol easy to use?

The three books under review look at mathematically rigorous versions of some of these problems mentioned above. To give a flavor for the subject we present two definitions and two protocols.

A division of a good among n people is *proportional* if each person thinks they got at least $\frac{1}{n}$. A division of a good among n people is *envy-free* if each person believes his piece is at least as big as everyone elses.

Here is a discrete envy-free protocol among Bob, Carol, and Alice to divide a cake in three pieces. It is due independently to Selfridge (1960) and Conway (1993)[3] As is the convention we write the protocol and then put in parenthesis what the player should do in his or her self interest. A phrase like "Bob takes a piece" actually means "Bob takes the piece that is largest in his valuation."

1. Bob cuts the cake into three pieces. (Equal pieces.)

2. Carol trims at most one piece. The trimming is set aside and called T. (Carol trims the unique largest piece, if it exists, to create a tie for largest.)

3. Alice takes a piece.

4. If Alice did not take the trimmed piece then Carol must take it. If Alice did take the trimmed piece then Carol can take whichever of the two pieces left that she wants.

5. Bob gets the piece that is left. Note that the cake minus T has been split in an envy-free way.

6. Assume that Carol got the trimmed piece (if Alice did then the procedure is similar). Note that if Carol gets all or part of the trimming T then Bob does not mind since he thinks Carol's piece plus T is equal to his piece. Alice cuts T into three (equal) pieces.

7. Carol takes a piece.

8. Bob takes a piece. Note that Bob does not mind if Carol got a bigger piece of T then he did.

9. Alice gets the remaining piece.

[3]The books under review reference Selfridge and Conway as I did above; hence I assume there results were never formally published.

It is not hard to show that the above protocol is envy-free. Note that it takes at most five cuts. There is a discrete envy-free protocol for $n = 4$ and even for general n; however, the number of cuts they take depends on the players valuations of the cake and is finite but unbounded. For example, there is a valuation that the four players could have so that 10^{100} cuts are needed. It is an open problem to find a bounded protocol for envy-free division with $n = 4$.

We now present a continuous proportional protocol that works for general n but is not envy-free.

1. The knife is passed over the cake left to right until some player yells STOP. (That player thinks the piece between the left end and the knife is $1/n$.)

2. The player who yells stop gets the piece between the knife and the left end. (Ties are decided arbitrarily.) He should be happy since he thinks its $1/n$ of the cake. The rest of the players should be happy because since they didn't yell STOP they thing that at least $1 - 1/n$ is left.

3. Repeat the protocol with the $n - 1$ players left and with the cake that is left.

2 Fair Division (From Cake Cutting to Dispute Resolution)

This book deals with both the continuous and the discrete case. Both proportional and envy-free solutions are considered. Both discrete and moving-knife solutions are considered. Real world examples are discussed (e.g., negotiating the Panama Canal Treaty). In addition there are full chapters on voting, auctions, and the divide-the-dollar game (how to get two parties to divide a dollar fairly).

The book is well written and would be appropriate for an interdisciplinary undergraduate course. Students from mathematics, computer science, economics, and sociology would be the audience. No particular math background is needed; however, mathematics maturity is required. There are no exercises, which might limit its usefulness as a textbook for a standard HW-exam course; however, there are many references to the Economics and Political Science literature, which would allow for a course where students write papers.

The book is well organized. Most of the chapters are on a well defined topic (e.g., "Envy-free division for $n \geq 3$"). There are many footnotes which allow for easy personal reading since you can choose which footnotes to skip.

3 Cake-Cutting Algorithms (Be Fair if You Can)

This book deals mostly with the continuous case, i.e., cake cutting. They assume that the players do not know each others valuations. Both proportional and envy-free solutions are considered. Both discrete and moving-knife solutions are considered. Upper and lower bounds on the number of cuts are also discussed. Some mathematics of interest is used: Ramsey partitions and bipartite graph matching.

The book is well written and would be appropriate for an interdisciplinary undergraduate course. Each chapter includes exercises. Hence this could be a real course with HW and exams. Students from mathematics, computer science, economics, and sociology would be the audience. No particular math background is needed; however, mathematics maturity is required.

In the preface the author states "The first six chapters provide a leisurely survey of the problems, Chapter 7 is a quick reference which summarizes known results, while the last four chapters contain technical proofs for previously introduced problems." This format is good for students and for

planning a course. The first seven chapters could form a course. A more mature class could do the first seven chapters and some subset of $\{8, 9, 10, 11\}$. For your own personal reading it can be inconvenient to flip back and fourth many times..

4 Differences between *Fair Division* and *Cake Cutting*

The following topics are in *Fair Division* but not in *Cake Cutting*.

1. Dividing a set of discrete objects (Chapters 3,5).
2. Dividing a dollar (Chapter 8).
3. Auctions (Chapter 9).
4. Voting (Chapter 10).
5. What if a player knows the other one's valuation. (Section 1.4. There is not that much rigorous to say.)
6. Applications. (Parts of Chapters 1 and 4, but also scattered throughout.)

The following topics are in *Cake Cutting* but not in *Fair Division*

1. The number of cuts needed to divide a cake proportionally (Chapters 2,8,9).
2. Unequal shares. This uses Ramsey Partitions. (Chapters 3, 11).
3. Exact-fair division where everyone thinks they have *exactly* $1/n$. (Parts of Chapters 5 and 10.)
4. Dividing up chores (Section 5.5).
5. An application of Hall's theorem (Chapter 6 which is short).
6. Exercises and suggested projects.

5 The Win-Win Solution

This book is for the layperson; however, readers such as ourselves can still benefit from it. This book looks at a few schemes for dividing a set of discrete goods among two people and then applies them to very real world scenario. The Camp David accords, the (first) Trump divorce, the Spately Island disupte, the debate over the Clinton-Dole debates, and the Princess Di divore are all discussed in detail. Other disputes are discussed in passing. By applying a technique called *Adjusted Winner* one obtains a fair settlement which seems to match the real settelments reached. One catch is that the participants have to know how much they value various goods and report on this honestly. The estimates given in the book seem reasonable.

The main technique discussed is Adjusted Winner. Each particpant intially (and privately) distributes 100 points over all the goods available. Then in phase one the goods are distributed (roughly) based on who wants what more. A second phase transfers goods from one participant to another so that each participant gets at least 50 points in his estimation.

This book has much discussion of how to try to take advantage of knowledge of your opponents valuation. They conclude that using this knowledge to mispresent yourself for an advantage could is very dangerous— you might get far less than you would being honest.

6 Fair Allocation

This book is a collection of six self-contained articles on fairness issues that arise in politics and public policy. Each article begins with a real world problem, models it mathematically, and discusses solutions. Real data is presented. Each article is well written though a bit dense. A good senior math major or theory-inclined computer science major could read it. The authors do not have any political viewpoint to push which, while not surprising, is refreshing during this election year. This book was published in 1986. For a survey of some newer results on this topic see [1].

The articles tend to have more definitions and examples then theorems. This is probably caused by the clash of clean mathematics with the dirty world of politics. The book is from 1985 hence there has been some new material on some of these issues since then. We briefly describe each chapter by describing the problem they deal with. We could end each review with the line "This article survey's known solutions and the problems they may have."

6.1 The Apportionment Problem

The U.S. Constitution dictates that the number of representatives a state sends to the house or representatives is proportional to its population. This can lead to a fractional number of delegates ("The 3.41 of the New York delegates voted NO.") While it would seem that rounding off would solve the problem, this may lead to problems.

6.2 Inequality Measures

Politicians often say things like "That tax plan will further widen the gap between rich and poor." Such statements are not rigorous. How do you measure the gap?

6.3 Cost Allocation

If several towns are going to cooperate and build a common water treatment plant then what is the best way to divide the cost? There are many factors such as population and the cost savings achieved by pooling the efforts.

6.4 The Allocation of Debts and Taxes

When dividing an estate among several people what is the fairest allocation? The people may have different rights to the estate. For example perhaps the estate should be split $(1/2, 1/6, 1/12, 1/12)$. If the estate is small you may not want to invoke those rights since it could leave someone with (essentially) nothing. But this leads to another question— what is "small"?

6.5 Voting

If there are two candidates competing for an office then having a vote and having the majority candidate win is fine. If you have three candidates then several problems arise: (1) It is possible that no candidate gets the majority. (2) People may not vote honestly. For example one often hears "I want to vote for Harry Browne (Libertarian Party) but I know he can't win so instead I'll vote for George W. Bush." (3) If people rank their choices you may have that a majority prefer A to B, a majority prefer B to C and a majority prefer C to A. So what is the best scheme? A well known theorem by Arrow shows that you cannot have an voting system that satisfies a (reasonable) set of criteria. However, there are systems with various pros and cons that are worth looking at.

6.6 Auctions and Competitive Bidding

There are many types of auctions and many types of criteria for fairness. For example, lets say we want to maximize profit for the auctioneer. We might want to use a second-price auction where the winner pays the second highest bid. This might increase profit since the bidders can be more aggressive.

7 Opinion

I recommend both *Fair Division* and *Cake Cutting* for a course on this topic. You can reread the section of this review on the differences to decide which one is right for you. Whichever one you use as a text I recommend buying the other one for yourself. I also recommend *Fair Allocation* for personal reading, though not as a text. I recommend *The Win-Win Solution* as background for a course or to give to a friend or relative who is interested in these issues but not in the mathematics.

Above all else I recommend that theoretical computer scientists take an interest in this area. The problems studied are interesting and use the kind of mathematics we are familiar with (discrete math).

References

[1] E. Hemaspaandra and L. Hemaspaandra. Computational politics: Electoral systems. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 64–83. Springer-Verlag *Lecture Notes in Computer Science #1893*, August/September 2000.

Review of *Computational Geometry: Algorithms and Applications (2nd Ed)* [4]
Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf
Publisher: Springer, 2000
Hardcover, $34.95
ISBN: 3-540-65620-0

Review by:
Hassan Masum
Carleton University, Ottawa, Canada
hmasum@ccs.carleton.ca

1 Overview

Computational Geometry is a wide-ranging introductory text which exposes readers to the main themes in modern computational geometry. Each chapter introduces a subfield of computational geometry, via natural problems and basic algorithms; exercises and notes help to flesh out the chapter material. This second edition of the book is obviously the product of much effort by the authors, and although some improvements are possible, on the whole this book is worth considering both as a text for a first computational geometry course and as a refresher on basic concepts.

[4] Hassan Masum 2000

Features of interest include:

Beginning each chapter with a motivating real-world example, to naturally introduce the algorithms. The solution of this example leads to the key algorithmic idea of the chapter.

Emphasis on derivation of algorithms, as opposed to a cookbook-style presentation. The authors often spend a large amount of time to work through several suboptimal solutions for a problem before presenting the final one. While not suitable for the already-knowledgeable practitioner, this gives the far-larger category of students or other less-than-expert readers training in the process of generating new algorithms.

Good layout, with wide margins containing intuition-generating diagrams. The Notes and Comments section at the end of each chapter also provides useful orientation to further algorithms and context in each subfield.

Wide coverage of algorithms. As the authors say: "In general we have chosen the solution that is most easy to understand and implement. This is not necessarily the most efficient solution. We also took care that the book contains a good mixture of techniques like divide-and-conquer, plane sweep, and randomized algorithms. We decided not to treat all sorts of variations to the problems; we felt it is more important to introduce all main topics in computational geometry than to give more detailed information about a smaller number of topics."

2 Background and Contents

Computational Geometry studies algorithms and data structures for solving geometrical problems. Since many application domains such as GIS (Geographic Information Systems) and computer graphics are often concerned with very large problem instances, the traditional focus is on algorithms with sub-quadratic time performance. External-memory algorithms which minimize I/O operations have also become increasingly important.

Canonical operations on point sets include Delaunay triangulation (finding the triangulation which maximizes the minimum angle), convex hull generation (gift-wrapping the points), and Voronoi diagram generation (partitioning space into regions closest to each point). In two dimensions all of these can be solved with straightforward algorithms in O(N log N) time for an N-point set. Another common theme in computational geometry is generation of auxiliary data structures to speed up operations, such as kd-trees for orthogonal range searching and BSP trees for efficient 3D graphics.

Usually the algorithms are targeted to two or three dimensional Euclidean space. In higher dimensions, the time and conceptual complexity can drastically increase; examples are convex hulls, clustering, and polytope volume calculation. (Aside from the coolness factor, higher-dimensional geometry is useful in surprisingly many applications, such as database range queries and clustering in data analysis.)

The chapter contents are as follows:

9 Delaunay Triangulations
10 More Geometric Data Structures (interval, priority search and segment trees)
11 Convex Hulls
12 Binary Space Partitions
13 Robot Motion Planning
14 Quadtrees
15 Visibility Graphs
16 Simplex Range Searching
Bibliography and Index

3 Opinion

This book is well written and enjoyable to read. It would benefit from an additional higher-level grouping of chapters, by a criterion such as data structure type or application domain; currently the 16 chapters are not arranged in any obvious way, which makes it hard for the novice to conceptually organize the material.

A little more depth would be nice; some chapters seem a bit shallow, with just the simplest algorithm for one problem in a field presented. Algorithmic modifications for large-scale and online computational geometry are lacking, as is discussion of the utility and general usage of geometry subroutine libraries like LEDA (which form an essential part of the practical usage of computational geometry for most students).

An applications chapter at the end would be valuable to many readers. This could survey the main categories of computational geometry applications, and show how to pull together the individually-presented algorithms of previous chapters to solve all aspects of a problem. It would also give a unified overview of the main application domains, as opposed to the small sampling currently given in section 1.3. Answers or at least hints for some of the end-of-chapter problems should also be given in the text, as a pedagogical aid for the reader.

The Web site for the book (www.cs.uu.nl/geobook/) is quite sparse but contains some useful information. Good Web resources for geometry ideas are Geometry In Action and Geometry Junkyard, both available off David Eppstein's site (www.ics.uci.edu/ eppstein/).

While more than satisfactory as an introductory text, this book is somewhat lacking in depth and would need supplements for use as a graduate text or self-study resource. Nevertheless, *Computational Geometry* is worthy of strong consideration as a classroom text or as a pleasant guided introduction to the field.

Review of
Parameterized Complexity[¶]
Authors: R. Downey and M. Fellows
Publisher: Springer Verlag
Hardcover,xv+533 pages, DM 110.
ISBN 0-387-94883-X

Reviewer: Gabriel Istrate
Center For Nonlinear Sciences
Los Alamos National Laboratory
MS B258, Los Alamos, NM 87545, U.S.A.
istrate@lanl.gov

1 Overview

One of the most pervasive critiques of Computational Complexity, coming from the applied Computer Science community, is that it is an abstract, largely irrelevant field, that fails to capture the various intuitions concerning the complexity of "real-world" instances they have gathered through experiments. One aspect of these intuitions is that while most problems of practical interest tend to be intractable from a standard complexity-theoretic point of view, in many cases such problems have natural "structural" parameters, and practically relevant instances are often associated with "small" values of these parameters.

Parameterized Complexity is an extension of Complexity Theory that attempts to formalize and deal with this intuition. It has already been applied to fields such as Databases, Logic and Constraint Programming, Computational Biology, Nonmonotonic Reasoning, Natural Language Processing, Cognitive Science, and Robotics. Some of its ideas have even been discovered independently by practitioners working in these areas. We do not attempt a further presentation (the interested reader can consult two excellent surveys [DF99, DFS99]), but want to illustrate this point through a revealing example: it has long been known that the register-allocation problem for imperative programming language amounts to a coloring problem on the associated control-flow graph, thus it seems, at a first glance, hard even in an approximate sense. On the other hand Thorup [Th98] has shown that such graphs are characterized by a bounded value (at most 10) of an intrinsic parameter called *treewidth*, that guarantees the existence of an approximation algorithm with substantially better performance than the one available for general graphs.

The present book is an attempt to a first presentation of this emerging area by two of its most active proponents.

2 Summary of Contents

The book consists of nineteen chapters of significantly varying lengths. Chapter 1 is a readable introduction to the field. The remaining Chapters are grouped (fairly loosely) into three sections:

Parameterized Tractability, *Parameterized Intractability* and *Structural and Other Results*. The book also contains two Appendices, containing a compendium of parameterized problems and their classification and a number of open topics and research directions.

Chapter 2 introduces the basic framework of fixed-parameter tractability (FPT), and relates it to the advice view of nonuniform complexity, a theme revisited in Chapter 5. Chapters 3, 6, 7 and 8 present various techniques for establishing fixed parameter tractability. Extremely rewarding are Chapters 6 and 7, that deal with the concept of graphs of bounded treewidth (in particular Courcelle's theorem on the polynomial time decidability of monadic second-order graph properties on such graphs), and with the results of Robertson and Seymour on graph minors, respectively. Chapter 4 presents the connection between fixed-parameter tractability and the approximability of optimization problems.

Chapters 9 to 15 are devoted to the parameterized counterpart of the theory of NP-completeness. Parameterized complexity is more subtle than its classical version, in that there are NP-complete problems that are *not* equivalent in the parameterized sense, and give raise to a hierarchy (called *the W–hierarchy*) based on reducibility to circuit classes of bounded "weft". Chapter 9 introduces the notion of parameterized reduction. The first level of the hierarchy, $W[1]$, is defined in Chapter 10, where a parameterized analog of Cook's theorem is also proved, namely that CNF-k-satisfiability, parameterized by the number of ones in a desired solution, is complete for for $W[1]$. A number of other problems complete for $W[1]$ are presented in Chapter 11. The general W–hierarchy is discussed in the next chapter. The following three chapters present a number of parameterized complexity classes of higher complexity, denoted $W[SAT]$, $W[P]$, that are analogs of the finite levels of the polynomial hierarchy and of $PSPACE$, and a class XP whose complete problems are *provably* not in FPT.

The last four chapters in the book have a slightly narrower, more technical, perspective. Chapter 16 presents quantified-based characterizations for the various parameterized complexity classes, and the impact of such results on the (non)collapse of the W–hierarchy. Chapter 17 explores the connection to classical models of limited nondeteminism and the "guess and check" complexity classes of Cai and Chen. Chapter 18 deals with the collapse of monotone (and antimonotone) versions of the W–hierarchy. The book concludes with a recursion-theoretic discussion of the difference between the arithmetical complexities of classical and parameterized reductions. For this chapter familiarity with (part of) Soare's book [So87] would be extremely helpful.

3 Opinion

The main problem with the book might not have to do with its contents but rather its physical appearance: I have seen now three copies of the book, all of them used only sparingly, and in all three cases the binding seems to have deteriorated substantially more than expected. Granted, a sample of this size is extremely small, but a word of caution might still be in order. I have also been somewhat frustrated (especially when reading Chapter 6) by the number of typographic errors.

As to the contents: as the authors acknowledge in a particular context[1], Parameterized Complexity has aspects that can seem fairly idiosyncratic to many computer scientists, even of theoretical formation. Also, due to obvious space constraints the book could not (and does not) do full justice to the various practical applications. We, nevertheless, believe that the authors have done a great job in making a strong case for this research direction, and in presenting the most important

[1] "It has recently been reported that 9 out of 10 classically trained complexity theorists regard the $W[t]$ hierarchy as the *Weirdness Hierarchy* ...", pp. 486

results of the field in an unified, clear and lucid manner. Chapter 1 is eloquent and extremely enjoyable, and together with overviews [DF99, DFS99] can serve as an excellent introduction to the area. The topics treated in the book will appeal to people with varying interests: Chapters 3, 8 and portions of Chapters 6 and 7 to people interested in algorithmics, Chapters 4, 5, 9, 10, 11, 15 to structural complexity theorists, while Chapter 19, 7 and (partly) Chapter 6 will be most rewarding to people interested in logic, recursion theory and foundations.

The book can be read sequentially and is essentially self-contained. Several portions of it require a significant degree of mathematical maturity. We feel that a most profitable first course would involve Chapters 1, 2, 3, 4, 9, 10, 17 and most of Chapter 6. Chapter 7 is an especially valuable reading, and should be definitely a choice for all in–depth studies. The book is accessible to graduate students (or advanced undergraduates) and could also be a valuable tool for in more theoretical flavors of Artificial Intelligence. A second edition could probably include several recent results (such as those of Gottlob et al. on the complexity of acyclic database queries, or tree decompositions).

References

[DF99] R.G. Downey and M. Fellows. Parameterized Complexity after (almost) 10 years: Review and Open Questions. In *Combinatorics, Computation and Logic*, Springer Velag 1999.

[DFS99] R.G. Downey, M.R. Fellows, and U. Stege. Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 49, R. Graham, J. Kratochvil, J. Nesetril and F.S. Roberts (eds.), pp. 49–99, 1999.

[So87] R. Soare. Recursively Enumerable Sets and Degrees: a Study of Computable Functions and Computably Enumerable Sets. Springer-Verlag, 1987.

[Th98] M. Thorup. All Structured Programs have Small Tree-Width and Good Register Allocation. *Information and Computation* 142(2): 159–181 (1998).

Review of: **Modern Graph Theory** [2]
by Author: Béla Bollobás
Publisher: Springer-Verlag

Reviewed by E. W. Čenek,
University of Waterloo,
Dept of Computer Science
ewcenek@hopper.math.uwaterloo.ca

1 Overview

Graph Theory is still a relatively young subject, and debate still rages on what material constitutes the core results that any introductory text should include. Bollobás has chosen to introduce graph theory - including recent results - in a way that emphasizes the connections between (for example) the Tutte polynomial of a graph, the partition functions of theoretical physics, and the new knot polynomials, all of which are interconnected.

On the other hand, graph theory is also rooted strongly in computing science, where it is applied to many different problems; Bollobás's treatment is completely theoretical and does not address these applications. Or, in more practical terms, he is concerned whether a solution exists, rather than asking whether the solution can be computed in a reasonably efficient manner.

One of the pleasures[3] of working in graph theory is the abundance of problems available to solve. Unlike many more traditional areas of Mathematics, knowing the core results and proofs is frequently insufficient. Often solving a new problem requires a new approach, or a subtle twist on an existing one, combined with some bare knuckle work. Bollobás emphasizes this in the problems available at the end of each chapter; he includes in total 639 problems, ranging from the reasonably straightforward to the very difficult. I spent time with friends working on these problems, and was intrigued by the variety of the proofs that we came up with.

2 Content Summary

Bollobás's text is written for the mature mathematics student, who is capable of reading at a reasonably fast pace. The text contains ten chapters, each of which is reasonably self sufficient, except for there dependence on chapter 1.

Each chapter consists of, on average, about five sections, plus the problems. Within the chapter, the pace escalates from section to section; the proofs and motivations given in the first section are given in detail, and are easy to follow, and the pace speeds up from section to section, as readers are drawn deeper into the subject. The following summary describes, in very brief detail, what each chapter covers.

2.1 Fundamentals

An introduction of the basic definitions and notations used throughout the book, including introductions to paths, cycles, trees, planar graphs, and Euler paths. The last section includes an application of Euler trails to algebra.

2.2 Electrical Networks

This section is motivated by the study of electrical networks. If every wire is an edge, every connection between two or more wires is a vertex, and every edge is weighted depending on the resistance of the wire, then the result is a weighted multigraph describing the electrical network. Given the graph it is possible to describe the flow of electrical current. Vector spaces and matrices are used.

2.3 Flows, Connectivity, and Matching

In studying maximum flow in a weighted directed graph, each edge has a flow capacity, and the flow is the amount of current[4] that can flow from the *source* to the *sink*, where each intermediate vertex has no more current coming out than it does coming in. From this, properties in graph connectivity and matching are developed.

[3] Or curses, if you have just spent weeks, months, years pursuing what turns out to be a very uninteresting solution

[4] Or, in telecommunications, bits of information

2.4 Extremal Problems

Extremal problems concern the maximum and minimum parameters of a graph, as well as the specific graphs for which these maximums and minimums are attained. For instance, what is the smallest and largest cycles in the graph (the *girth* and *circumference* respectively. Or what are the forbidden subgraphs of a certain class of graphs.

2.5 Colouring

A graph can be coloured using k colours by assign one of $1 \dots k$ to each vertex, so that no two adjacent vertices have the same colour. Alternatively, the edges can be coloured so that two edges that share an endpoint have different colours. Sample theorem: if G is a perfect graph whose largest clique has k vertices, then the vertices of G can be colored with k colors.

2.6 Ramsey Theory

Dirichlet introduced the Pigeon-hole Principle, which guarantees that for any n classes, if we have m objects, $n < m$, then at least one of the classes will contain more than one object. Ramsey extends this principle so that we are not looking merely for multiple objects in one class, but a large substructure. For instance, if the edges of a graph with an infinite set of vertices N are each coloured red or blue, then for some infinite set $M \subseteq N$ all the edges joining vertices of M get the same colour. If N is, instead, finite, how large does N have to be to guarantee the existence of a set M of size k?

2.7 Random Graphs

What does the *average* graph in a class of graphs look like? What properties does the average graph have? A random graph with n vertices and m edges actually consists of a probability space consisting of all graphs with n vertices and m edges, each being equiprobable. For instance, if m is sufficiently large relative to n, almost all graphs have a Hamiltonian cycle.

2.8 Graphs, Groups and Matrices

This chapter is a brief introduction to algebraic graph theory, and concentrates on the interplay between graphs and groups, and the use of matrix methods. A graph can be represented in and $n \times n$ $[0, 1]$ matrix M where $M_{ij} = 1$ if and only if vertices i and j are adjacent. The eigenvalues of this matrix are linked with certain basic properties of the graph, such as its maximal degree, and whether the graph is bipartite.

2.9 Random Walks on Graphs

A random walk on a graph is exactly what the name implies; a walk from vertex to vertex through the graph, picking the next edge at random at each vertex. Finite Markov chains are random walks on weighted directed graphs, with loops allowed. A connection between electrical networks and random walks is established, and the parameters of random walks are considered.

2.10 The Tutte Polynomial

The Tutte polynomial, developed in 1954, reduces a graph to a polynomial, created recursively using *cut* and *fuse* operations. This polynomial contains information about the graph, and loses

less information than the characteristic polynomial from Chapter 8. Polynomials can also be used to describe knots, including the Jones and Kauffman polynomials which were discovered in the 1980's. Using these polynomials, which are closely related to Tutte's polynomial, one can prove such things as the inequivalence of two knots. This chapter is very introductory, and only begins to scratch the surface, as Bollobás notes.

3 Opinion

As an introduction to the mathematical study of graph theory, I enjoyed the book. The pacing early in each chapter was slow for me, but it sped up before the pacing got tedious, and the book made pleasant evening reading, as well as good discussion fodder. The problems were enjoyable and challenging, and many of the proofs were interesting, including some that were positively lyrical. A grounding in the different fields of mathematics that Bollobás discusses is not absolutely necessary, since he takes care to provide the relevant definitions, although comfort with these other fields makes for easier reading. The mathematical grounding is useful for students of mathematics and computing science alike, although extra reading will be required for students of computing science to get a complete overview.

On the other hand, I regretted the lack of practical considerations. A frequent problem in computing science asks not whether a solution exists, but whether the solution - if it exists - can be found in a reasonable amount of time. Graph theory can be used, and applied, in many fields in both mathematics, and computer science, and as such makes a wonderful bridging of the two fields. Graph theory is rapidly maturing in computing science, too, but a computing science student may not find this book a practical introduction to graph theory, considering it completely fails to address NP-completeness. For example, in chapter 1, Bollobás introduces Hamiltonian cycles, and proves how many edge disjoint Hamiltonian cycles a complete graph can be decomposed into. However, he does not mention that in the more general case, finding a Hamiltonian cycle for any graph is NP-complete.

On the other hand, the proofs and problems are presented in a way that encourages more reading on the subject. Some of the chapters, especially Chapter 10, introduce concepts without attaining much depth, providing just enough background information to whet one's appetite.

Review of $A = B^5$
by Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger
Published by A.K. Peters, Ltd., Wellesley, MA, 1996, 212 pages
Hardcover, ISBN Number 1568810636

Reviewer: Vladik Kreinovich
University of Texas at El Paso
Department of Computer Science

Often, the problem of finding the computational complexity of a certain algorithm can be reduced to computing a combinatorial sum – i.e., a sum like this:

$$\sum_k \binom{2n-2k}{n-k}\binom{2k}{k}. \tag{1}$$

[5]

This is often true when we are interested in the worst-case complexity, and this is even more frequently true when we are looking for the average-case complexity. There are so many combinatorial sums in these problems that even writing them down is difficult: suffice it to say that Donald E. Knuth has spent so much time making these formulas look right in his famous *The Art of Computer Programming* books that after Volume 3, he decided to take a break from these books and develop an automated way to type such formulas – an effort which eventually led to TEX and its dialects.

Even teaching a computer to *write down* all these formulas is so tough a task that it requires a multi-year effort – well, teaching a computer to *derive* these formulas is even a greater challenge. Knuth himself formulated this challenge in the very first volume (*Fundamental Algorithms*) of his book, as Exercise 1.2.6.63: *Develop computer programs for simplifying sums that involve binomial coefficients*. Knuth gave this problem the rating of 50 – meaning that it is one of the toughest problems in the book. It was tough. For several decades, it avoided solution. The good news is that after 30 years of extraordinary efforts (largely the efforts of this book's authors), this problem is largely solved: there are algorithms, there are effective computer programs which make handling the cumbersome combinatorial sums unbelievably easy. This book, proudly starting with the formulation of the original Knuth's problem and with an enthusiastic preface by Knuth himself, is a culmination of these efforts. This book is about the problem, the history of its solution, the resulting algorithms, and finally – about the programs.

Actually, there are *two* problems here. The *original problem* is when we have an expression for the sum, and we would like to have a more compact expression for it. Usually, when we cannot immediately find an answer, a natural idea is to compute this sum for $n = 0, 1, 2, \ldots$, and try to guess the answer. Very often, the answer is easy to guess, and then we run into a *second problem:* to prove that this guess is indeed a correct formula for all n. For example, for the above sum (1), Knuth computed it for $n = 0, 1, 2, 3$, and came up with the values $1, 4, 16, 64$. A natural guess is that for every n, the desired sum is equal to 4^n, but proving that this sum is indeed always equal to 4^n turned out to be hard.

In general, the second problem is as follows: given two combinatorial expressions A and B, check whether they always coincide, i.e., whether $A = B$ (this is where the unusual book title comes from).

One of the main reasons why these problems turned out to be so hard is that not only did we not have efficient *programs* for checking such equalities, but researchers were not even sure that this problem was algorithmically solvable at all. Indeed, we want to check formulas of the type $\forall n(A(n) = B(n))$, with a universal quantifier running over all natural numbers. For computable functions $A(n)$ and $B(n)$, checking such formulas is equivalent to solving a halting problem – an algorithmically undecidable task.

However, in spite of the fact that checking such equalities is impossible for *general* computable functions, Knuth believed that it may be possible for a *specific* class of computable functions – functions which contain sums of combinatorial expressions. This hope and optimism came naturally from decades of frustration. This phrase sounds somewhat paradoxical, so let me clarify it. Few (maybe none) computer science theoreticians are fond of handling combinatorial sums (which appear again and again in complexity problems). These sums are challenging, tough, often require a lot of effort – sometimes as much (or even more) effort than the computer science part of the problem – but the reward and recognition for solving a combinatorial problem is usually much-much smaller than for solving the computer science problem. Why? Because the solution is usually very technical, rather short, not very exciting, usually – a clumsy complicated combination of simple ideas and transformations rather than than a stroke of genius. This combinatorial part is very frustrating, but in this very frustration lies hope: since all these sums seem to be computed

by using the same ideas, maybe these ideas are indeed sufficient, and a computer can take care of finding the appropriate combination of these ideas? This hope turned out to be true.

The history of the resulting algorithms starts with Carl Friedrich Gauss (1755-1837), a genius who was recognized, in his time, as the King of Mathematicians. Probably everyone heard the (apocryphal) story how a ten-year old Gauss, when asked to find the sum $1 + 2 + \ldots + 100$, came up with a general formula $1 + 2 + \ldots + n = n(n+1)/2$. The trick that he used to design this formula – grouping together the first and the last term, the second and the second from last, etc. – can actually help to add up an arbitrary arithmetic progression. It is less known (and true) that Gauss kept an interest in similar sums for his entire life. He found many interesting formulas for *specific* sums $\sum_k t(k)$ – and he was also looking for *general* formulas of this type.

The two most well-known cases in which we have an explicit formula for the sum are *arithmetic* and *geometric* progressions. A geometric progression $t(k) = \text{const} \cdot q^k$ can be described by the condition that the ratio of every two consecutive terms is a constant: $t(k+1)/t(k) = q$. For an arithmetic progression $t(k) = a + b \cdot k$, a similar ratio is a rational function:

$$\frac{t(k+1)}{t(k)} = \frac{a + b \cdot (k+1)}{a + b \cdot k}.$$

It is therefore natural to define a *generalization* of geometric and arithmetic progressions by considering series in which the ratio $t(k+1)/t(k)$ is a rational function:

$$\frac{t(k+1)}{t(k)} = \frac{P(k)}{Q(k)}$$

for some polynomials $P(k)$ and $Q(k)$. Gauss called such series *hypergeometric*. Example of hypergeometric series include Taylor series for most elementary functions: e.g., $\exp(x) = \sum_k t(k)$, where

$$t(k) = \frac{x^k}{k!},$$

and the ratio $t(k+1)/t(k) = x/(k+1)$ is a rational function of k. The usefulness of this notion for combinatorial sums comes from the fact that binomial coefficients are hypergeometric series: if $t(k) = \binom{n}{k}$, then $t(k+1)/t(k) = (n-k)/(k+1)$ is a rational function of k.

How can we describe the sum of such terms? In the two above cases in which we have an explicit formula for the "indefinite" sum $s(n) = t(1) + \ldots + t(n)$, the sum is either itself hypergeometric – for an arithmetic progression, or a hypergeometric term $z(n)$ plus a constant c ($s(n) = z(n) + c$) – for a geometric progression. It turns out that such a term $z(n)$ can be found in many other cases. The question of when the sum $s(n) = \sum_{i=0}^{n} t(i)$ can be represented as $z(n) + c$ for some hypergeometric term $z(n)$ was analyzed, in the late 1970s, by R. W. Gosper Jr. who was working on the first symbolic computation program MACSYMA. Gosper developed an algorithm which, given a hypergeometric term $t(n)$, checks whether the sum is hypergeometric, and if it is, produces the corresponding hypergeometric expression for the sum. This algorithm was actually implemented in *Macsyma*. From the *mathematical* viewpoint, whenever we can find such an expression $z(n)$, it is great. But do we gain anything from the *computational* viewpoint?

At first glance, if we consider *algebraic complexity* (number of arithmetic operations needed to compute a term), we do not gain anything at all. Indeed, what we do gain computationally when we know that a term $t(n)$ is hypergeometric, i.e., that the ratio

$t(k+1)/t(k)$ is a rational function of k? Computing the value of a rational function requires a finite number C of arithmetic operations, so we need C operations to compute $t(1)$ from $t(0)$, C operations to compute $t(2)$ from $t(1)$, etc., until we compute $t(n)$. So, totally, we need $O(n)$ steps to compute $t(n)$. Similarly, if we know that the sum $s(n)$ is hypergeometric, we can compute it in $O(n)$ steps.

On the other hand, even if the sum $s(n)$ is *not* hypergeometric, we can still compute it in $O(n)$ steps: indeed, we start with $t(0) = s(0)$, and then, for $i = 1, \ldots, n$, compute $t(i+1) = t(i) \cdot (P(i)/Q(i))$ (constant number of operations) and $s(i+1) = s(i) + t(i+1)$ (one more operation). The multiplicative constants in these two $O(n)$'s may be different, but it is not even clear which one is faster: on one hand, we need an extra addition to compute $s(n)$ as a sum, but, on the other hand, $s(n)$ can be hypergeometric with a more complex polynomials P and Q.

In short, from the *algebraic* complexity viewpoint, there may be no advantage in using this mathematically interesting result. However, from the viewpoint of *actual* computation time, there is a serious advantage, because actual computation can use *special functions* in addition to arithmetic operations. Specifically, if we factorize both polynomials $P(k)$ and $Q(k)$, we get the expression

$$\frac{t(k+1)}{t(k)} = \frac{(k+a_1) \cdot \ldots \cdot (k+a_p)}{(k+b_1) \cdot \ldots \cdot (k+b_q)} \cdot x$$

for some (generally complex) numbers a_i, b_j, and x. Thus, by using a gamma function $\Gamma(x)$, for which $\Gamma(x+1) = x \cdot \Gamma(x)$ and $\Gamma(n+1) = n!$ for integer n, we get

$$t(n) = c \cdot \frac{\Gamma(n+a_1) \cdot \ldots \cdot \Gamma(n+a_p)}{\Gamma(n+b_1) \cdot \ldots \cdot \Gamma(n+b_q)} \cdot x^n, \tag{2}$$

where

$$c = t(0) \cdot \frac{\Gamma(b_1) \cdot \ldots \cdot \Gamma(b_q)}{\Gamma(a_1) \cdot \ldots \cdot \Gamma(a_p)}.$$

We can also use the formula $x^n = \exp(n \cdot \ln(x))$ to compute x^n. So, if we count the application of exp and Γ functions as one step each in our description of a generalized algebraic complexity, then we can conclude that we can compute $t(n)$ in finitely many computational steps. In this case, finding an explicit hypergeometric expression for the sum $\sum t(i)$ is extremely computationally useful: it brings the complexity down from $O(n)$ to $O(1)$ operations.

One thing deserves mentioning here: from the viewpoint of *practical* computations, the formula (2) is not very useful, because it requires us to divide two extremely large numbers (of orders $n!$) to get a number of reasonable size. This division (similarly to a more well known case of subtracting two large almost equal numbers) is not a good way to computing, because for the result to come out correct, we need to compute both the numerator and the denominator with a very high accuracy. A much better formula can be obtained if we use, instead of the actual (fast growing) gamma functions, auxiliary functions $\Gamma'(n, a_i) = \Gamma(n+a_i)/n! = \Gamma(n+a_i)/\Gamma(n+1)$. In terms of these auxiliary functions, the expression (2) takes the easier-to-compute form

$$t(n) = c \cdot \frac{\Gamma'(n, a_1) \cdot \ldots \cdot \Gamma'(n, a_p)}{\Gamma'(n, b_1) \cdot \ldots \cdot \Gamma'(n, b_q)} \cdot n!^{q-p} \cdot \exp(n \cdot \ln(x)). \tag{3}$$

This expression was actually proposed (in slightly different notations) by Gauss himself. Interestingly, *mathematicians* (and even the authors of the book under review) consider the introduction of the $\Gamma(n+1)$ terms in Gauss's formulas as an unnecessary complication – because it does make formulas (slightly) longer, but, as we have just seen, it make perfect *computational* sense.

In most combinatorial sums used in the analysis of algorithm complexity, the sum goes over all the values of the index for which the corresponding binomial coefficients are different from 0. We can therefore describe the desired sums as *definite* sums $\sum_k t(k)$, where k runs over all the integers. If the corresponding indefinite sum $s(n) = \sum_{k=0}^{n} t(k)$ is hypergeometric, then we can also compute the corresponding definite sum. But what if the indefinite sum is *not* hypergeometric (and often it is not)? In many of such cases, the definite sum is either hypergeometric itself, or is a sum of several hypergeometric term. Chapter 8 describes an algorithm which, given an integer r, checks whether a definite sum $\sum_k t(k)$ can be represented as a sum of r (or fewer) hypergeometric terms (and explicitly produces these terms if they exist). This algorithm, first proposed in Petkovšek's 1991 Ph.D. dissertation, capitalizes on the techniques previously proposed by the two other authors of the book.

In some cases, the desired sum $\sum t(k)$ is not hypergeometric, so we need a larger class of functions to describe such sums. The main reason why such a class is needed is that the class of hypergeometric functions is not algebraically closed: while the *product* of two hypergeometric functions is (trivially) also hypergeometric, the *sum* is often not: one can easily check that even the sum $t(n) = a_1^n + a_2^n$ of two geometric progressions is not hypergeometric.

In this particular non-hypergeometric sum, the first term $t_1(n) = a^n$ is a solution to the following first-order difference equation with constant coefficients: $(S - a_1)t_1 = 0$, where $(St)(n) \stackrel{\text{def}}{=} t(n+1)$; similarly, the second term $t_2(n) = a_2^n$ is a solution to $(S - a_2)t_2 = 0$, hence the sum is a solution to the following *second-order* difference equation with constant coefficients: $(S - a_1)(S - a_2)t = 0$, i.e., $t(n+2) = (a_1 + a_2) \cdot t(n+1) - a_1 \cdot a_2 \cdot t(n)$.

In general, each hypergeometric term $t_i(k)$ is, by definition, a solution to the first-order difference equation with *rational* coefficients $t_i(k+1) = (P_i(k)/Q_i(k)) \cdot t_i(k)$. Therefore, the sum $t(k) = t_1(k) + \ldots + t_r(k)$ of such terms satisfies a higher-order difference equation with rational coefficients:

$$t(k+r) = \frac{P_1'(k)}{Q_1'(k)} \cdot t(k+r-1) + \ldots + \frac{P_r'(k)}{Q_r'(k)} \cdot t(k). \tag{4}$$

Functions satisfying such equations (4) are called *holonomic*. The set of holonomic functions is closed under addition, multiplication, indefinite addition, etc. A natural algorithmic definition of such a function starts with functions which are solutions of equations (4), and then allows addition, multiplication, etc.

There exist algorithms for checking whether two given holonomic functions (of one or several variables) coincide or not.

The programs in MAPLE and MATHEMATICA implementing all above algorithms can be downloaded from the book's website http://www.cis.upenn.edu/ wilf/AeqB.html. Moreover, the entire book can be downloaded for free (the visionary publisher agreed to it!). For those who want to study the book seriously: there are several minor typos – which are all mentioned on the book's website.

It is worth mentioning, that for several variables, the problem of checking whether $A = B$ quickly becomes computationally intensive. This is not surprising since one can show that satisfiability can be described in these terms. Namely, if we have a propositional formula with v Boolean variables $z_1, \ldots, z_v$, then we can define n auxiliary holonomic functions $a_i(n_i), 1 \le i \le v$, as follows: $a_i(0) = 1$ and $a_i(n_i + 1) = -a_i(n_i)$ (hence, $a_i(n_i) = (-1)^{n_i}$). Constants are holonomic functions, and sums

and products of holonomic functions are holonomic, so for every i, the function

$$s_i(n_i) = \frac{1}{2} \cdot (a_i(n_i) + 1)$$

is also holonomic. We assign, to every Boolean variable z_i, the corresponding function $f^{[z_i]} = s_i(n_i)$, to its negation, the function $f^{[\bar{z}_i]} = 1 - f^{[z_i]}$; to a disjunction, the function $f^{[a \vee b]} = f^{[a]} + f^{[b]} - f^{[a]} \cdot f^{[b]}$, and to conjunction, $f^{[F_1 \& F_2]} = f^{[F_1]} \cdot f^{[F_2]}$. As a result, to the original Boolean formula F, we assign a holonomic function $f^{[F]}(n_1, \ldots, n_v)$ such that this function f is identically 0 if and only is not satisfiable. Thus, checking equalities for thus defined holonomic functions is *NP-hard*.

Moreover, we can add $\sum_{n_i=0}^{1}$ instead of $\exists z_i$ and $\prod_{n_i=0}^{1}$ instead of $\forall z_i$ and thus get a holonomic function equivalent to an arbitrary *quantified satisfiability* problem – hence, we can prove that the corresponding problem is *PSPACE-hard*.

In this review, I have only mentioned the computational complexity of the problems and algorithms, and I said nothing about a beautiful mathematical theory behind them, a theory for which Herbert Wilf and Doron Zeilberger received a prestigious Steele prize from the American Mathematical Society. I just want to mention that a large part of the algorithms for computing a definite sum $f(n) = \sum_k F(n,k)$, where $F(n,k)$ is a hypergeometric function of both variables, is based on the possibility to find a "dual" hypergeometric expression $G(n,k)$ for which either

$$F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k) \tag{5}$$

– then $f(n) = \text{const}$ – or a more sophisticated equality hold for some polynomials $a_j(n)$

$$\sum_{j=0}^{J} a_j(n) \cdot F(n+j,k) = G(n,k+1) - G(n,k) \tag{6}$$

– in this case $f(n)$ is a holonomic function satisfying the difference equation

$$\sum_{j=0}^{J} a_j(n) f(n+j) = 0.$$

There is an algorithm which, given F, always returns G and a_i. The left-hand side of (6) can be unraveled via unraveling several consequent first-order difference operators – like a telescoping fishing rod. This is how Zeilberger called his algorithm – *creative telescoping*.

The equality (5) is as simple and as beautiful as its continuous analog

$$\frac{\partial F}{\partial x} = \frac{\partial G}{\partial y}$$

which is a part of a definition of a complex analytical function $F(x,y) + \mathrm{i}G(x,y)$, and the consequences of the simple equality (5) are as deep, unexpected, beautiful (and as computationally helpful) as those of complex analysis.

The book also contains an interesting history, from the pioneer 1945 algorithm by Sister Mary Celine Fasenmyer (who happened to die, at the age of 90, in the exact same year in which this book was published), to algorithms and results of Gosper, Zeilberger, Wilf, and Petkovšek.

Although the book is very technical, it is written in a very popular and understandable way. It starts with the basics, it gently guides the reader through the programs, through the formulas, and

through the numerous examples. Because of the book's unusual level of understandability, when it was published in 1996, it was selected as a Main Selection of the Library of Science – an honor rarely reserved for technical books.

I just love this book, and I hope y'all will too.

Review of[6]

Communicating and mobile systems: the π-calculus
Author: Robin Milner
Publisher: Cambridge Univ Press, 1999
Paperback: ISBN 0521658691 $22.45
160 Pages

Reviewer: Riccardo Pucella
Department of Computer Science
Cornell University
email: riccardo riccardo@cs.cornell.edu

For a long time, the quest for a formal foundation of concurrent programming has kept semanticists happy and busy. What many were looking for was a calculus playing the same role for concurrent programming as the λ-calculus did for functional programming. One of the many difficulties encountered was that there are many aspects to concurrent programming and one has to decide what to consider primitive. In the early eighties, Milner introduced CCS [2] and Hoare introduced CSP [1] as calculi to model concurrent processes where communication (or interaction) between processes is taken as primitive. The main characteristic of these calculi was that one could reason algebraically about such processes, via equational laws from which one could define various notions of process equivalence. Two processes are deemed equivalent when they "have the same behavior" for some suitable notion of behavior.

In 1989, Milner extended CCS to take full advantage of named channels. To communicate over a channel, a process needs to know the name of that channel. Processes can moreover pass channel names over channels. The resulting calculus, the π-calculus, allows process A to pass the name of a channel c to another process B, at which point B can communicate via channel c to another process, with which B could not communicate before receiving the channel name. This feature enables the π-calculus to have a dynamic communication topology, which can be used to model mobility. In this setting, mobility is taken to be the mobility of links between components.

For the past ten years since the introduction of the π-calculus (ten years... recall that ten years ago, the first version of Microsoft Windows was barely out the door), the available literature on the π-calculus has consisted of original papers (for instance [5], [4], [7]) and PhD theses (for instance [6]). No longer.

Milner's short monograph, peaking at about 160 pages, gives a well-rounded and self-contained introduction to the material. According to the Preface, the book is based on lecture notes for a final-year undergraduate course, and the material is acknowledged to be challenging. No formal prerequisites are listed, short of a working knowledge of automata theory. However, the material does require a certain level of mathematical maturity. The presentation is in two parts, splitting the book roughly in half.

Part I ("Communicating systems") covers the basic material on communicating system, with a focus on the algebraic treatment of concurrency and communication. There is no mobility in this first part. It summarizes the work of Milner on CCS, explained more thoroughly in say

[3]. The presentation is very much geared towards the introduction of mobility in the second part of the monograph. After a quick introductory chapter, Chapter 2 looks at the interaction of automata, and why classical language equivalence for automata is not appropriate when we consider interacting automata. Chapter 3 introduces sequential processes, as a way of describing labelled transition systems, and defines the all-important concept of bisimulation. Chapter 4 extends the notion of processes to concurrent processes. Chapter 5 generalizes the labelled transitions systems and bisimulation results of chapter 3 to the concurrent setting, and introduces reaction rules for concurrent processes (which are reduction rules taking into account communication). Chapters 6 and 7 focus on notions of equivalence which can be established by looking at how a process interacts externally with others (regardless of which internal actions it performs).

Part II of the monograph ("The π-Calculus") introduces mobility into the framework of part I, that is the idea of sending channel names over communication channels to enable communication. Chapter 8 motivates this idea, by giving examples where this kind of mobility arises naturally in systems. Chapter 9 introduces the π-calculus itself and its reaction rules. Both the monadic version (channels carry a single value at a time) and the polyadic version (many values per channel) are described. Chapter 10 gives sample applications, including how to model simple systems, how to represent data structures, how to program with lists in a functional style. Chapter 11 introduces a simple type system for the calculus, which classifies the kind of information exchanged during an interaction. The type system is used to help design communication protocols to model object-oriented features in the π-calculus, as well as functional programming features. Chapter 12 turns to theory and defines bisimilarity for π-calculus processes. Chapter 13 studies observational equivalence for π-calculus processes. Chapter 14 concludes with a discussion and an overview of related work.

Positive points. The monograph is short, self-contained, and extremely readable. It covers all the important points in enough detail for the subtleties involved to be understood, while still being general enough that the material is applicable to other concurrent calculi variants (especially the material in part I). There are many examples throughout the book, and they form an intrinsic part of the material, serving both a motivating and an illustrative purpose. The sample applications of the π-calculus help the reader make the transition from seeing the π-calculus as an abstract mathematical toy to seeing it as a formalization of a simple concurrent programming language.

Slightly less positive points. The book is short and very focused. Therefore, there is not as much theory as one expects from a book about a formal model of concurrent computation. Moreover, many very interesting venues of investigation are not described (the current work on bisimulation in more general concurrency models, to pick one example, or the variations obtained by considering different semantics for the primitives). In all fairness, the book does not claim to be a theory book, or to cover the broad spectrum of approaches to concurrent calculi, or even to bring the reader up to date with present-day research. Milner refers both to his earlier work on CCS [3] and the forthcoming book by Sangiorgi and Walker "A Theory of Mobile Processes" for a more in-depth treatment of those aspects.

In summary. This monograph is an excellent introduction to the π-calculus. It will not bring much to the researcher already familiar with the π-calculus, save a convenient self-contained reference booklet, but is by far the best starting point for the beginner.

References

[1] C. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[2] R. Milner. *A Calculus of Communicating Systems.* Number 92 in Lecture Notes in Computer Science. Springer-Verlag, 1980.

[3] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[4] R. Milner. The polyadic π-calculus: a tutorial. In W. Brauer F.L. Bauer and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993.

[5] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1):1–77, 1992.

[6] D. Sangiorgi. *Expressing mobility in process algebras: first-order and higher-order paradigms.* PhD thesis, Department of Computer Science, University of Edinburgh, 1993. CST-99-93, also published as ECS-LFCS-93-266.

[7] D. Walker. Objects in the π-calculus. *Information and Computation*, 115:253–271, 1995.