**The Book Review Column**[1]
by Frederic Green

Department of Mathematics and Computer Science
Clark University
Worcester, MA 01610
email: fgreen@clarku.edu

This column, the spotlight is on complexity, computational and communicational:

1. **Mathematics and Computation**, by Avi Wigderson. A sweeping look at computational complexity and adjacent fields, especially how they interact with mathematics. Review by Frederic Green.

2. **Communication Complexity and Applications**, by Anup Rao and Amir Yehudayoff. A textbook on this important topic. Review by Michaël Cadilhac.

As always, please contact me to write a review; choose from among the books listed on the next pages, or, if you are interested in anything not on the list, just send me a note.

## BOOKS THAT NEED REVIEWERS FOR THE SIGACT NEWS COLUMN

### Algorithms

1. *Algorithms and Data Structures Foundations and Probabilistic Methods for Design and Analysis*, by Helmut Knebl
2. *Algorithms and Data Structures*, by Helmut Knebl
3. *Beyond the Worst-Case Analysis of Algorithms*, by Tim Roughgarden

### Computability, Complexity, Logic

1. *Applied Logic for Computer Scientists: Computational Deduction and Formal Proofs*, by Mauricio Ayala-Rincón and Flávio L.C. de Moura.
2. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*, by Martin Grohe.
3. *Semigroups in Complete Lattices*, by P. Eklund, J. Gutiérrez García, U. Höhle, and J. Kortelainen.

### Miscellaneous Computer Science

1. *Elements of Causal Inference: Foundations and Learning Algorithms*, by Jonas Peters, Dominik Janzing, and Bernhard Schölkopf.
2. *Partially Observed Markov Decision Processes,* by Vikram Krishnamurthy
3. *Statistical Modeling and Machine Learning for Molecular Biology*, by Alan Moses
4. *Language, Cognition, and Computational Models,* Theirry Poibeau and Aline Villavicencio, eds.
5. *Computational Bayesian Statistics, An Introduction,* by M. Antónia Amaral Turkman, Carlos Daniel Paulino, and Peter Müller.
6. *Variational Bayesian Learning Theory,* by Shinichi Nakajima, Kazuho Watanabe, and Masashi Sugiyama.
7. *Knowledge Engineering: Building Cognitive Assistants for Evidence-based Reasoning*, by Gheorghe Tecuci, Dorin Marcu, Mihai Boicu, and David A. Schum.
8. *Quantum Computing: An Applied Approach*, by Jack D. Hidary

### Discrete Mathematics and Computing

1. *Mathematics in Computing*, by Gerard O'Regan
2. *Understand Mathematics, Understand Computing – Discrete Mathematics That All Computing Students Should Know*, by Arnold L. Rosenberg and Denis Trystram

### Cryptography and Security

1. *Computer Security and the Internet: Tools and Jewels*, by Paul C. van Oorschot

### Combinatorics and Graph Theory

1. *The Zeroth Book of Graph Theory: An Annotated Translation of Les Réseaux (ou Graphes) – André Sainte-Laguë (1926)*, translated by Martin Charles Golumbic
2. *Finite Geometry and Combinatorial Applications*, by Simeon Ball
3. *Combinatorics, Words and Symbolic Dynamics,* Edited by Valérie Berthé and Michel Rigo

## Programming etc.

1. *Formal Methods: An Appetizer*, by Flemming Nielson and Hanne Riis Nielson
2. *Sequential and Parallel Algorithms and Data Structures*, by P. Sanders, K. Mehlhorn, M. Dietzfelbinger, R. Dementiev

## Miscellaneous Mathematics

1. *Introduction to Probability*, by David F. Anderson, Timo Seppäläinen, and Benedek Valkó.
2. *Algebra and Geometry with Python*, by Sergei Kurgalin and Sergei Borzunov.

**Review of**[2]
**Mathematics and Computation**
**by Avi Wigderson**
**Princeton University Press, 2019**
**440 pages, Hardcover, $49.95.**

**Review by**
**Frederic Green** (fgreen@clarku.edu)
**Department of Mathematics and Computer Science**
**Clark University, Worcester, MA**

# 1 Introduction

Mathematics and computation are inextricably entangled[3]. We couldn't do one without the other. The need to calculate can be traced to early human history, and mathematics developed in large part to enable computation. And computation is necessary to propel mathematics. One often loses sight of the fact that the great mathematicians of the past were also prodigious computers: For example, Gauss, Kummer and the other great pioneers of number theory did vast amounts of computation to arrive at or reinforce many of their insights.

And after all, the modern concept and technology of computation grew out of mathematics. While the idea of mechanizing calculation, or even thought, goes back for centuries, having been pondered by philosophers, mathematicians, scientists, and engineers, the astounding technological revolution of the late $20^{th}$ century had its seeds in purely mathematical considerations. Cantor's set theory, Hilbert's problems, the formalization of logic, and Gödel's incompleteness theorem ultimately gave rise to the mathematical theory of computation. Turing's famous paper on "computable numbers" was (in Wigderson's words) the "big bang" that led to modern computer science, and in particular the central ideas of Computational Complexity Theory.

Mathematics and computation continue to work hand in hand, and it is of vital importance to understand their interaction, and how each one nurtures the other. That, in large part, is what this book is about, as we next explore.

# 2 Synopsis

The book is divided into 20 chapters. It would be impossible to improve on the summary Wigderson gives in his own introduction. Nevertheless, since you're reading this review, there's a good chance you don't have that introduction, so here's my own version.

One of my professors in graduate school structured final exams like certain cultural events. One of them took the form of a menu (some problems were appetizers, others main courses, and finally a dessert), another the form of a concert program (a number of movements, with labels such as "Andante" and "Allegro"). I can't help but fancy that this book is structured something like a performance, perhaps a play with incidental music. There is an Overture (the introduction, Chapter 1), a Prologue (a broad look at the roots of the field, in Chapter 2, aptly titled "Prelude"), a long first act (Chapters 3 - 12, studying the many fundamental aspects of computational complexity, concentrating on the key resource of time), an "Interlude" (Chapter 13, on



[3]Given the technical overtones of the word, I hope you forgive its colloquial use here.

particular interactions between complexity and the rest of mathematics), a somewhat briefer second act (Chapters 14 - 19), concentrating on other resources such as time, and various applications, and finally an "Epilogue" (Chapter 20), putting forth a perspective on the Theory of Computing broadly conceived.

Here, then, is a more detailed synopsis of the action:

In addition to the summary mentioned previously, we learn in the Introduction about the backstory referred to above: The origins of both mathematics and computation, and the meteoric ascent of computation over the course of the past century. In the brief Prelude we are introduced to the principal characters, various mathematical problems (regarding such diverse notions as Diophantine equations, logic, smooth manifolds, and elliptic equations), how we understand such problems via algorithms, and the efficiency of those algorithms.

The action then commences in earnest in Chapter 3 ("Computational complexity 101"), which motivates and defines classification/decision problems and the complexity classes at the heart of computational complexity: $\mathcal{P}$, $\mathcal{NP}$, and $\mathrm{co}\mathcal{NP}$. We encounter both the nondeterminism and verification characterizations of $\mathcal{NP}$, and how it contrasts with $\mathrm{co}\mathcal{NP}$. We see the idea of reducibility (central not only to complexity theory, but to all of computer science), and its use in defining completeness for both the classes $\mathcal{NP}$ and $\mathrm{co}\mathcal{NP}$. Many examples of problems in $\mathcal{P}$ as well as $\mathcal{NP}$-complete problems are given, and the scene closes with some reflections on the depth and impact of the $\mathcal{P}$ vs. $\mathcal{NP}$ question. The horizons are broadened in the subsequent chapter to classes "around" $\mathcal{NP}$, including (probably) much larger classes such as $\mathcal{PH}$, $\#\mathcal{P}$, and $\mathcal{PSPACE}$. They are also refined in discussions of constraint satisfaction problems, average-case complexity, and one-way functions with their relation to cryptography.

The very hard question of hardness now makes its entrance in the next chapter: How do we prove the widely believed conjecture $\mathcal{P} \neq \mathcal{NP}$? Oracles relative to which either outcome holds imply that old-fashioned diagonalization will not suffice. Instead, there is some hope that exponential circuit lower bounds would provide a technique. However, the known circuit lower bound techniques also face imposing barriers in the form of natural proofs, an idea revisited in a later chapter.

The verification view of $\mathcal{NP}$ evokes the idea of mathematical proof, and indeed how much effort (complexity) it might take to prove a theorem. *Proof complexity* is the subject of the next chapter (6, if you've lost count). Cook and Reckhow showed that there are polynomially bounded proof systems iff $\mathcal{NP} = \mathrm{co}\mathcal{NP}$, thus endowing a deep significance to the (presumed) inequality of those classes. There are a number of concrete proof systems, algebraic (Nullstellensatz and Polynomial Calculus systems), geometric (e.g., Cutting Plane proofs), and logical (e.g., Frege systems). As different as they may seem, there are many technical connections between proof complexity and circuit complexity, although unfortunately another commonality is the difficulty in making truly significant advances in either one.

The next 4 chapters (7 – 10) are devoted to various aspects of randomness, a major theme in the field, the author himself being a major authority on that theme. While randomness is immensely useful for efficient computation, evidence has accumulated over the years that it may not be as powerful a resource as originally thought. Indeed, the evidence supports the conjecture $\mathcal{P} = \mathcal{BPP}$, originally thought to be false, and now widely held. The vast field of derandomization, and its relationship with computational and cryptographic pseudo-randomness, and how hardness leads to pseudo-randomness, are discussed in Chapter 7. For some classes of objects (e.g., random graphs) certain properties hold for almost all elements of the class (e.g., the property of being "$(3 \log n)$-Ramsey"). This manifestation of pseudo-randomness, called "abstract pseudo-randomness," is covered in the following chapter. It is remarkable that some of the most important problems of mathematics, including the Riemann Hypothesis and $\mathcal{P}$ vs. $\mathcal{NP}$, can be cast in this form. E.g., for $\mathcal{P}$ vs. $\mathcal{NP}$, nearly all functions require exponential circuits. The trick is to find a *particular* function (e.g., SAT) that has this property, an idea colorfully described as "finding hay in haystacks." A central tool in

pseudo-randomness is the expander graph, also discussed in this chapter. We then move on, in Chapter 9, to the problem of extracting randomness from less than perfect randomness. And Chapter 10 turns to the role of randomness in proofs. Here we meet the principal character in the world of interactive proofs, $\mathcal{IP}$, and the surprising (and non-relativizing) correspondence $\mathcal{IP} = \mathcal{PSPACE}$. This was a milestone along a line of research that led to other amazing results such as $\mathcal{MIP} = \mathcal{NEXP}$, and the scaled-down version of the latter which exhibited probabilistically checkable proofs for $\mathcal{NP}$, which has enormous impact on hardness of approximation. The discussion of zero-knowledge proofs happily conveys to the reader much more than zero knowledge about this singular phenomenon. This chapter ends with a sweeping perspective on the extraordinary applicability of such proofs to daily life as well as the sciences.

Quantum computing (QC) is surely among the most revolutionary ideas in computation, regardless of its practical outcome. It elevates the Church-Turing thesis to the status of a physical law. Given its brevity Chapter 11 is a surprisingly deep survey of how quantum physics has interacted with computational complexity, in ideas as diverse as the fundamental nature of QC itself, its power via the famous algorithms of Shor and Grover, and how such concepts as $\mathcal{BPP}$ and $\mathcal{NP}$ have been imported into the classes $\mathcal{BQP}$ and $\mathcal{QMA}$. Various formulations of QC (e.g., the quantum Turing machine, adiabatic computation, quantum circuits) are discussed, as is the notion of quantum interactive proofs, quantum certification, and even building quantum computers.

The last scene in the book's "first act," Chapter 12, looks into the complexity of polynomial arithmetic and its associated model of computation, arithmetic circuits. This captures the easiness or hardness of some of the most important quantities in mathematics, such as symmetric polynomials, matrix multiplication, the determinant and the permanent. These considerations lead to natural analogs of $\mathcal{P}$ and $\mathcal{NP}$, namely $\mathcal{VP}$ and $\mathcal{VNP}$. Although the challenges are still imposing, the prospect of separating $\mathcal{VP}$ and $\mathcal{VNP}$ appear somewhat brighter, and have led to intriguing developments such as geometric complexity theory and a bridge between boolean and arithmetic complexity via polynomial identity testing.

The "Interlude" recounts a number of important interactions between mathematics and complexity. Some of the instances involve such areas as Number Theory (in the guise of primality testing, culminating in the AKS algorithm), Group Theory (considerations which led, among other things, to interactive proof systems), Statistical Physics (illustrating the intriguing connection between such physical quantities such as the partition function on the one hand, and the complexity of counting problems on the other), and Invariant Theory (a prominent example being geometric complexity theory). Many such interactions make appearances elsewhere in the book, but these concrete cases dramatize forcefully how important the two fields are to each other.

The latter part of the book mainly focusses on resources other than time. The first of those chapters (14) is a brief introduction to space complexity. Here we encounter the fundamental complexity classes $\mathcal{L}$, $\mathcal{BPL}$, and $\mathcal{NL}$ (deterministic, probabilistic, and nondeterministic logspace, respectively). Key results, such as Savitch's Theorem and Immerman/Szelepcsényi's $\mathcal{NL} = \mathrm{co}\mathcal{NL}$ among others, sketch the contrasts and challenges in space vs. time complexity. Speaking of sketches, there is also a quick summary of streaming algorithms, which use very little space for massive (possibly infinite) amounts of streaming data. Inevitably, one must move on to *constant* space, which is treated in the final section, and includes an account (with some proof outlines) of how seminal results on finite automata led to the remarkable theorem of Barrington on the completely unexpected power of width 5 branching programs.

The remaining chapters cover a remarkably diverse set of topics. Communication complexity is the subject of Chapter 15. The emphasis here is on its applications, which include such diverse fields as VLSI, pseudo-randomness, the limits of linear programming, and formula lower bounds. The notion also introduced an interactive element to information theory, with corresponding developments in coding theory.

Prediction is very difficult, especially about the future. So said Niels Bohr[4]. But surprisingly, as we learn in Chapter 16 on on-line algorithms, it's not as useful as we may think actually to *know* the future: The past can give us algorithms so competitive that it's essentially not worth it. Learning (in a certain sense) from the past forms a nice segue to computational learning theory (Chapter 17). This includes accounts of generalities such as the classification problem (illustrated through the classification of hyperplanes via perceptrons and linear programming), as well as detailed discussions, with definitions and further examples, of inductive inference and PAC-learning. Cryptography (Chapter 18) is probably the killer app of computational complexity. It also brings together many ideas, entailing such connections as that between computation and information theory, probabilistic computation (semantic security), interactive proofs (e.g., zero knowledge), and the complexity of particular problems (especially factoring). The chapter explores many aspects of the field, including the various definitions and their motivations, and recent advances such as homomorphic encryption. The chapter ends on a sobering question regarding the complexity of factoring and its ramifications. Chapter 19 is about distributed and (mostly) asynchronous computing, covering some of the classical problems such as the dining philosophers and consensus. It includes fairly detailed outlines of the proofs of a few impossibility results for deterministic algorithms, some of which incorporate novel connections with algebraic topology. But the striking power of randomization is again illustrated by several algorithms that evade those impossibilities through probabilistic computation.

Finally, we come to Chapter 20, the "Epilogue." It is one of the most interesting chapters in the book, by far the longest, and, like any good epilogue, ties numerous plot threads together. The theme of this chapter is the enormous depth and reach of the Theory of Computing (ToC). How it collaborates and interacts with other fields is prominently discussed, as are ToC methodologies and the "computational complexity lens on the science." The latter helps to resolve problems in such fields as molecular biology, evolution, neuroscience, and quantum physics. That last example regards one of the most exotic and intriguing of such foci, namely the use of computational complexity to understand the firewall paradox of black holes (as I understand it, as manifested in what is known as the "ER=EPR" hypothesis of Maldecena and Susskind[5]). There are also fascinating discussions of the philosophical, technological, and pedagogical implications, as well as some well-taken constructive criticism for the ToC community.

# 3 Opinion

Far too many mathematical papers and texts fall short on *explanations* (I'm tempted to say[6] that the non-readability property in the universe of such papers is pseudo-random, or even that too many zero-knowledge proofs make their way into the literature). While one does see some attention paid to such issues, we[7] are left wanting more: Why do we define things a certain way? Why is this theorem interesting? How might one come up with this proof? What is the intuition behind it? How did people come to study these concepts, where do they come from, and where are they going?

This book fills in this gap for an enormous swath of ToC, largely but not exclusively as regards computational complexity, and how it interacts with mathematics. At the same time, it gives a panoramic view of the field. It is a very hi-def panorama. Wigderson includes numerous careful definitions (with lucid explanations and backgrounds in all cases), and many intuitive and cogent summaries of important proofs. While the definitions often approach rigorous ones, the style throughout is informal, engaging, and always infused with enthusiasm and a gentle sense of humor. The text is adorned with numerous enlightening, often

[4]The quote is often attributed to the likes of Yogi Berra, Samuel Goldwyn, and others.

[5]Google it!

[6]using definitions from this book

[7](Well, some of us at any rate.)

entertaining, footnotes[8], sometimes with the profusion of the ones in this review, but much more informative. There are frequent "exercises" integrated into the text, in the form of invitations to the reader to try proving something before an answer is given (if indeed it is). Of extra special value are the very extensive references, an invaluable resource for anyone who wants reliable pointers to the literature.

It's hard to criticize a book that I enjoyed reading as much as this one. But here goes: I would have liked to have an index. This quibble is really a moot point, however. By prior arrangement with the publisher there is a free on-line (fully searchable) pdf version at `https://www.math.ias.edu/avi/book`.

While the text is ever mindful of the proliferation of applications of ToC, it is firm in the conviction that the field is an intellectual pursuit that is driven above all by curiosity. Reminiscent of the saying that "computer science is no more about computers than astronomy is about telescopes" (commonly, though in all likelihood apocryphally, attributed to Dijkstra), Wigderson asserts,

> "[The] intrinsic study of computation transcends human-made artifacts, and underlies natural and artificial processes of all types."

I have long held this viewpoint, often to the bafflement of my (highly application-oriented) students.

Who will benefit by reading this book? I would guess just about anyone with an undergraduate education in the sciences, or with a sufficiently quantitative background. The arguments are described conceptually rather than mathematically, but with sufficient precision that some knowledge at that level is necessary to appreciate most of the book (although a significant part of it doesn't even require that). The audiences likely to benefit most include researchers in other disciplines who wish to learn more about the field; graduate students and researchers in adjacent scientific fields who would like to do research in ToC, and want to learn more what it's about; and lastly, researchers in ToC! With regard to that last audience, I am in some respects reminded of Aaronson's "Quantum Computing Since Democritus"[9]. Here the style is decidedly different (as anyone who knows these two authors would easily guess), and the scope is considerably broader. But it serves as an excellent and inspirational guide to the field not only for outsiders but also for those of us who are already involved. I will keep this book well within reach whenever I find myself in unfamiliar parts of the literature.

"Mathematics and Computation" is a truly outstanding book. One could hardly find a more expert and capable field guide than Avi Wigderson. If you want to get a good idea of what happens in almost any subfield of ToC, start here!

[8] (which, like this one, can be safely ignored)

[9] Reviewed by yours truly on these pages in SIGACT News **44**(4) (2013), pp. 42-47.

**Review of**[10]
**Communication Complexity and Applications**
**by Anup Rao and Amir Yehudayoff**
**Cambridge University Press, 2020**
**266 pages, Hardcover, $53, ISBN 1108497985**

**Review by**
**Michaël Cadilhac (michael@cadilhac.name)**
**School of Computing**
**DePaul University**

# 1 Overview

At its core, communication complexity is the study of the amount of information two parties need to exchange in order to compute a function. For instance, Alice receives a string of characters, Bob receives another, and they should decide whether these strings are the same with as few rounds of communication as possible. Multiple settings are conceivable, for instance with multiple parties or with randomness. Upper and lower bounds for communication problems rely on a wealth of mathematical tools, from probability theory to Ramsey theory, making this a complete and exciting topic. Further, communication complexity finds applications in different aspects of theoretical computer science, including circuit complexity and data structures. This usually requires to take a "communication" view of a problem, which in itself can be an eye-opening vantage point.

This book focuses on classical and more recent results in communication complexity and provides applications in theoretical computer science. It does so in a very efficient, engaging, and at times entertaining way. I first present the contents of the book before giving my opinion on it.

# 2 Contents

The book is neatly divided in two main parts, one focusing on communication complexity, the other on applications.

## 2.1 Part I: Communication Complexity

**Chapter 1** starts by exploring deterministic two-party protocols. The fundamental notion of *rectangle* is introduced and used to examine protocol trees. The first lower bounds are deduced from these concepts. The chapter is concluded with a discussion on direct sums, i.e., solving several copies of a problem.

From the outset, the bold stylistic choices of the authors are in full play, with lots of pictures, examples, comments, and an engaging tone.

**Chapter 2** introduces the no less fundamental notion of *rank of the matrix associated with a communication problem* (that is, the matrix $M$ for which $M_{xy}$ is the boolean output of the protocol when Alice receives $x$ and Bob $y$). Starting with the fact that the communication complexity of a problem is between the logarithm of the rank and the rank, a wealth of lower and upper bounds are presented. The log-rank conjecture

is then introduced: the communication complexity is upper bounded by a power of the log of the rank. A weaker upper bound is proved ($O(\sqrt{rank}\log^2 rank)$).

**Chapter 3** focuses on randomized communication complexity, devising protocols for some problems. Yao's minimax principle is proved: worst-case randomized complexity is related to average-case.

**Chapter 4** adds more players in the arena by considering multi-party protocols using the numbers on foreheads model. This means that each player knows the input of all players but themselves. After providing some protocols, lower bounds are studied, relying on Ramsey theory.

**Chapter 5** studies *discrepancy*, a measure of randomness. The precise definition quantifies how random is a function over a set (given a distribution). The use case here is to look at how random a communication problem looks on well-structured sets, for instance rectangles: if the discrepancy is at most $\gamma$, then the communication complexity is at least $\log(1/\gamma)$. This is used to derive lower bounds for randomized protocols and multi-party protocols. A significant portion of the chapter is dedicated to using discrepancy to show a lower bound on disjointness in the multi-party deterministic setting, a recent result.

**Chapter 6** looks at information and *entropy*, a measure of information content. Entropy is essentially the measure of how many bits are needed to encode an object. After a detour through combinatorial applications of entropy and the creation of an arsenal of mathematical tools, lower bounds take the stage again. This culminates in the proof that computing disjointness with error $1/2 - \epsilon$ in the randomized setting requires $\Omega(\epsilon^2 n)$ communication.

**Chapter 7** develops an interactive view of entropy: what is the amount of actual information transmitted between two parties? In other words, what did the parties learn? This leads to several definitions and the notion of *compression* of a protocol, i.e., a new protocol of smaller length that *simulates* the original protocol. These are fairly recent results that culminate in the Direct Sum Theorem: The randomized complexity of $k$ copies of a protocol with complexity $c$ is at least $\Omega(c\sqrt{k}/\log c)$. The chapter is concluded with a short literature review of the past 10 years on this topic.

**Chapter 8** focuses on *lifting*, a technique borrowed from lower bounds in monotone circuit complexity. Here, the technique is used to derive lower bounds in communication complexity from lower bounds in decision tree complexity, a simple, restricted model of computation. This is used in particular to show that the log-rank conjecture cannot hold with an exponent less than 2.

## 2.2 Part II: Applications

**Chapter 9** deals first with circuit complexity. Karchmer-Widgerson games are introduced: these are communication problems defined by a boolean function. The game for $f\colon \{0,1\}^n \to \{0,1\}$ is for Alice to receive an entry mapped to 0, Bob an entry mapped to 1, and for them to find a position where their entries differ. If this game can be solved with $d$ bits of communication, then there is a circuit of depth $d$ computing $f$. This is used to derive lower bounds. The chapter then turns to proof systems, first with the complexity of proving the Pigeonhole Principle with resolution refutations, then the complexity of vertex cover using cutting planes refutations.

**Chapter 10** studies memory complexity, through *branching programs*, a sort of binary decision diagram where each layer queries the same variable. A restriction thereof, *streaming algorithms*, further requires the data to be read once, and in sequence. Using communication complexity, some lower bounds are derived for streaming algorithms first, then for general branching programs.

**Chapter 11** deals with *data structures*, focusing on both time and space complexity. After presenting some classical data structures, lower bounds are devised, again by bridging the existence of some efficient data structures with that of efficient communication protocols. The lower bounds are in the form of trade-offs, e.g., the product of query time and space usage is at least a certain value. This is done in the case of *static* data structures as well as *dynamic* ones, which can be updated. The chapter is concluded with lower bounds on data structures for graph problems.

**Chapter 12** looks at *polytopes* and how they can encode computational problems. In this representation, the number of facets of the polytope is tightly related to the complexity of solving the problem. This leads to the definition of the *extension complexity* of a polygon: this is the least number of facets of any extension of the polygon. (Extension here means that there is a linear map from the extension to the original polygon.) After covering examples of polygons that have a low extension complexity, upper and lower bounds on that value are derived, in particular using the tools of Chapter 6. Applications in circuit complexity are presented.

**Chapter 13** introduces *distributed computing*. Therein, a network is defined as an undirected graph and each vertex represents one of the parties. Their goal is to compute something together with no prior knowledge of the underlying graph. An example is given, in which parties color the underlying graph with few colors, then lower bounds are proved. This is done by grouping vertices together so that the overall protocol looks like a two-party protocol.

# 3 Opinion

We all (hopefully) had a teacher who was too excited about the material for their own good. Tangents, digressions, tidbits, they can be hard to follow even though they are engaging and share their enthusiasm. Now imagine that this teacher had to encapsulate that energy into a book: although this would provide structure, it would be limiting and contrived, and this would be done at a loss. The authors of this textbook beautifully solved that conundrum with *extensive* use of margin notes, some providing aha-moments, while other notes give more context and doorways to more advanced topics. These margin notes are an essential feature of this book and are done with care, taste, and gusto, showcasing a mastery not only of the topics, but also of drawing software. One criticism is that it sometimes happens that these side notes aren't optional reading, although they look like it, and this can be slightly confusing.

The choice of topics makes for an appealing selection and a complete cocktail, with a clear direction. The more advanced topics are very well presented and require little prior understanding of theoretical computer science. The tone is somewhat conversational, making the book very appropriate for self-study.

This conversational style however impacts structure, not the least due to the choice of the authors to leave the numbers out of section titles. For instance, subsections have the same typography as sections but for the use of italic. A more confusing example is the "Proof Systems" section, which is only one 6-line-long paragraph, while it probably was meant as an encapsulating section for the remainder of the chapter. This, again, makes the book slightly more appropriate to read cover to cover rather than to use as a reference.

In conclusion, this is a great book, both in terms of contents and style. I believe that its streamlined and welcoming presentation makes it relevant to advanced undergrads and graduate students alike, while the more advanced topics will also be of interest to researchers with prior knowledge in communication complexity.