The Book Review Column [1]

by **Nicholas Tran** (ntran@scu.edu)
Department of Mathematics & Computer Science, Santa Clara University

1 Notable new releases

Leslie Valiant, the father of computational learning theory, has written a new book, *The Importance of Being Educable: A New Theory of Human Uniqueness* (Princeton University Press, 2024), which proposes "educability" as a potentially more scientific theory of human cognition.

Essays on Coding Theory by Ian F. Blake (Cambridge University Press, 2024) is an accessible introduction to modern developments in coding theory. The author was awarded the IEEE Millennium Medal for his work in this area.

Theorems of the 21st Century, Volume 1 by Bogdan Grechuk (Springer Nature Switzerland, 2019) deposits the first installment of important yet approachable results gleaned from the *Annals of Mathematics* in the first decade of the twenty-first century, from the point of view of an International Mathematical Olympiad gold and silver medalist.

2 This column

Sarvagya Upadhyay relishes the wealth of information in *Edsger Wybe Dijkstra – His Life, Work, and Legacy*, edited by Krzysztof R. Apt and Tony Hoare. This insightful retrospective of the Humble Programmer's career includes his Turing Award speech and selected papers, as well as biographical essays and technical perspectives from his distinguished peers.

David J. Littleboy finds a gem in *Introduction to Proofs and Proof Strategies* by Shay Fuchs. Its admirable conciseness and excellent presentation make this undergraduate textbook an ideal introduction to mathematical rigor for both mathematics and computer science majors.

Bill Gasarch rediscovers his inner child when jointly reviewing *Arm in Arm* by Remy Charlip and *Once Upon a Prime: The Wondrous Connections between Mathematics and Literature* by Sarah Hart. These two books explore the roles of mathematics in literature, children's and adult.

Shoshana Marcus lauds accessibility and timely content in *Data Science in Context* by Alfred Spector, Peter Norvig, Chris Wiggins, and Jeannette Wing. She believes the emphasis on ethical considerations in this book is particularly relevant to the issues facing data-driven AI such as ChatGPT today.

3 How to contribute

Summer is here; why not read a book and write a review for SIGACT News? Either choose from the books listed below, or propose your own. In either case, the publisher will send you a free copy of the book. Guidelines and a LaTeX template can be found at `https://algoplexity.com/~ntran`.

BOOKS THAT NEED REVIEWERS FOR THE SIGACT NEWS COLUMN

Algorithms, Complexity, & Computability

1. Knebl, H. (2020). *Algorithms and Data Structures: Foundations and Probabilistic Methods for Design and Analysis.* Springer.
2. Chen, H. (2023). *Computability and Complexity.* The MIT Press.
3. Ferragina. P. (2023). *Pearls of Algorithm Engineering.* Cambridge University Press.
4. Esparza, J., & Blondin, M. (2023). *Automata Theory: An Algorithmic Approach.* The MIT Press.

Miscellaneous Computer Science & Mathematics

1. Grechuk, B. (2019) *Theorems of the 21st Century.* Springer.
2. Nahin, P. (2021). *When Least Is Best: How Mathematicians Discovered Many Clever Ways to Make Things as Small (or as Large) as Possible.* Princeton University Press.
3. Chayka, K. (2024). *Filterworld: How Algorithms Flattened Culture.* Doubleday.
4. Valiant, L. (2024). *The Importance of Being Educable: A New Theory of Human Uniqueness.* Princeton University Press.

Data Science

1. Hrycej, T., Bermeitinger, B., Cetto, M., & Handschuh, S. (2023). *Mathematical Foundations of Data Science.* Springer.

Discrete Mathematics and Computing

1. Harchol-Balter, M. (2023). *Introduction to Probability for Computing.* Cambridge University Press.
2. Ross, S., & Peköz, E. (2023). *A Second Course in Probability.* Cambridge University Press.

Cryptography and Security

1. Tyagi, H., & Watanabe, S. (2023). *Information-Theoretic Security.* Cambridge University Press.

2. Blake, I. (2024) *Essays on Coding Theory.* Cambridge University Press.

Combinatorics and Graph Theory

1. Beineke, L., Golumbic, M., & Wilson, R. (Eds.). (2021). *Topics in Algorithmic Graph Theory* (Encyclopedia of Mathematics and its Applications). Cambridge University Press.

2. Landman, B., Luca, F., Nathanson, M., Nešetřil, J., & Robertson, A. (Eds.). (2022). *Number Theory and Combinatorics: A Collection in Honor of the Mathematics of Ronald Graham.* De Gruyter.

Review of [1]

Edsger Wybe Dijkstra – His Life, Work, and Legacy
by **Krzysztof R. Apt and Tony Hoare** (Editors)

ACM Books and Morgan & Claypool Publishers, 2022
Paperback, 550 pages, $72.00

Review by **Sarvagya Upadhyay**
(supadhyay@fujitsu.com)
Fujitsu Research of America
4655 Great America Parkway, Suite 410
Santa Clara CA 95054, USA

1 Overview

Edsger Wybe Dijkstra (1930–2002) was a Dutch computer scientist known for his significant contributions to theoretical computer science, programming, and software engineering. Always interested in mathematics and physics, Dijkstra briefly explored theoretical physics before ultimately completing his doctoral studies in computer science.

Dijkstra is widely recognized for his eponymous single-source shortest-path algorithm, formal advancements in computer programming, and foundational work in program verification and structured programming. One of his most famous stances is his vehement opposition to GOTO statements, a view that is widely accepted by generations of programmers that followed him and demonstrates his commitment to clear and structured code.

Dijkstra's contributions to computer science earned him numerous accolades, most notably the ACM Turing Award in 1972. His citation reads "for fundamental contributions to programming as a high, intellectual challenge; for eloquent insistence and practical demonstration that programs should be composed correctly, not just debugged into correctness; for illuminating perception of problems at the foundations of program design." His influence is so profound that there's an award named after him, the Edsger W. Dijkstra Prize in Distributed Computing.

This book showcasing his impact involves 31 computer scientists from diverse fields touched by his research, and is divided into five parts. The first part features Dijkstra's Turing Award speech, "The Humble Programmer", offering a glimpse into his philosophical and methodological approaches to programming and problem-solving. Following this, the next two parts delve into his specific contributions, with the second part exploring various facets of his research from the point of view of contemporary researchers, while the third part includes nine of his seminal publications. The fourth part delves into Dijkstra's life through biographical essays, highlighting his influence on the field and personal anecdotes that shed light on his character. Finally, the fifth part discusses Dijkstra's role as a lecturer, the E. W. Dijkstra archive, and provides concise professional biographical information.

2 Summary

As mentioned in the previous section, this book consists of five parts and includes contributions from 31 computer scientists. The Preface of this book states that five contributors are themselves recipient of the prestigious ACM Turing award.

Part I: The Humble Programmer The book opens with Dijsktra's Turing award speech. I have revisited these 18 pages on a few occasions. It reflects his knack for communicating his thoughts in a clear and accessible manner. I really loved going through these 18 pages. Dijkstra received the Turing Award during the nascent stages of computer science, amidst a digital revolution. And although he may have not envisioned the evolution of different computing paradigms, as a quantum computer scientist, I find his speech highly relevant to the ongoing quantum revolution. This starts with the first page itself, where he discusses how his hardware counterparts could showcase a physical device and explain its workings to showcase their professional competence. I see quantum computing in a similar early stage. For instance, displaying a superconducting quantum computer, resembling an exquisite chandelier, can awe visitors. Yet, a researcher or developer working on quantum applications may not impress to the same extent.

Beyond biased perspectives from a quantum computer scientist, Dijkstra's speech contains valuable insights still applicable in today's computing landscape. For instance, his critique of the design of "third-generation computers" in the mid-sixties prompts us to consider optimizing hardware devices for maximum benefits. The concept of hardware accelerators or specialized processing units like GPUs aligns with Dijkstra's vision of computing design benefiting us through both raw computing power and ease of application development.

Finally, I find Dijkstra's vision of the future of programming particularly fascinating and somewhat prophetic. Each generational shift in programming and what can be achieved through it has historically been disruptive, catching the world off guard and eliciting a sense of awe. Dijkstra accurately predicted the necessity of such dramatic revolutions in programming, and the world has since recognized and adapted to these seismic shifts. Economic factors have also driven the need for greater programming efficiency. Additionally, the profession seems to have evolved towards what Dijkstra envisioned as "intellectually manageable" programming.

There are several thoughts that crosses through your mind while reading through his speech, and I will leave it to the readers to draw their own conclusion. It is rich in anecdotal history and wisdom that is still relevant to this era.

Part II: Technical Perspectives on Dijkstra's Work This part comprises of nine chapters each discussing a different aspect of Dijkstra's work.

1. This first chapter focuses on Dijkstra's single-source shortest-path algorithm, described by Mikkel Thorup in his own words as "one of the most famous and broadly used greedy algorithms." Thorup delves into the algorithm itself, the subsequent work it inspired over the past sixty years, and the different implementations. Despite the algorithm's simplicity and clarity, along with its relevance demonstrated through comparative analyses with newer works, I find the chapter quite dense. It contains a substantial amount of information into just five pages. I would have preferred a more extensive treatment of the topic.

2. The second chapter is written by Butler Lampson on programming concurrent systems. It delves into Dijkstra's influential ideas from the 1960s aimed at mitigating the inherent nondeterminism of concurrency. As mentioned in the chapter, Dijkstra thought about these topics when machines had only one CPUs and concurrency was almost non-existent. The chapter briefly explores concurrency in CPUs and distributed systems and how the principles such as atomicity and commuting can alleviate nondeterminism in concurrent operations.. It finally gives the classification on ways to program concurrency.
3. The third chapter, authored by Leslie Lamport, focuses on concurrent algorithms. Dijkstra published five papers on concurrent algorithms, which this chapter explores in some detail. The problems that Dijkstra developed concurrent algorithms for are listed in chronological order: (i) mutual exclusion problem that asks for synchronization among processes to perform critical sections of a program, (ii) self-stabilization devoted to fault-tolerance in concurrent algorithms, (iii) on-the-fly garbage collection that will perform garbage collection concurrently with the main program as opposed to the traditional method of doing it by interrupting the program, and (iv) termination detection in distributed computing. Lamport emphasizes Dijkstra's pivotal role as the driving force behind these papers, despite some of them being of collaborative nature with co-authors, and refers to each one of them as Dijkstra's algorithm.
4. The fourth chapter in this part of the book discusses on Dijkstra's work on self-stabilization. The fundamental problem that Dijkstra thought on was how can a distributed system attain correct behavior after being in an incorrect state. This chapter is authored by Ted Herman where he presents a description of Dijkstra's self-stabilizing algorithm, the impact of the work, and finally his thoughts on its significance. I enjoyed reading this chapter especially the conclusion and Herman's lighthearted reasoning of why "Dijkstra's project on self-stabilization was superbly successful."

4. The fifth chapter, written by Reiner Hähnle, touches upon the topic of program verification and Dijkstra's central role in this topic. Dijkstra's usage of the term *program verification* referred to the correctness of a program with respect to all possible executions. It starts with Dijkstra's thoughts on program verification and critique by researchers in the field. Dijkstra favored *correctness-by-construction* approach as opposed to *post hoc verification*, where the program is verified after construction. The chapter focuses on his most important contribution, the weakest precondition calculus. There are some interesting points that Hähnle raises in view of Dijkstra's legacy in this field, which is the final section of this chapter.
5. The sixth chapter by David Gries delves into Dijkstra's work into program correctness from both an anecdotal and technical perspective. Dijkstra's thought along the lines of how to write a program and its proof of correctness simultaneously as part of his program development process. Gries outlines four key themes that motivated Dijkstra's pursuit of correct programs based on their correspondence. Subsequent sections explore Dijkstra's work on structured programming interspersed with quotes from several of his scholarly work, his participation in 1968 NATO Conference on Software Engineering and reflections on the event, and his personal state of mind during a period when he "wrote over 25 EWDs (technical reports) that touched upon program development." The author then highlights a few of Dijkstra's work relevant to program correctness.
6. The seventh chapter, co-authored by Krzysztof R. Apt (one of the editors of the book) and Ernst-Rüdiger Olderog reviews "Dijkstra's contribution to nondeterminism by discussing the

relevance and impact of his guarded commands language proposal." The authors start with explaining nondeterminism in computer science and continue to discuss about *angelic nondeterminism*, where one identifies successes and failures, and only the former counts. This type of nondeterminism is heavily studied and continues to have great impact in computational complexity theory research. The authors subsequently explore guarded commands, and illustrate computing maximum of two numbers and Euclid's greatest common divisor algorithm in favor of guarded commands. They go on to describe how concurrent programs and nondeterministic programs can be linked via guarded commands and several other concepts related to them. They also explore subsequent developments influenced by Dijkstra's work.

7. In this chapter, Christian Lengauer explores Dijkstra's views on software construction through personal reflections and correspondences with the man himself. The chapter opens with anecdotal accounts that offer insights into Dijkstra's personality, including a memorable image of them on a tandem bike in Nuenen. Lengauer delves into Dijkstra's strong disagreements with Backus, his preferences such as favoring textual representations over pictorial ones, and his opinionated stance, which could be both advantageous and detrimental to his peers. The chapter concludes on a poignant note with Dijkstra's final words to Lengauer.

8. This chapter, co-authored by Robert van de Geijn and Maggie Myers, begins with the a few interactions between the former author and Dijkstra. They discuss about the goal-oriented programming advocated by Dijkstra and how it can be utilized to discover new algorithms in dense linear algebra. As an example, the authors first give a simple in-place algorithm for inverting an upper triangular matrix and how by "systematically identifying multiple loop invariants", a highly parallelizable alternative to the simple algorithm can be derived. The authors then summarize the impact of Dijkstra's and his peers on architecture and software development of numerical algorithms.

9. In the ninth chapter, focusing on Dijkstra's technical contributions, delves into calculational proofs. Penned by Vladimir Lifschitz, the chapter commences with a letter from Dijkstra illustrating how natural number properties can be demonstrated using the calculational style. This approach to proofs was detailed in Dijkstra's book "Predicate Calculus and Program Semantics," which he co-authored with Carel Scholten. Although I have encountered several concepts discussed in the book thus far, calculational proofs were entirely new to me. However, after going through the chapter, I can say that I now know more about calculational proofs.

10. Jayadev Mishra discusses Dijkstra's broad range of interests in mathematics and how he expressed his ideas and proofs with exceptional elegance in a series of notes known as EWDs. In an earlier chapter, Lengauer mentioned Dijkstra's fondness for fountain pens, and Mishra's chapter showcases some of Dijkstra's beautifully handwritten notes done with a fountain pen. Mishra focuses on a few theorems where Dijkstra came up with a very elegant argument to prove them. I was pleasantly surprised by his graph-theoretic arguments to prove Fermat's and Wilson's theorems in number theory, and the AM–GM inequality. This chapter was a delight to read.

Part III: Selected Papers This part contains nine of the most influential papers authored and co-authored by Dijkstra. All of the papers have been discussed in the previous part.

Part IV: Biographical Essays This section comprises several chapters authored by computer scientists who had personal and professional relationships with Edsger Dijkstra. Notable contributors include Krzysztof R. Apt, E. Allen Emerson, David Gries, Tony Hoare (the co-editor of the book), Brian Randell, and Fred B. Schneider. A dedicated chapter explores how Dijkstra was perceived by his friends, colleagues, and students. Contributors to this chapter include Lex Bijlsma, Ken Calvert, K. Mani Chandy, Eric C. R. Hehner, Wim H. Hesselink, Rajeev Joshi, Don Knuth, Alain J. Martin, Jayadev Mishra, David A. Naumann, J. R. Rao, Hamilton Richards, Mark Scheevel, and David Turner. The personal reminiscence and reflections from these researchers highlight the profound impact Dijkstra had on generations of computer scientists. I highly recommend reading these accounts as they portray a figure universally admired and respected.

Part V: Varia This is final part of the book consisting of two chapters. The first chapter in this part is devoted to Dijkstra as a key lecturer and director of the Marktoberdof summer schools. The first summer school was held in 1970, and Dijkstra continued to be the director until 2000. Manfred Broy discussed in detail about these summer schools, and after reading it, I wish we had more such events that reach out to a wider audience. The final chapter (Hamilton Richards) is about the Edsger W. Dijkstra archive, an online repository of Dijkstra's unpublished manuscripts.

3 Opinion

It is evidently clear to me after having gone through the book that preparing this book was a monumental task. I enjoyed reading the book. My favorite parts of the book was Dijkstra's Turing award speech and the biographical essays contributed by various researchers in the field. This differs significantly from a standard biography, where the perspective is usually limited to one or maybe two authors. The multiple biographical essays of Dijkstra offer diverse portraits, enriching the experience for readers curious about his life.

Dijkstra started working as a programmer way before programming was even considered a profession. He even alluded to it in his speech when discussing his wedding with his wife Ria and how his profession was changed to "theoretical physicist" by the municipal corporation of the town of Amsterdam. His impact on the field is profound, as emphasized in the technical chapters dedicated to his work. However, these chapters aren't just overflowing with praise; they also critically analyze his views, scholarly contributions, and thought processes. This critical perspective is expected from the authors of these chapters. They demonstrate the research trajectories Dijkstra initiated – some widely adopted and acknowledged, while others not receiving as much attention. It goes without saying that Dijkstra's writings were the result of meticulous thought. These chapters also underscore the inherent humility in research. They highlight both the successful and not so successful research endeavors of Dijkstra.

The technical chapters can be daunting for a lot of readers. The authors have made significant effort to write the chapters in a way that could reach to a wider audience in computer science. Readers should not read this book in anticipation of understanding everything. Rather they should read it to celebrate Edsger W. Dijkstra's life and work, and perhaps gain insights into how the

brilliant mind of his worked. I would like to end this review by mentioning few quotes of Dijkstra that are pertinent even today. The below quote can be found in Chapter 10 of the book.

> Simplicity is a great virtue, but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.

Or his views on the evolution of computer science as a field during his lifetime.

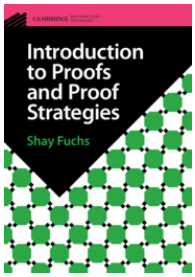
> Computer science is no more about computers than astronomy is about telescopes.
>
> The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.

Review of [1]

Introduction to Proofs and Proof Strategies

Shay Fuchs

Cambridge University Press, 2023
USD 44.99, Paperback, 349 pages



Review by
David J. Littleboy
(djl@alum.mit.edu)
Technical translator, retired
Tokyo, Japan

1 Overview

This book is an undergraduate mathematics textbook that could (and should!) be used in undergraduate computer science curricula. It's that good.

First, the background. Mathematics education, from secondary through pre-calculus and through calculus itself is mainly concerned with solving problems by algebraic manipulations and is only secondarily concerned with foundational issues and formal proofs. Mathematics educators concerned with undergraduate pure math education worry that students who enjoy and are good at those sort of problems may have trouble dealing with foundational issues and reading and writing proofs. One approach to this is Lara Alcock's *How to Study for a Mathematics Degree* [1], which explains to the student reader what life will be like as an undergraduate mathematics students, and includes an extensive bibliography of work in this area as well as recommendations for further reading. While excellent, it prepares the student more psychologically than mathematically. Fuchs, on the other hand, prepares undergraduate students to handle the sorts of proofs they will be faced with in upper-level mathematics courses. As such, it also can serve that purpose for undergraduate computer science majors, who will be required to exhibit far greater mathematical sophistication than this reviewer (MS, Comp. Sci. '84) was.

More recent undergraduate computer science texts such as Lehman [4] and Gossett [2] do cover this material, although not as thoroughly. At the back of my mind is the concern that presenting this material as part of a larger book/course may implicitly deprecate it somewhat and fail to emphasize that this material applies to all of mathematics (including real and complex analysis) as well as the mathematics used in computer science. So it seems to me that dedicating a term to this material makes sense.

There are several reasons I like this book, but the main one is that it includes a large number of problems that drill the student thoroughly on the material covered. In addition, it is laser-focused on the necessary material while avoiding even mentioning the unnecessary. It may seem surprising that an undergraduate text that prepares students for number theory, abstract algebra, topology

and the like does so without even mentioning those fields, but the main body of this book does exactly that. (To get ahead of myself, Part II of this book shows how the proof techniques presented are used in combinatorics, real analysis, complex numbers, and linear algebra. And includes a fair number of problems in these areas.) The tight focus and avoidance of extraneous material allows the author to keep this book to an extremely svelte 342 pages, while the main section itself is a mere 204 pages. In this day and age of the enormous tome textbook, this is a real pleasure.

2 What's in the Book?

This book consists of two parts, the first being the main material and the second the additional topics mentioned above to show the use of the material presented. The organization of the material presented in the first part is particularly well thought out. For example, one might think that in presenting the use of sets in mathematical proofs, relations would come before functions. But the author presents relations as the last chapter in Part I. This allows him to include longer discussions of equivalence relations and how modular arithmetic defines equivalence classes in that chapter.

Chapter 1 introduces proofs using the pre-calculus/high-school algebra potential math and computer science majors will be comfortable with: the quadratic formula, inequalities, various types of means. But now the student must read and write proofs of results. The text includes in-line exercises related to the material being presented, and provides solutions to all of these exercises at the end of the chapter. Both the exercises and the end-of-chapter problems range from simple to quite difficult. Chapter 2 introduces sets, functions, and the field axioms, although it puts off introducing injections, bijections and the like for another chapter. Here, the field axioms are introduced independently, that is, not as part of abstract algebra (i.e., groups, rings, fields). This strikes me as a good idea, since the point is to introduce the use of axioms in proofs.

The short Chapter 3 "Informal Logic and Proof Strategies" is the core of this book. The basic idea here is that proofs can be classified into one of three types: direct proof, proof by contrapositive, and proof by contradiction (leaving proof by induction for a later chapter). While [4] also lists proof by enumeration as a proof type, this book deals with that as a special case occurring in certain situations, since each resulting case is an independent proof. As an older reader, and one trained in AI of the 1970s and 1980s, I had hoped for more here, that is for a classification of the sorts of manipulations that get one from the claim to the result, e.g., completing the square, parity arguments (e.g., dominoes on a chess board). (Along these lines, Fuchs, however, explicitly notes where the induction step is used in inductive proofs.) But given the purpose of and main audience for this book, I think the author has the right idea: for students learning how to write proofs for the first time, this is the classification they need to know.

Fuchs, however, implicitly rejects the idea of classifying proofs further, arguing that beyond the major categories, proof is an art, not a science. He writes: "Is there also an algorithm for generating proofs? Unfortunately, no. In fact, I would rather say: Fortunately, no!". He goes on to argue that it is this creative and artistic nature of mathematics that makes it so attractive.

Although this book was developed and written before the explosion of generative AI, the argument for mathematics as art, in my opinion, does speak to the excessive enthusiasm of the generative AI researchers who believe that LLMs will be able to do and contribute to higher mathematics.

This chapter also exhibits the laser focus that I like about the book: the introduction to logic as used in proofs only introduces the necessary and key ideas and techniques. It's not an extensive, wide-ranging overview of the field of logic in general; it's the parts of logic you absolutely need to

have an operational understanding of, and exercises are provided to make sure that you do acquire that operational understanding.

Aside: Despite its laser focus, this book goes into lovely detail on two problems whose treatment in other sources have irritated me. The first is the standard proof of the irrationality of the square root of 2. A common way of expressing this proof depends on evenness, which is because it's the square root of 2. This is not incorrect, but it doesn't work for the square root of 3 (or anything else for that matter). But Fuchs presents a more general proof and has several related problems. A computer science text I reviewed a while ago claimed that if N is the product of the first k primes, then $N + 1$ will be prime, which is incorrect, since it could be composite. Fuchs not only does not make this mistake, he goes into this in more detail, again, in the problems. It really is the problems that make this book.

The next four chapters cover mathematical induction, bijections and cardinality, integers and divisibility, and relations. As always, each chapter has text-related exercises with solutions, and a good number of end-of-chapter problems drilling the material. (Aside: In the margin next to a formula that appeared in the text I wrote "Prove me", since it wasn't immediately obvious. Proving that formula was one of the in-text exercises a few paragraphs down.)

I claim, perhaps unreasonably, that this text should be used in your computer science curriculum. The zeroth point is, of course, that other than using quite a few examples and problems involving pre-calculus algebra (since the author does have real analysis in his sights), the material is pretty much exactly what the student will need to do upper-level undergraduate computer science courses. And, again, in this day and age, being extremely comfortable with abstract algebra and being ready to handle arguments in number theory and linear algebra and other advanced mathematical areas will be necessary to deal with cryptography, AI (that is, SIMD algorithms), and quantum computing.

But my main point is that you are going to have to teach this material anyway, so doing it in an independent course would make sense. And this is a good way of doing that.

Are there better ways? Perhaps. MIT offers a math course for computer science majors that covers some of this material plus a lot more, and the textbook [4] is (or at least was) available as a free, 900-page, pdf download. The first 282 pages cover some of the material in this book with, of course, computer science material included. But there are fewer problems, and they don't include drill sorts of problems. A student who had taken a course based on Fuchs would be able to breeze through much of the first part of this text/course and focus on the CS-specific material. (Aside: this freebie textbook is also seriously great. But because it doesn't drill that thoroughly, it's going to be hard for many students.)

3 The Writing

The author, in addition to teaching mathematics at the university level, has credentials and experience in secondary education teaching, and it shows in the writing. It is respectful of the reader yet points out issues that the reader may find difficult or confusing. Unlike another text I recently reviewed, in which there was something that irritated or angered on nearly every page, the writing here doesn't intrude or irritate. That may sound like faint praise, but it's not: the writing is exactly what is needed.

4 Criticisms

Nothing in this world is perfect, and reviews are supposed to be critical, so here goes. First, I'd like there to have been more problems that involve summation and product notation. Upper-level computer science texts such as [3] can be really dense of these, and more practice would be useful were this text to be used in a computer science context. Second, no solutions are provided for the end-of chapter drill problems, only solutions to the example problems used in the text are given. A few solved problems at the end of each chapter would make this book more useful for self study. Finally, the index is a tad sparse. Computer science is mentioned in a couple of places, but doesn't appear in the index. Also, I wanted to look up the problems on proving irrationality, but "irrational" didn't index them.

This book does not include a bibliography, but I don't think it needs to. A listing of the author's favorite mathematics books and textbooks, or a suggested further readings list, would be nice to have, but isn't necessary. (Again, the bibliography and further reading recommendations in Alcock [1] are excellent.)

5 Conclusions

To my eye, ear, and sensibility, this book succeeds in doing what it sets out to do: to teach students the fundamentals of actually doing mathematics at a more advanced and abstract level than they have seen. Its organization of topics works well and I can't find fault with the material covered. (Aside: It is designed for a course taught at the first-year undergraduate level, presumably in parallel with first-year calculus, and as such, avoids the use of examples from calculus, only mentioning it in two or three places.) The large number of problems means that the students will have extensive experience with these new concepts by the end of a course. Since all of the concepts presented are used in computer science, it could provide the basis for a mathematics for a computer science course as well.

References

[1] Alcock, L. *How to Study for a Mathematics Degree*, Oxford University Press, 2013.

[2] Gossett, E., *Discrete Mathematics with Proof*, Wiley, 2nd. ed., 2009.

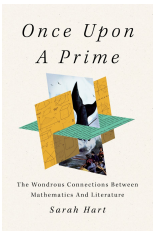
[3] Graham, R. L., Knuth, D. E., Patashnik, O. *Concrete Mathematics*, Addison-Wesley, 2nd. ed., 1994.

[4] Lehman, E., Leighton, F.T., Meyer, A. R. *Mathematics for Computer Science*, Creative Commons, 2015.

Joint Review of [1]

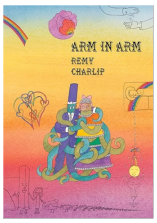
Once Upon a Prime:
The Wondrous Connections Between
Mathematics and Literature
by **Sarah Hart**

Publisher: Flatiron Books
$14.99 Kindle, $18.99 paperback, $22.74 Hardcover
290 pages. Year: 2023



and

Arm in Arm
by **Remy Charlip**



Publisher: NYRB Children's Center
$19.95 hardcover, Much cheaper used. 40 pages
Year: Originally 1969. But Also 2019

Reviewer: **William Gasarch** (gasarch@umd.edu)

1 Introduction

In 1970 I read *Arm in Arm*. It was a delightful book (for a nine-year old), full of word play and logic. Either I liked the book because I liked math, or I liked math because I liked the book. The cliché comment is to ask, which came first, the chicken or the egg? Two comments on that:

1. This cliché is no longer accurate, since *the chicken came first* (see https://nymag.com/intelligencer/2010/07/chicken-egg_mystery_allegedly.html). In any case, I like math and I like the book.

2. The book *Arm in Arm* has a picture of a chicken and an egg with the following conversation:

   Chicken: Who was first, me or you?

   Egg: Don't question it, be grateful we have one another.

In 2023 I read *Once Upon a Prime: The Wondrous Connections Between Mathematics and Literature*. To briefly describe it I can't do better than the title itself.

Why review these books together? *Once Upon a Prime* talks about, say, how Hamlet has narrative distance since there is a play-within-a-play. *Arm and Arm* uses recursion, which is similar. More generally, *Arm and Arm* is an example of some of the points the book makes. Note that *Arm and Arm* is a 40-page illustrated book for children (recommended for ages 6–9 though I think it should be 6–12 plus a sixty-year old reliving his youth); by contrast, *Once upon a Prime* is about serious literature.

2 Once Upon a Prime

2.1 Summary of Contents

The book has three Parts. Part I has 4 chapters, Part II has 3 chapters, and Part III has 3 chapters.

For each chapter I discuss a few points it makes. Each chapter makes far more points than I discuss.

Part I: Mathematical Structure, Creativity, and Constraint

Chapter 1 (titled *One, Two, Buckle My Shoe: The Patterns of Poetry*) considers poetry.

1. How would you describe a limerick? The rhyming pattern is AABBA. This chapter gives a delightful example of a way to generate 3^5 limericks by having a choice of 3 for each line. All 3^5 limericks make sense!

2. Some Sanskrit poetry has a limit on the length of each line. But what is length? Some syllables are light and count 1 for length, and some are heavy and count 2 for length. How many patterns for n-length lines are there? The reader can work out that it is the nth Fibonacci number. This is not so much an *application* of Fibonacci numbers as one of the *origins* of Fibonacci numbers, which predates Fibonacci by a few hundred years, or a thousand years, depending on who you count. This is briefly discussed. A while back I did a blog post on the origins of Fibonacci numbers (see `https://blog.computationalcomplexity.org/2021/12/did-lane-hemaspaandra-invent-fib-numbers.html`).

Neither of the points above are about the *content* of the poem. We present two poems where the content is mathematical.

From Chapter 1:

As I was going to St. Ives
I met a man with seven wives
Each wife had seven sacks
Each sack had seven cats
Each cat has seven kits
Kits, cats, sacks, and wives
How many were going to St. Ives?

I leave it to the reader to solve it. Hint: you can do it without any calculation.

The following poem was not in the book, but I include it since I like it:

A challenge for many long ages
Had baffled the savants and sages
  Yet at last came the light
  Seems that Fermat was right
To the margin add 200 pages

(I actually do not know who first wrote this. If you do, please email me.)

Chapter 2 (titled *The Geometry of Narrative: How Mathematics Can Structure a Story*) discusses how a story can have a mathematical structure.

1. A story can have parts that are happy and sad. Think of the classic *Boy meets Girl*, *Boy loses Girl*, *Boy regains Girl*. That can be thought of as a function that goes up then down again and then up again. There are more complicated structures as well.

2. The book *The Luminaries* by Eleanor Catton has 12 chapters. For $1 \leq i \leq 12$, chapter i has $13 - i$ sections. Hence, for all $1 \leq i \leq 12$,

   $$\text{Chapter Number} + \text{Number of Sections} = 13.$$

   Also, each chapter is half the size of the previous one. The book has some interesting numbers like 4096 ($4096 = 2^{12}$). The book won the Booker Prize.

The Luminaries raises the following point: does using math to structure a book make it a better book? For *The Luminaries* the structure and the plot and the characters are interrelated so the answer is YES, and the fact that the book won the Booker Prize is very strong evidence that it's a great book.

Chapter 3 (titled *A Workshop for Potential Literature: Mathematics and the Oulipo*) begins talking about a group (or perhaps a set) of people who, in 1960, formed *L'Ouvroir de littérature potentielle*, or *Oulipo* for short. This translates roughly to *Workshop for Potential Literature*. Their goal was to invent new structures for literature. This chapter then looks at ways to structure literature, some of which came from this group.

One way is to put constraints on what one writes. *La Disparition* and *A Void* are a French book and its English translation. Neither one uses the letter e (the translation must have been really hard). *Gadsby* is a book (in English) which does not use the letter e. This chapter describes why, for *La Disparition*, this constraint added to the book (hence it was not just a gimmick); however, for *Gadsby* it added nothing (hence it was just a gimmick). Far more complicated structures and constraints, for other books, are discussed. One uses the Fano plane. So there is some serious math here.

While reading this chapter I thought they have it backwards: rather than come up with a structure (perhaps a gimmick) to write a book (e.g., the ith chapter has i^2 sections), first write your book and then see how you can structure it better to reflect the plot or characters. This chapter does point out that sometimes these structures are just a gimmick.

Chapter 4 is titled *Let Me Count the Ways: The Arithmetic of Narrative Choice*. True Story: I was at the Worlds Fair in Montreal in 1968. One of the exhibits was that you watch a movie, but at certain points in the movie the audience voted on how a crucial plot point would go. I was annoyed since my vote usually lost, and so I never did find out how my choices would have worked out. There are works of literature that use the same principle. This chapter discusses those and other mechanisms where a story may actually be several stories.

Part II: Algebraic Allusions: The Narrative Uses of Mathematics

Chapter 5 (titled *Fairy-Tale Figures: The Symbolism of Numbers in Fiction*) explores the following question: Why do some numbers occur in literature more than others? Note: 3 (3 wise men, 3 wishes, Goldilocks and the 3 bears, 3 witches in Macbeth, the Trinity), 7 (7 Dwarfs in Snow White, the 7th seal, 7 deadly sins), 12 (12 tribes of Israel, 12 Apostles), 40 (Ali Baba and the 40 thieves). There are also large numbers that appear a lot: 100, 1000, and sometimes 99 or 999.

This chapter gives intelligent speculation about why these numbers occur so often. This is not a scholarly study; however, I am glad about that. A scholarly study would be more boring and just as speculative.

Chapter 6 (titled *Ahab's Arithmetic: Mathematical Metaphors in Fiction*) points out that many works of literature have actual math in them if you know to look for it. *Moby Dick* actually talks about cycloids, so this is strong evidence that Melville knew some math. *Daniel Deronda* by George Eliot talks about statistics. There are many other examples given. None of these books have mathematicians or scientists as characters.

Chapter 7 (titled *Travels in Fabulous Realms: The Math of Myth*) *uses* math and science to determine if descriptions of mythical creatures can be true, e.g., Lilliputians (6 inches tall), Brobdingnag (72 feet tall), various space aliens. Largely the answer is NO. The chapter also looks at the size and shape of real animals to determine if they could be larger or smaller.

Part III: Mathematics Becomes the Story

In Chapter 6 it was noted that some works of literature *use* mathematics. However, none of the books discussed were *about* mathematicians or math. Chapter 8 (titled *Taking an Idea for a Walk: Mathematical Concepts so Compelling They Escape into Fiction*) discusses books where math plays a central role. Often the math is used since it was somehow in the public mind at the time. These include the following.

1. *Flatland* by Edwin Abbott: This book is about a society which is two-dimensional. It is a satire of Victorian England.

2. *The Time Machine* by H.G. Wells: Time is the fourth dimension, and the machine is able to travel in that dimension.

3. *Jurassic Park* by Michael Crichton: DNA from the past is used to create dinosaurs. Chaos theory and fractals are used and are an important part of the story.

These books are about serious math, and this chapter explains this serious math very well. Even though the readers of this column are mathematically sophisticated, they might learn some math from this chapter that they didn't already know.

Chapter 9 (titled *The Real Life of Pi: Thematic Mathematics in the Novel*) discusses how math can be an underlying theme in some books. The constant π (actually approximations to it) is used in *Life of Pi*. Then it goes on to discuss the book *The Library*, which leads to some intricate combinatorics and geometry. Then Lewis Carroll's *Alice in Wonderland* and *Through the Looking Glass* are discussed (note that Lewis Carroll was a mathematician), since they use some math and physics. Finally it discusses *The Hitchhiker's Guide to the Galaxy*.

Chapter 10 (titled *Moriarty was a Mathematician: The Role of the Mathematical Genius in Literature*) considers how mathematicians are portrayed in literature. They are often geniuses. They are often unemotional, which is not accurate. This chapter gives pointers to lots of books that portray mathematicians, some quite well.

Who should read this book?

For a mathematician who wants to know how literature both used math and can be analyzed by math, this is a great book. There is even some math that a mathematician might not know. After reading it, a mathematician may want to read some of the books mentioned.

What if someone does not know any math? The book would be rough in spots. After reading it, a non-mathematician may want to look up some math.

3 Arm in Arm

Arm and Arm has lots of cute poems and very short stories and ... hard to say what it really is. I gave one example in the introduction (the Chicken and Egg conversation). I'll give one more. There are two pictures on two pages that face each other. Note that the book is illustrated, so unless SIGACT News wants to pay for color printing and an artist, I must make do with telling you about the picture, and giving you the words.

1. One of the pictures is mostly snow and 6 children. There is also a window, presumably of a house. One of the children says the following: *Isn't it better to be out in the cold snow saying "Isn't it better to be out in the cold snow rather than in a warm bed?" rather than in a warm bed saying "Isn't it better to be out in the cold snow rather than in a warm bed?"?*

2. The other picture is of a large bed with 6 children. There is a window which indicates that it is snowing outside. One of the children says *Isn't it better to be in a warm bed saying "Isn't it better to be in a warm bed rather than out in the cold snow?" rather than out in the cold snow saying "Isn't it better to be in a warm bed rather than out in the cold snow?"?*

Who should read this book?
This is a great gift for anyone from 6 to 12 years old.

4 Coda

Once Upon a Prime does not discuss the following:

1. The book *Reality Conditions* by Alex Kasman, which is a collection of short stories by a mathematician, about math and mathematicians. I reviewed it (`https://www.cs.umd.edu/~gasarch/bookrev/37-3.pdf`).

2. The book *Riot in the Calc Exam and other Mathematically Bent Stories* by Colin Adams, which is a another collection of short stories by a mathematician, about math and mathematicians. I reviewed it (`https://www.cs.umd.edu/~gasarch/bookrev/41-2.pdf`).

3. Movies and TV shows that have a math theme (e.g., the movie *Proof* in a big way, the movie *Mean Girls* in a small way, and the TV show *Numb3rs* in a big way). I reviewed *Numb3rs* (`https://www.cs.umd.edu/~gasarch/bookrev/37-3.pdf`). Note that *Math in Movies and TV Shows* would be another book.

The above noted omissions are *not* a criticism. Au contraire, the fact that I would like to see books about the math in these venues (perhaps by Sarah Hart) shows that, like a good novelist, she leaves us wanting more.

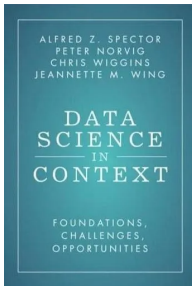
Arm in Arm is short at 40 pages. That I want more is a compliment to the author and illustrator Remy Charlip.

Review of [1]

Data Science in Context:
Foundations, Challenges, Opportunities

Alfred Z. Spector, Peter Norvig,
Chris Wiggins, and **Jeannette M. Wing**

Cambridge University Press, 2022
$39.99, Hardback, 335 pages



Review by

Shoshana Marcus
(shoshana.marcus@kbcc.cuny.edu)
Department of Mathematics and Computer Science
Kingsborough Community College
of the City University of New York

1 Overview

The term *data science* is a buzzword that has garnered much attention in recent years. Do you ever find yourself wondering, *what exactly is data science?* Do you find yourself thinking, *how does data science impact my life?* Do you find yourself considering, *how can data science further improve my life?* Do you find yourself contemplating, *how can I protect myself from the vulnerabilities exposed by data science?* If so, then this book should help you put these new trends into proper perspective!

As this book defines it, "Data science is the study of extracting value from data - value in the form of insights or conclusions." Data science combines advances in computing with the aspirations and methods of statistics and operations research. The process begins with the collection of large data sets, then the data is processed, conclusions are drawn, and the system applies what was learned to new sets of data. This cycle is naturally an ongoing process.

As the authors of this book observe, data science is *transdisciplinary*. A new field has emerged from the interactions between many different disciplines, thus, enabling data science to achieve its theoretical, methodological, and practical results. The field of data science derives primarily from the fields of statistics, operations research, and computing. The nascent field has also been informed by the sciences due to its abundance of applications with datasets so large they are otherwise untenable. The humanities and social sciences have provided perspectives that ensure societal benefits are maximized and harmful effects are kept at bay. This book demonstrates and emphasizes data science's integration of many forms of knowledge, techniques, and modes of thought.

"In this book, four leading experts convey the excitement and promise of data science and examine the major challenges in gaining its benefits and limiting its harm. They offer frameworks for critically evaluating the ingredients and the ethical considerations needed to apply data science productively, illustrated by extensive application examples" (prelude).

In the words of author Peter Norvig (p. 273), "The challenge for machine learning and data science is to build systems that align with society's real needs, and work for everyone. I hope this

book will inspire researchers to develop ideas that contribute to this; will enable developers to build systems that work for the betterment of all; and will empower consumers to know what they can ask for." This book presents a range of perspectives which should hopefully benefit researchers and users of data science alike.

2 Summary of Contents

In the Age of Technology, data science has become pervasive. Data science drives popular software used by billions of people every day, providing new tools, forms of entertainment, economic growth, and potential solutions to difficult and complex problems. These opportunities come with significant societal consequences, raising fundamental questions about issues such as data quality, fairness, privacy, and causation. Through many examples, this book illustrates data science's broad and growing impact, multi-faceted challenges, and its powerful future.

The challenging goal of data science is to build systems that align with society's real needs, and are beneficial to everyone affected. By understanding how things work and exposing vulnerabilities, we are able to increase benefits and reduce risks.

The pace of innovation is astounding and this makes it hard to respond to changes in time. The authors emphasize the ethical concerns of data science throughout this book. Although it is not simple to balance ethical considerations with primary objectives, all data science practitioners bear the responsibility of doing so. As implied by the title, the authors instill the mantra that a successful data scientist considers the context that defines success, expanding the primary focus of problem solving in ways that are effective and efficient. Consumers of data science products and results also play a role in ensuring that ethical considerations are given proper deliberation.

Data science relies heavily on artificial intelligence; big data and machine learning go together. Data science applications rely on algorithms for big data as well as statistical methods for deduction in heuristics. This book is replete with diverse examples across many domains. Some of these applications, such as product recommendation, are used by the population at large. Others applications are used by scientists; the field of computational biology has flourished due to the advances of data science. Scientific modeling problems such as protein folding and genome-wide association studies have become tractable with the tools and models that we have today, equipped to efficiently process enormous sets of data. Data science applications are continually evolving due to growth in data, advances in computational capacity and developments in machine learning.

Data science revolves around a virtuous cycle of increasing usage generating more data that improves quality and garners more usage. The data scientist needs to focus on effective data gathering, modeling, and application, as well as error correction. As this book explains, the route-finding software we have come to rely on (e.g., Waze and Google Maps) are not simply performing the operations research task of finding the shortest path while taking speed limits into account. These sophisticated applications are data driven since their solutions dynamically adapt based on factors that are constantly in flux, such as the current road conditions and historical delays (like rush hour). We would also like the algorithms to take the safety of drivers into account and to consider the privacy of homeowners on quiet streets. Drivers have low tolerance for small errors, such as directing a user to travel on a closed road or to drive the wrong way on a one-way street. When these applications are embedded in self-driving cars, we have even lower tolerance of failure. Errors have extreme legal, ethical, and financial risks.

Spelling correction is one of the first applications discussed at length in this book. Intuiting

the word intended by the typist, when errors are allowed, is not as simple a problem as it seems at first glance. Not only do we expect spelling correction in documents we type, but we rely on these tools to use search engines on the World Wide Web effectively. Speech recognition is more complex and nuanced than spelling correction, and thus is not as simple to formulate or to solve. Music recommendation systems take this to a whole new level and introduce many other complicating factors. It becomes difficult to even intuit the users' preferences and to clearly define the problem. In music recommendation systems, the users' reactions are tracked - how you listen to music, skip around, etc. This brings us to consider privacy concerns.

The Covid-19 pandemic and vaccination controversies form a case study throughout this book. This is a topic that many readers can easily relate to due to its recentness and the abundance of uncertainty and controversy involved. Covid-19 mortality predictions were vastly disparate from actual occurrences. This is certainly humbling and can be understood within context. The data were insufficient and erroneous, the necessary models are inherently complex, the disease is rapidly changing by its nature, and this complexity was exacerbated by feedback phenomena that were catalyzed in part by government actions.

Consumers have come to depend on recommendation systems because the Web has grown so large and no individual can sift through all the information. This benefits the sellers as well since they can reach much larger markets and do more business, which in turn has encouraged many sellers to embark on e-commerce. Similarly, in the financial sector, we have all come to rely on data science to detect fraudulent activity in our checking accounts and credit cards, as well as to aid analysts in predicting stock price fluctuations.

This book introduces the Analysis Rubric for data science, which delineates the major considerations for evaluating data science's suitability for a proposed application. Table 1 delineates the elements of the Analysis Rubric and its application to spelling correction. Spelling correction readily meets the needs surfaced in the Analysis Rubric; this explains why it performs quite well in applications. Speech recognition meets the needs of the Analysis Rubric almost as well, even though it is a more complex problem. Music recommendation is quite challenging for data science, and this is indicated by the Analysis Rubric since the objectives are not even clear. Covid-19 mortality prediction systems did not fare well; applying the Analysis Rubric highlights some of the major interferences.

Tractable data	One can easily procure an appropriate corpus of online text.
Technical Approach	A basic version is relatively simple to code.
Dependability	Privacy and security are not major issues; care must be taken to prevent an attacker from spamming the system with incorrect spellings (perhaps to promote their brand name).
Understandability	Users don't really care how a spelling corrector works, nor does the system need to understand the cause of a spelling error.
Clear Objectives	The clear goal is providing the correct spelling of the word the user meant to type.
Toleration of Failures	Users are accepting if the system does not correct a word's spelling or guesses incorrectly.
Legal, Ethical concerns	Do not seem to pose problems.

Table 1: Analysis Rubric applied to spelling correction (adapted from pages 63-64)

This book devotes a separate chapter to each element of the Analysis Rubric and surveys the challenges in that aspect of applying data science properly. First there is the need for sufficient data of reliable quality. In many applications, gathering data is difficult amidst privacy concerns. This is a fundamental problem in medical research. Additionally, abusers of systems can intentionally misrepresent data. It is important to enforce the most restrictive data access rights as possible to ensure data retains its integrity.

It is interesting to hear anecdotally, in the authors' experience advising companies, that many companies are excited to employ data science because they have the initial data. Yet, they are apprehensive about the mathematical complexity of building a machine learning model. Over time, they often realize that model building may be easier than establishing and maintaining a data pipeline.

Designing and deploying dependable models requires much ingenuity. It is challenging to derive truthful insights and conclusions. The inherent uncertainty poses a significant problem. For some inputs, even the best experts disagree on *the* correct output, so any model will necessarily disappoint some of the time. Another problem is that the world is constantly changing. A system trained on historical data may no longer perform well in the future. It is vital to continuously monitor a deployed system to watch out for any unexpected changes, and to correct for them by updating the model. A third problem is that it can be difficult to specify what we want to optimize in an application. The designer must be careful that inductive bias does not creep in when software makes assumptions to infer generalization from the training data. This can happen easily when hypothesis drives the design of a system.

In order to be adopted by society, a data science application must demonstrate its dependability. It often takes more time and effort to make a system dependable than to collect the data and build the algorithmic model. A chapter of this book addresses four key aspects of dependability: privacy, security, resistance to abuse, and resilience.

A chapter of this book explores the understandability necessary in data science. In some applications, it is not sufficient for the algorithm to make accurate predictions; the users want to understand what is going on. Understandability is important to the developers of the application (to help them do their jobs), to the users (to trust the application), to the general public (for assurance that the application will not harm society), to the regulators (for compliance and accountability), and to the scientific community (to reproduce results or to extend them).

The authors contend that data science done poorly obfuscates facts and truth. In the words of the authors (p. 186), "Every dataset has a story to tell, but we need to tease out that story by clearly explaining where the data came from and what it says, putting it in the context of other studies of the same phenomenon, making sure we distinguish correlation and causation, and presenting the story in a way that is not prone to cognitive biases and will not lead to misunderstanding."

An essential and non-trivial component in using data science effectively is in setting the right objectives. Often, the objectives seem clear at the beginning of project development, but later in the process, team members realize that objectives are not as clear as they initially seemed to be. This difficulty in setting objectives regularly arises from data science being applied to really complex and hard problems.The text goes through several examples of data science projects to demonstrate the challenge in establishing clear objectives that are acceptable across parties involved.

Traditionally, software developers strive to eliminate bugs and achieve certainty that the software computes the correct answer. In data science, additional challenges get in the way and errors are likely to perpetually occur. As such, the scenarios must inherently be tolerant of some failures,

to a reasonable degree. If not, perhaps the setting is not amenable to a data science solution. We must remember, it is often difficult to agree upon *the* correct answer to a data science problem.

Interestingly, it is mentioned near the end of this book that the United States is one of many countries that are racing to outdo each other at data science and artificial intelligence. Governments are investing heavily in these efforts. This is reminiscent of the race to the Moon during the Cold War. Competition builds momentum and drives progress.

The key ideas of this book are summarized in this fitting paragraph (adapted from p. 229). "Data science has been successfully applied in many applications and it will be applied to many more. New techniques, greater computational power, and creativity will combine to make currently impossible and impractical applications feasible. Individuals and institutions dependent on data science for their success are likely to become even more so." The societal concerns were minimal a few decades ago, when data science was in its infancy. These concerns have grown substantially and are garnering increasing attention at both the business and the political level.

The concluding section of this book provides pragmatic recommendations. The authors present a clear argument that the prevalence of data science and the diversity of its manifestations beg for increased educational opportunities in data science at all levels. Thus, data science should be woven into existing curricula at primary, secondary, and post-secondary levels, as a mandatory and essential component. Author Alfred Spector ran a seminar at MIT based on this book in Spring 2023. The companion website DataScienceInContext.com includes class materials replete with lecture slides, so that the contents of this book can be brought to other educational settings. Although it is complicated to put in place, more legally mandated regulation is certainly in order. It seems necessary and prudent to update existing laws to encompass the societal changes that have been brought about by the proliferation of data science applications.

3 Opinion

This book is accessible to all who are interested in learning about the developing field of data science. It should open discussions and guide each one of us to new realizations. Understanding what data science is and how it is useful should stimulate those who develop the tools, those who use the tools, and policy makers, to understand how data science can continue to be used more effectively as well as more ethically across many domains. Consumers of data science products and their outputs should feel empowered to know what they can ask for.

In the short interim since this book was published, ChatGPT has emerged. Now generative AI tools have become widely available and are now used in a wide variety of settings, to improve quality of output as well as to boost productivity. Policy makers are grappling with the ethical implications of these new tools. Users are trying to understand how much they can trust these tools and to develop an appropriate level of reliance on them. The topics of this book certainly relate to the range of issues that arise in data-driven AI. Thus, this book has become increasingly more relevant to a broader audience now that generative AI tools have become widely available.

The contents of this book are well organized, providing many tables and charts to organize ideas. This book is well written and technically correct, as evidenced by the abundance of references. Technical terms mentioned in this book are first defined and then thoroughly explained, making no assumptions about the reader's familiarity with these terms. Although no formal background in any discipline is necessary, a rudimentary familiarity with statistics makes the book that much easier to read.

The four authors of this book are committed to promoting data science education. It is this commitment which inspired them to write this book. It is my hope that the book is successful in making the field of data science approachable and understandable to all!