The Book Review Column[1]

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

1 Notable New Releases

The Joy of Cryptography: An Undergraduate Course in Provable Security (MIT Press, 2026), by Mike Rosulek (Oregon State University), is a comprehensive introduction to the fundamentals of provable security for advanced undergraduates.

Introduction to String Algorithms (Princeton University Press, 2026), by Carl Kingsford (Carnegie Mellon University), provides a guide to modern data structures and algorithms for large-scale string processing. The book is suitable for advanced undergraduates, graduate students, researchers, and practitioners in this field.

Basic Graph Theory (Cambridge University Press, 2026), by Béla Bollobás (University of Cambridge) and Robert Morris (IMPA, Rio de Janeiro), is a gentle introduction to graph theory, focusing on beautiful questions, ideas, and proofs, as well as illustrating powerful techniques such as the probabilistic method.

Discrete Mathematics: A Combinatorial Approach (Springer, 2026), by Christos A. Athanasiadis (University of Athens), is an undergraduate textbook that provides a concise introduction to combinatorial and discrete mathematics, with an emphasis on enumeration and generating functions.

2 This Column

Fred Green, who edited this column from 2015 to 2022, returns with his review of *Mathematical Logic: An Introduction* (De Gruyter, 2023), by David Cunningham. He recommends this book to undergraduates seeking a rigorous but rewarding treatment of the subject.

Guide to Using Generative AI in Programming (Springer, 2026), by Antti Laaksonen, is a timely undergraduate textbook on harnessing the power of generative AI tools to aid in learning, teaching, and practicing software development. I think this useful guide is essential for programmers in the post-AI world.

3 How to Contribute

Take a book along on your summer vacation 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

1. Brody, J. (2025). *The Joy of Quantum Computing: A Concise Introduction*. Princeton University Press.

2. Dalzell, A., & McArdle, S., & Berta, M., & Bienias, P., & Chen, C.-F., & Gilyén, A., & Hann, C., & Kastoryano, M., & Khabiboulline, E., & Kubica, A., & Salton, G., & Wang, S., & Brandão, F. (2025). *Quantum Algorithms: A Survey of Applications and End-to-end Complexities*. Cambridge University Press.

3. Erciyes, K. (2025). *Guide to Distributed Algorithms: Design, Analysis and Implementation Using Python*. Springer.

4. Morazán, M. T. (2025). *Programming-based Formal Languages and Automata Theory: Design, Implement, Validate, and Prove*. Springer.

5. Herbert, S. (2026). *Quantum Computing: Foundations and Practice*. Oxford University Press.

6. Kingsford, C. (2026). *Introduction to String Algorithms*. Princeton University Press.

Computability & Logic

1. Badia, G., Crossley, J. N., & Stillwell, J. C. (2026). *What is Mathematical Logic?, 2nd ed.* Oxford University Press.

Miscellaneous Computer Science & Mathematics

1. Wilson, R. (2025). *Sum Stories: Equations and Their Origins*. Oxford University Press.

2. O'Rourke, J. (2025). *The Mathematics of Origami*. Cambridge University Press.

3. Meister, M., Lee, K. H., & Portugues, R. (2025). *Mathematical Biology*. The MIT Press.

Data Science

1. Alpaydın, E. (2025). *Fundamentals of Probability and Statistics for Machine Learning*. The MIT Press.

2. Tromp, J. (2025). *A Geometrical Introduction to Tensor Calculus*. Princeton University Press.

Discrete Mathematics and Computing

1. Devadoss, S., & O'Rourke, J. (2025). *Discrete and Computational Geometry, 2nd ed.* Princeton University Press.
2. Athanasiadis, C. A. (2026). *Discrete Mathematics: A Combinatorial Approach.* Springer.

Cryptography and Security

1. Garfinkel, S. (2025). *Differential Privacy.* The MIT Press.
2. Beyne, T., & Rijmen, V. (2025). *Linear Cryptanalysis.* Cambridge University Press.
3. Rosulek, M. (2026). *The Joy of Cryptography: An Undergraduate Course in Provable Security.* MIT Press.

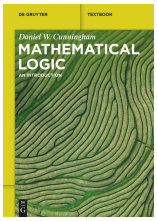
Combinatorics and Graph Theory

1. Mubayi, D., & Verstraete, J. (2026). *Extremal Graph and Hypergraph Theory.* Cambridge University Press.
2. Fijalkow, N. (Ed.) (2026). *Games on Graphs: From Logic and Automata to Algorithms.* Cambridge University Press.
3. Bollobás, B., & Morris, R. (2026). *Basic Graph Theory.* Cambridge University Press.

Review of[1]

Mathematical Logic: An Introduction

David W. Cunningham



Walter de Gruyter GmbH, 2023
256 pages, Softcover, $68 (on Amazon)

Review by
Frederic Green fgreen@clarku.edu
Departments of Mathematics and Computer Science
Clark University, Worcester, MA

One of my colleagues once said to me, "I don't do proof theory. I do mathematics." I guess logic (of the strictly mathematical kind) is one of those subjects that's too easily taken for granted by the working mathematician. Indeed, a firm grounding in mathematical logic does not seem to be regarded as an absolute necessity to the math or CS undergraduate. However, this strikes me as a gap that is worthy of being filled, and the subject provides a perfect landscape for a deep understanding of how proofs work in practice, with all due respect to my colleague. In this way, a textbook on mathematical logic directed towards the undergraduate, such as this one, is a welcome addition.

Full disclosure: Although my research for the past 40 years or so has amounted to proving theorems, I never took a course in logic[2] *per se*. But most of what attracted me to computational complexity in the first place was its foundational issues stemming directly from mathematical logic. My teaching of "logic" has only amounted to those fragments of it introduced in discrete math, and those refreshers that one typically gives in algorithms and computability/complexity courses. So seminal results such as Gödel's theorem, for me, are principally viewed from the computational standpoint, and reading through this book was a very enlightening experience.

The present volume starts at the very beginning and over the course of six chapters engages in a rigorous climb, passing many fine views, and at last achieving the summit of Gödel's incompleteness theorems. The contents of the chapters are roughly as follows:

1. *Chapter 1, Basic set theory and basic logic:* This establishes standard basic notations and concepts, such as functions and relations, basic logic and logical connectives, and also including elementary set theory, countability and cardinality. One distinguishing feature is an explicit theorem that very carefully shows how functions and sets can be defined by recursion. This tool is used throughout the text, for example to formalize languages and (e.g., in Chapter 3) to extend functions on terms to functions on formulas and prove they are unique.

2. *Chapter 2, Propositional logic:* Here the basic logic of Chapter 1 is expanded to more precisely encompass the relevant language of propositional logic, the technical meaning of truth assignments, tautologies, satisfiability, etc. Theorems on tautological completeness and compactness are then proved, and the idea of deducibility introduced.

[1] ©2026, Frederic Green
[2] Having been originally trained as a physicist, this is hardly surprising.

3. *Chapter 3, First-order logic:* Those first two chapters were the foothills, and here the ascent gets steeper as the subject grows in significance. Here we grapple with the meanings of "truth" and "proof." After a substantial section on the language of first-order logic and its structure, we delve into the nature of truth, encapsulated in Tarski's definition of satisfaction. We next encounter structures, definability in a structure, the beginnings of model theory, and the homomorphism theorem of logic. From here we go to the notion of proof, founded on the idea of formal deductions from logical axioms via rules of inference.

4. *Chapter 4, Soundness and completeness:* If something is provable, is it true? Moreover, if something is true, is it provable? This chapter revolves around two key properties of first-order logic: first, the soundness theorem (if a set of wffs is true in a structure, then so is any wff that is deducible from that set); and second, Gödel's *completeness* theorem (given a set of axioms that are true in all models, any statement that is true in those models is deducible (i.e., provable) from those axioms). One consequence of completeness is the Compactness Theorem, which states that if every finite subset of a set Γ of wffs is satisfiable, then Γ is satisfiable. The chapter concludes with applications of the Completeness and Compactness. Consequences of the latter include nonstandard models and Robinson's idea of nonstandard analysis. The basic results underlying nonstandard models of arithmetic are presented. There are also sections presenting the Löwenheim-Skolem theorems, theorems regarding the completeness and axiomatizability of logical theories, and finally proving the correctness of prenex normal form. (At least up to this point, this was my favorite chapter.)

5. *Chapter 5, Computability:* Here we start with a fairly standard informal, intuitive notion of algorithm, which is used to illustrate the notion of decidable sets, total, partial, and computable functions, and the undecidability of the halting problem. There follows a section on formalizations, focusing especially on the Turing machine, and partial recursive functions, as built out of primitive recursion and "partial search" (the unbounded μ-operator), all defined here. It ends with a statement of the Church-Turing Thesis. The next section is a detailed investigation of primitive recursive functions, with a long list of results of such functions and relations. Building on such constructs as bounded quantification and bounded recursion, and thus bounded search and the bounded μ-operator, one can obtain results such as the primitive recursiveness of primality and finding the n^{th} prime number, as well as coding and decoding a number as a sequence of numbers, using the unique factorization into primes. All these in turn lay the groundwork for the primitive recursiveness of Gödel numbers, of central importance in the coming chapter. (This chapter was enjoyable and certainly closest to my own comfort zone.)

6. *Chapter 6, Undecidability and incompleteness:* Much of the preceding five chapters set the stage for this one. It begins by posing the question of whether we can determine if a set of axioms is decidable. The second section introduces a (finite) set of axioms for basic number theory, which incorporates successor, (in)equality, addition, multiplication, and exponentiation, which it calls the Ω-axioms, with the theory thereof denoted $\mathcal{L}$. A number of positive results are then proved to indicate the strengths of the theory, e.g., that Ω can deduce terms involving addition, multiplication, and exponentiation, as well as any true quantifier-free formula and existential sentences. The results are then generalized, showing how $\mathcal{L}$ can represent functions and relations. We can say a formula represents a function if (given the "input" to the function and its "output") it is true iff the function equals its output, i.e., expressed the

graph of the function. Thus we find that we can, in this sense, deduce (from Ω) a formula in $\mathcal{L}$ representing the graph of a function. A series of results builds up to the fact that any recursive function is representable in this language. We then move to the arithmetization of $\mathcal{L}$. Leveraging the results of the previous chapter, a series of results leads to the primitive recursiveness of a number of functions and relations, e.g., the Gödel numbers, determining if (the Gödel number of) one formula is a generalization of another, the set of tautologies, and indeed every representable function and relation. Finally, we proceed to the incompleteness theorems of Gödel, largely on with the fixed-point lemma of logic. It uses this to prove Tarski's Undefinability Theorem, which essentially says that the set of true first-order formulas is not definable within arithmetic. Although Tarski's theorem post-dated the incompleteness theorems, Gödel came upon a form of it while proving them, and it can be used to that purpose, as is done here. From this, Gödel's first incompleteness is quickly stated and proved, and similarly for the second one. (And *this* one was my favorite chapter of all!)

The early sections already display great concern for the reader; some proofs are quite detailed. The same is true of definitions. One instance of this concerns a definition (3.1.6) of terms, as extended from variables and constants, by a set of functions. As it can be a little abstract, the text follows through, in making the definition more concrete by example, in terms of one of the earlier theorems on recursion. Examples (and *non*-examples, something too often missing) are frequently serve to motivate definitions. Many problems are stated and solved in detail in the text itself, which is great preparation for the student working the exercises without the benefit of solutions. Of course, as the book progresses, it leads the reader into deeper territory. As a consequence, more is expected of the reader to fill in the details (indeed some exercises are used in proving later results). Of course this encourages more and more active reading, a rewarding experience which counts as solid pedagogy. Nevertheless, the most important theorems (to cite just one example, the Completeness Theorem of Chapter 4, using Henkin's method of adding witnesses) are carefully and cogently presented.

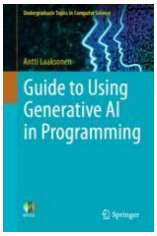
In conclusion, this book is eminently suited to a junior- or senior-level course, and I learned a great deal from it. It is also worth mentioning another book of the author's on set theory, reviewed by me on these pages[3], which makes an excellent companion volume to this one.

[3] *Set Theory: A First Course*, by Daniel W. Cunningham, *ACM SIGACT News* **48**(3), (2017), pp. 7-9, review by F. Green

Review of[1]

Guide to Using Generative AI in Programming

Antti Laaksonen



Springer, 2026
183 pages, $49.99 Softcover, $39.99 eBook

Review by
Nicholas Tran
Department of Mathematics and Computer Science
Santa Clara University

Antti Laaksonen's *Guide to Using Generative AI in Programming* is a concise primer on using generative AI tools in programming. Written for undergraduate programmers who seek to improve their coding efficiency, the book explains the capabilities and limitations of generative AI tools for programming and demonstrates how they can be used to generate, analyze, test, and debug code in twelve short chapters that present techniques through examples. The book also illustrates how to learn programming actively using generative AI as both a tutor and an assessment tool. Acknowledging the rapidly evolving power of generative AI, the book ends with an ambivalent discussion of the future of programming. When the author asked a chatbot to respond to the statement that, although AI "may have some programming skills, but human programmers are still superior," the chatbot replies that "AI can enhance and support, but not replace, the unique strengths of human programmers." Whether this is an example of AI sycophancy is left as an exercise for the reader.

1 Summary of Contents

Chapter 1 gives an overview of the contents and organization of the book and provides some evidence of the book's human authorship. **Chapter 2** reviews the evolution of tools and resources available to programmers up to the arrival of generative AI. **Chapter 3** introduces general-purpose chatbots and editor-integrated assistants based on large language models that can generate code. It outlines best practices for using these tools, including how to write effective prompts and the need to verify the accuracy of the generated code. It ends with a brief discussion of the key concepts in the technology of generative AI such as language models, tokens and embeddings, neural networks, and context and attention.

Chapters 4 through 7 describe how to use generative AI tools in four main programming activities. **Chapter 4** explains how generative AI can be better than traditional search engines in retrieving documentation (especially from different sources) when provided additional context and examples. **Chapter 5** illustrates prompting techniques to generate code that matches more precisely the programmer's needs. A particularly useful tip is to ask the AI to generate code based on examples or even drawings and to automate routine tasks with shell scripts. **Chapter 6** discusses

the use of AI tools to inspect code for logical errors, to generate unit tests, and to simulate user interactions to test user interfaces. One interesting example is the use of AI to generate a large number of test cases to compare the output of an algorithm against that of a brute-force solution for the same problem. **Chapter 7** demonstrates how AI tools can effectively explain the purpose of a given piece of code; in contrast, these tools are no better than traditional tools in reformatting code or improving its style. Converting code to another language or refactoring code to improve its structure are also tasks that AI tools can perform better than traditional tools, although the result should be verified for correctness.

Chapter 8 reinforces the caution made in the earlier chapters about accuracy and reliability of AI-generated code by discussing the limitations of generative AI tools, especially the tendency to "hallucinate" or generate made-up information to please the user.

Chapter 9 shows that well-known software engineering principles such as precise specification, and top-down modular design are still important when using generative AI tools to develop a complex project. As an example, the chapter walks through the development of a Tetris-like word game as a web application using HTML and JavaScript. The example shows that imprecise specifications of the intuitive concept of "rotation" led to buggy versions of the game, but once the correct definition and a good modular design was provided, the AI tools were able to generate code that worked correctly on the first try.

Chapter 10 shows that generative AI tools can be used effectively as a personal tutor to explain well-known concepts, to help debug code, and to create exercises to test the user's understanding, but it also warns about the risk of passive learning, where the user simply verifies the AI's solution without engaging in the active problem-solving process. As an example, the chapter demonstrates how to learn to program in the language Prolog.

Chapter 11 discusses the use of generative AI tools for teaching from the instructor's perspective. It gives various methods for recognizing AI-generated solutions and shows how to use AI as a source of inspiration for creating exercises.

Chapter 12 ends with a discussion of the future of programming in the age of generative AI.

2 My Opinion

Undergraduate programmers at all levels will learn something useful from this book (especially Chapter 10), and its no-nonsense style makes it easy to absorb the material. Their instructors will find Chapter 11 useful as well; although the book does not discuss GitHub Copilot at length, I personally find it convenient (in a disturbing way) in preparing exams and their solutions.

The reader should take heed of the book's warning to take advice from generative AI with a healthy dose of skepticism. Inspired by the book, I asked ChatGPT to explain the JFLAP code of a DFA accepting binary strings starting with 00 and ending with 1 and of a Turing machine accepting $\{0^n1^n : n \geq 0\}$. Its answers are generally correct but not perfect: several iterations were needed to get the simplest description of the DFA, and a warning of a nonexistent bug was issued for the Turing machine. ChatGPT and Claude were much less successful with my request to generate a two-tape Turing machine to perform binary addition, giving solutions that stopped immediately after the first move on simple inputs.

Currently free plans for high-quality generative AI tools impose a daily access limit that is too low for many applications described in the book.