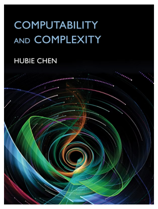
Review of [1]

Computability and Complexity

Hubie Chen

The MIT Press, 2023
416 pages, $65 Hardcover



Review by
David Luginbuhl (`dluginbuhl@fit.edu`)
Electrical Engineering and Computer Science
Florida Institute of Technology

1 Overview

Theory of computation and computational complexity have been well-studied and written about extensively over the past several decades, much of it chronicled in this publication. We are all familiar with the many textbooks that have captured the themes in a way that is accessible to college students (and we all have our favorites, I am sure). While many of these texts cover topics in both computability and complexity, they tend to emphasize the former to establish the fundamentals (and my experience has been that focusing on computation usually takes up most, if not all, of a semester-long formal languages course). Hubie Chen, in his new book *Computability and Complexity*, strives to achieve more of a balance between these two main topics. Through his highly mathematical treatment and in-depth proofs, he also builds a strong connection among the major themes of the text.

2 Summary of Contents

In the *Preface*, Chen describes the audience for this book, stating that it is written for a course "at the upper undergraduate level." He also maintains that it would be useful for "students, researchers and workers in disciplines that draw on or depend on this theory" or as a reference for a theory-related course.

In the *Introduction*, Chen says that he is "motivated by two questions:

- What is *computable?*
- What is *efficiently computable?*"

Chapters 1 and 2 answer the first question, with a focus on finite state machines (Chapter 1) and Turing computability (Chapter 2). Chapters 3 and 4 comprise a detailed response to the second question, exploring issues of time-bounded computation (Chapter 3) and several additional complexity-related topics (Chapter 4).

[1]©2025 David Luginbuhl

The *Agreements* section in the front matter provides a very short introduction (three pages) to the mathematical foundations necessary to understand computation, particularly, formal language theory, set theory, and relations and functions. The treatment is terse, but this knowledge is assumed coming in, and Chen is only ensuring we are all on the same page in terms of notation. I would add that readers will want to have a high comfort level with mathematical proofs, because they are the bread and butter of this book, especially in the latter part.

Each of the four chapters ends with an extensive *Exercises and notes* section, which explores some concepts from the chapter in more detail, introduces interesting related topics, and provides a large number of exercises.

3 Chapter Highlights

Chapter 1 ***Automata Theory*** – This chapter might be more appropriately titled *Automata Theory and Regular Languages.* DFAs, NFAs, and regular expressions are all introduced, but unlike many texts in this area, Chen does not address any language classes between those represented by DFAs and those represented by Turing Machines. This is just an observation and is not meant as a critique; Chen does not need the intermediate languages or automata to support his overall discussion of computation.

One interesting aspect of Chen's approach to exploring finite automata is that he eschews extending the δ function (δ^*) in favor of *configurations* of the machine (indicating the current state and the rest of the string yet to be consumed). Chen uses this construct not only to walk through a computation, but also to define acceptance, the language of a machine, and as a result, whether or not a language is regular. Most theory texts introduce students to both the extended δ function and configurations, but Chen emphasizes the configuration approach.

The rest of the chapter is a fairly straightforward treatment of the various representations of regular languages - NFAs, ϵ-NFAs, and regular expressions. Closure properties are covered.

Chen introduces *pairwise separability* of strings as a way to discuss non-regularity of strings and minimization. The Pumping Lemma is found in the *Exercises and notes* section.

Chapter 2 ***Computability Theory*** – In Chapter 2, Chen turns to the concept of computation and to the Turing machine. As he says, "the *halting deterministic Turing machine* will be presented as a formalization of the intuitive notion of algorithms." So a definition of algorithms is in order, and I like Chen's: "a finite list of instructions; each instruction is finite and unambiguously describes an action performable mechanically, without recourse to judgment or creativity. An algorithm operates deterministically, and in discrete time steps. When executed on an input, an algorithm terminates after a finite number of steps, producing the desired output" (p. 71-72). Plenty of definitions of algorithms exist, but this one provides a nice connection to Turing machines.

For classes of languages, Chen prefers the terms *computable* and *computably enumerable* to *recursive* and *recursively enumerable.*

Chen introduces encodings to present concepts like Universal Turing machines and to prepare for the distinctions between computably enumerable, computable, and non-computably enumerable.

The chapter contains standard Turing machine topics such as closure properties and alternatives to deterministic Turing machines, in particular, nondeterministic Turing machines, multi-tape machines, and Random Access Machines. Reductions from one language to another are discussed near the end of the chapter to set up for further non-computability results.

Chapter 3 ***Complexity Theory*** – Having taken us through the basics of computation, Chen moves to complexity results, which I would say form the heart of this book. The first two chapters really set the stage for what follows here, and although the book (as indicated by the title) is about both computation and complexity, the coverage of complexity in this chapter is especially comprehensive and detailed. Because of this, I will provide a bit more detail on this chapter.

The first section provides some background in graph theory and focuses in particular on Eulerian cycles and graph coloring as basic algorithms for investigating complexity. Graphs are explained in detail, and associated problems are formalized as languages using the string encoding introduced in the previous chapter. The point here is to demonstrate their computability and also to use them as a basis for discussing time-efficient computation.

Chen then introduces time-bounded computation for both deterministic and nondeterministic machines and uses this to define PTIME and NP. Eulerian cycle detection and 2-colorings are provided as examples of PTIME languages - detailed proofs are provided. A non-graph example, determining whether an integer is a proper divisor of another integer, is also noted as a PTIME language.

Next up are NP languages, with Chen presenting some fundamentals and also establishing the basic relationship between NP and PTIME. Of course, he addresses "The P vs NP question." He gives some attention to the concepts of *solving* and *verifying* as a way to address this question, characterizing it as follows: "Throughout human experience, the act of *creating* has been felt to be distinctly more challenging than the act of *evaluating* a creation; whether or not this perceived discrepancy is genuine, in the milieu of polynomial-time computation, can be seen as the heart of the P versus NP question!" (p. 159)

Several languages are examined that are in NP, such as HAMILTONIAN CYCLE and 3-COLORING. Chen provides an alternative formalism for considering whether a language is in NP by defining the concept of a polynomial time deterministic verifier and proving in great detail that a language is in NP if there is a verifier for the language.

Chen describes closure properties of PTIME languages, with a focus on complementation in order to introduce coNP. He then describes and provides fundamental relationships between NP and coNP.

In order to discuss NP-completeness, we must have an understanding of reductions. To exemplify the concept of reducibility, Chen again makes use of graph-theoretic concepts, this time with the languages of INDEPENDENT SET and CLIQUE. In this case, he shows that INDEPENDENT SET reduces in polynomial time to CLIQUE and (for good measure) to VERTEX COVER. Chen also discusses basic properties of reducibility.

Chen introduces NP-completeness and NP-hardness using the notion of polynomial time reducibility, which sets us up for the first NP-completeness result. While some texts begin with BOOLEAN SATISFIABILITY (essentially, Cook's theorem), Chen chooses instead CIRCUIT SATISFIABILITY, showing that any NP language can be reduced to CIRCUIT SATISFIABILITY. He also examines coNP-hardness and coNP-completeness using languages related to CIRCUIT SATISFIABILITY.

Chen presents several additional NP-completeness proofs, moving from CIRCUIT SATISFIABILITY to BOOLEAN SATISFIABILITY and a number of variants. From there, Chen uses reductions to show NP-completeness for various graph problems, including INDEPENDENT SET, VERTEX COVER, DOMINATING SET, CLIQUE, HAMILTONIAN PATH and HAMILTONIAN CYCLE and related problems, as well as TRAVELING SALESPERSON.

This chapter can serve as a reference for NP-completeness results. The proofs are set up in great detail and are quite thorough. Early on, Chen even provides a strategy to approaching these proofs for those not wishing to examine them all in detail.

Chapter 4 ***Further Complexity Theory*** – The final chapter introduces a number of additional topics in complexity. It begins with space complexity, providing some definitions and fundamental relationships between time and space. Chen defines PSPACE and discusses relationships between PTIME and PSPACE. He introduces quantified propositional formulas to prove QUANTIFIED SATISFIABILITY as a PSPACE-complete language. He then provides a detailed discussion and proof of Savitch's theorem, showing that space bounds in NTMs and DTMs are polynomially related.

The next section of this chapter covers space and time hierarchy theorems. As Chen notes, these theorems show that "for each of the resources of space and time, *one can do more, with more of the resource*" (p. 291, emphasis in the original).

Chen next moves to fixed-parameter tractability and parameterized complexity, which were new concepts to me. They appear to provide a way to examine some types of problems (certain graph-theoretic and database query problems, for example) with more granularity than is allowed by the standard time complexity framework. Chen summarizes this as follows: "The notion of fixed-parameter tractability, in essence, allows one to identify problems where a seemingly unavoidable combinatorial explosion can be confined to a parameter, and thus to finely pinpoint sources of combinatorial hardness" (pp. 298-299). It is a fascinating topic that Chen addresses in sufficient detail and with motivating discussion for those interested in exploring further.

The last major topic in this chapter is compilability theory, which is related to the preceding parameterization discussion. As Chen states in introducing this concept, "When trying to solve instances of a problem, if it is the case that multiple instances of relevance share a feature in common, it may be fruitful to *compile* this feature into a format that allows for more efficient solution, even if the compilation process is relatively expensive" (p. 332). This section sets out to formalize that compilation process and show how it can be used.

The *Exercises and notes* section of this chapter introduces two additional topics worthy of consideration: logarithmic space complexity and treewidth. The latter is a graph-theoretic concept that Chen uses to provide more results on parameterized complexity.

I would treat this last chapter as more of a "for further reading" chapter. For those who wish to dig deeper into computational complexity, there is more than enough here to whet the appetite.

4 Summary

What stands out to me in Chen's book above all else is the rigor and detail in his proofs throughout the text. And there are a **lot** of proofs. But Chen is very good about including motivation and high-level introductions to proofs where appropriate. Often, he includes intermediate results to move the discussion forward and keep the attention of the reader. It is possible to examine many of the proofs at multiple levels of detail. With respect to the proofs, I believe he is true to his intent (noted in the Preface) to provide a "uniform treatment of core concepts and topics." There are many common threads here, and Chen does a good job of ensuring the reader is able to see how earlier results lead us to more advanced theorems.

Referring back to Chen's discussion of intended audiences, I would say that the first two chapters are undoubtedly written at the upper undergraduate level. Interested undergraduate students with the requisite mathematical maturity will be challenged by, but benefit from, the complexity discussion in the latter half of the text. As a whole, the book is appropriate for a graduate course in theory of computation and could be considered for a similar undergraduate course. I would certainly recommend it as a candidate text for a graduate research seminar in this area.

I agree that the extensive proofs provided, especially those in chapters 2 and 3, make this book useful as reference for those interested in these topics. It is deserving of a place on the bookshelf of anyone planning to do research in theory of computation.