Review of[1]
Theory of Computation
by Dexter C. Kozen
Springer, 2006
418 pages, Hardcover, $71.86 (Amazon)

Review by
Daniel Apon
dapon@cs.umd.edu

1 Introduction

Theory of Computation is designed to serve two purposes: (i) provide a student's first, rigorous survey of the foundations of computing, and (ii) give a taste of an assortment of advanced topics, to provide an avenue for further study. Its most immediate, striking aspect is a unique organization. Rather than using a chapter format, the book is divided into a series of "lectures." Each lecture is between 4 and 7 pages long and is designed to be a self-contained, readable unit on some topic. I found this approach extremely appealing.

The content of the book primarily focuses on computational complexity theory, though it briefly covers other relevant topics – for instance, there are a few lectures on algorithms for factoring and another on the basic bounds and inequalities used in probabilistic algorithm analysis. It's useful to know, from the outset, that a large portion of the complexity theory in *Theory of Computation* is disjoint from the material in other, related texts like Arora and Barak's *Computational Complexity: A Modern Approach*. The choice of which material to include is more complicated to judge without discussing the details, so — read on!

2 Summary

The material is based on Kozen's course, CS682: Theory of Computation, a semester course for first-year graduate students at Cornell University. In the first 270 pages, there are 41 primary lectures and another 10 supplementary or optional lectures interspersed throughout the text. In the sequel, there are 100 pages of homework exercises and (instructors be aware!) detailed solutions for *all* of the exercises.

I will begin by briefly highlighting the unique contributions of *Theory of Computation* with respect to the universe of complexity textbook material. Following that, there will be a general survey of the text, covering the essence of its total content.

2.1 What's unique in this book?

Let's begin with a few lecture titles: *Complexity of Decidable Theories. Applications of the Recursion Theorem. Complete Problems in the Arithmetic Hierarchy. The Friedberg-Muchnik Theorem. The Analytic Hierarchy. Fair Termination and Harel's Theorem.*

In many ways, the content of the book seems to be a reflection of a broader trend in complexity theory during the years of its creation: a general transition from logic-based complexity theory to more combinatorics-based complexity theory. As a result, both "worlds" are introduced in this book. In my opinion, it does an excellent job of discussing both – but more on that later.

2.2 Survey of Lectures

Note that the following grouping of lectures into sections is my own impression of the structure of the book based on what appear to be natural transitions in its focus or direction. The text itself is in fact 51 lectures in sequence, counting supplementary lectures.

2.2.1 Lectures 1-6: An Introduction

The 1st lecture by introducing the Turing Machine model. A proof of $\Omega(n^2)$ time for palindrome recognition on a one-tape TM follows. The 2nd lecture introduces all of the basic deterministic and nondeterministic space and time classes and their simple inclusions, as well as Savitch's theorem ($\mathsf{PSPACE} = \mathsf{NPSPACE}$). The 3rd lecture demonstrates the essence of known space hierarchies via padding techniques. The 4th lecture covers Immerman-Szelepcsényi's theorem (NSPACE is closed under complement for space $\geq \log n$). The 5th lecture introduces logspace computation and reducibility, and shows directed graph reachability complete for $\mathsf{NLOGSPACE}$. The 6th lecture proves the Cook-Levin theorem using *logspace* reductions (an interesting approach!).

2.2.2 Lectures 7-10: Alternation, PSPACE, and PH

Lecture 7 begins with a definition of an alternating Turing Machine and proves relationships between alternating and deterministic complexity classes $\left(\text{e.g. } \mathsf{ASPACE}(S(n)) \subseteq \mathsf{DTIME}(2^{O(S(n))}\right)$. In the 8th lecture, complete problems for PSPACE are introduced, including satisfying quantified Boolean formulae and finding a forced win in chess. Following this, the 9th and 10th lectures are on the polynomial hierarchy, introducing oracles, building PH, and then relating levels of PH to oracle-based classes, e.g. $\mathsf{NP}^{\mathsf{NP}}$.

2.2.3 Lectures 11-12: Parallel Complexity and NC

Lectures 11 and 12 form a brief introduction to NC. Uniform families of circuits are defined, a family of logspace-uniform NC circuits to compute Boolean $n \times n$ matrix multiplication is given, and $\mathsf{NLOGSPACE} \subseteq \mathsf{Uniform\text{-}NC} \subseteq \mathsf{P}$ is proven.

2.2.4 Lectures 13-14: Probabilistic Complexity and BPP

The 13th lecture begins by briefly reviewing notions of probabilities of events, expectation, conditional probability, pairwise independence, and so on. Then, probabilistic Turing Machines are defined and in turn used to define the classes RP and BPP. An example of a useful, efficient probabilistic algorithm is given: namely, testing whether a low-degree multivariate polynomial with integer coefficients is identically 0 (where the straightforward deterministic algorithm requires exponential time). Then, in the 14th lecture, amplification is introduced (in the context of reducing probability of error exponentially with repetition) and used for a proof of $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{p}} \cap \Pi_2^{\mathsf{p}}$.

2.2.5 Lectures 15-20: IP and PCP

In lecture 15, IP is defined. The 16th and 17th lectures cover the proof of IP = PSPACE. In lecture 18, $\mathsf{PCP}(r(n), q(n))$ is defined. Then lectures 19 and 20 discuss a proof of $\mathsf{NP} \subseteq \mathsf{PCP}(n^3, 1)$. Two comments: (i) $\mathsf{NP} \subseteq \mathsf{PCP}(\log n, 1)$ is known; the book gives an intentionally simplified exposition – this is good – and (ii) the PCP proof in the text uses the original argument involving arithmetization, linearity testing, and so on, as opposed to Dinur's later combinatorial proof with expander graphs – this is unfortunate. (I feel that Dinur's proof is, if nothing else, easier to follow. However, it was published the same year that *Theory of Computation* was published.)

2.2.6 Lectures 21-27: Complexity of Decidable Theories and ω-Automata

The next series of lectures give a treatment of the complexity of first-order and second-order theories. The 21st lecture gives a proof showing that the first-order theory of dense linear order without endpoints is PSPACE-complete. The first-order theory of reals with addition and multiplication is shown to be decidable in lecture 22, and is shown to be NEXPTIME-hard in lecture 23. The 24th lecture shows the theory of integer addition to be complete for the class accepted by alternating Turing Machines with at most n alternations running in time $O\left(2^{2^{n^{O(1)}}}\right)$. The monadic second-order theory of successor is shown to be decidable, but not in elementary time (i.e. time bounded by a stack of exponentials), using Büchi, Rabin, and Muller automata over lectures 25-27.

2.2.7 Lectures 28-29: Relativization and Sparse Sets

The next lectures cover a couple of theorems directly relevant to the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ question. Lecture 28 introduces Baker, Gill, and Solovay's proof that there are oracles A and B such that $\mathsf{P}^A = \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$ as well as discusses the rise and fall of the Random Oracle Hypothesis – that containments or separations that hold with probability 1 with respect to a random oracle hold in the unrelativized case (which is particularly apt in light of the proof of IP = PSPACE earlier!). Lecture 29 introduces Fortune's and Mahaney's theorems – that sparse languages cannot be coNP-complete or NP-complete, respectively, unless P = NP.

2.2.8 Lectures 30-31: Circuit Lower Bounds

The aim of these next lectures, combined with a subsequent supplementary lecture, is to prove two complementary results. Lecture 30 sets up the beginning of a proof that that there exists an oracle A such that $\mathsf{PH}^A \neq \mathsf{PSPACE}^A$. Lecture 31 proves $\mathsf{PARITY} \notin \mathsf{AC}^0$. Supplementary Lecture H combines this result with Håstad's Switching Lemma to complete the proof from Lecture 30.

2.2.9 Lectures 32-34: The Gap Theorem and the Recursion Theorem

In the 32nd lecture, the Gap Theorem is stated (that there exist arbitrarily large recursive gaps in the complexity hierarchy). This leads into Blum's Speedup theorem – that there exist pathological, computable functions with no asymptotically optimal algorithm. In lecture 33, Gödel numbering is defined in order to prove the Recursion Theorem, which is in turn directly used to prove Rice's theorem in lecture 34 – that every nontrivial property of recursively enumerable languages is undecidable.

2.2.10 Lectures 35-41: The Arithmetic and Analytic Hierarchies

The 35th lecture introduces Turing reducibility in order to define the arithmetic hierarchy, and in lecture 36, various complete problems are demonstrated, e.g. $\{M : L(M) \text{ is recursive}\}$ is shown to be complete for Σ_3^0. Lecture 37 introduces Post's problem ("show that there are more than two Turing degrees"), and the 38th lecture resolves the problem using a finite injury priority argument. This leads toward a definition of the analytical hierarchy in lecture 39, and a proof of Kleene's theorem in lecture 40. Finally, lecture 41 gives an real-world application for the analytical hierarchy in fair termination of concurrent programs, showing that the decision problem is Π_1^1-complete.

3 Opinion

I really enjoyed reading *Theory of Computation*, and I think that had a lot to do with its structure. A huge advantage of having the material divided up into lectures instead of chapters is that you end up with bite-sized morsels of reading. It takes a nontrivial commitment of your time to fully read through a 20-25 page chapter of a textbook. On the other hand, you can go right through one of Kozen's 4-7 pages lectures in just 15-20 minutes, then put the book down and feel as if you've accomplished learning something. (It's hard to overstate how *awesome* that feels.)

As a result, I think the book definitely lends itself quite easily toward use in the classroom and even for self-learning. I imagine one could easily structure as a course around reading a "lecture" in the textbook prior to each in-class lecture over the same material. Since each chunk of reading is so short, you won't burn out your students with the constant reading.

One word of caution for prospective instructors: Since the anwers for every exercise are in the text proper, you won't be able to give graded assignments directly out of the book. But is that necessarily a terrible situation? On the contrary, get creative!

On the topic of which material the book covers, I feel that its approach has some unique strengths and weaknesses. You can ask a question like, *should* a standard, graduate-level complexity course spend time covering, say, complete problems in the arithmetic hierarchy? There are always tradeoffs involved. Spend time on the arithmetic hierarchy, and you sacrifice time that could have been spent on derandomization, inapproximability, communication complexity, Barrington's theorem, or whatever else your favorite, hot topic might be.

On the other hand, Kozen's book does an excellent job of discussing many topics that are *not* in other texts. This material can be used to supplement any existing course in complexity theory at both the undergraduate and graduate levels.

Final verdict? Awesome lecture-style format. Very well-written explanations throughout the book. Definitely a useful, fun book.