Review of[1]

Turing Computability: Theory and Applications
by Robert Soare
Publisher: Springer
$54.00 hardcover , 253 pages, Year: 2016

Reviewer: William Gasarch gasarch@cs.umd.edu

1 Introduction

We discuss the basic objects of computability theory.

A function f is *computable* if there exists a Turing machine (a Java program also works) that will, on input x, output $f(x)$. A set is *computably enumerable (c.e.)* if there exists a Turing machine that will (a) on input $x \in A$ halt, and (b) on input $x \notin A$, not halt. If given B you can compute A (formally with an oracle Turing machine) then we write $A \leq_T B$. If $A \leq_T B$ and $B \leq_T A$ then we write $A \equiv_T B$. This is an equivalence relation. The equivalence classes are called *Turing degrees*.

Computability theory has its origins in logic; however, it has become a field unto itself. This book discusses some of the historical ties to logic as well as present the field on its own terms.

2 Is this a Second Edition of Soare's Prior Book: NO

Lets get one thing out of the way first: This book is emphatically *not* a second edition of Soare's book *Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets*. The new book does not have $0'''$-priority arguments, nor $0''$-priority arguments. It does have $0'$-priority arguments (also known as finite injury priority arguments). The old book was very focused on r.e. (now called c.e.) sets and degrees. The new book takes a much wider view of computability theory.

3 Summary of Contents

The book is in five parts each of which has chapters.

Part I: Foundations of Computability

This part has seven chapters. The first chapter (*Defining Computability*) is a combination of history and math. It describes how the modern notions of computability came about. The second chapter (*Computably Enumerable Sets*) covers the basics of c.e. sets. There is one unusual section on Lachlan Games which seem to be a method that encompasses many constructions of c.e. sets. Later in the book he shows how to phrase finite injury priority arguments in terms of Lachlan games. I am curious if one can phrase $0''$ or $0'''$ arguments this way.

The third and fourth chapters (*Turing Reducibility* and *The Arithmetic hierarchy*) contain standard material on these topics as well as an assortment of other topics: (1) n-c.e. sets, (2) the low and high hierarchy, and (3) the low basis theorem (there is later an entire Part on Trees).

The fifth, sixth, and seventh chapters (*Classifying C.E. sets*, *Oracle Constructions and Forcing*, and *The Finite Injury Method*) cover material that is more advanced— so much so that I wonder

why it is a Part called *Foundations of Computability*. This is not an objection, it is an observation and may also reflect a difference in mine and the authors tastes. Some topics are (1) simple, hypersimple, and hyperhypersimple sets, (2) forcing arguments to construct a set of Turing degree strictly inbetween decidable and Halting, and (3) a finite injury priority arguments to construct a c.e. set of Turing degree strictly inbetween decidable and Halting.

The entire Part is well written with a sense that the author has a broad prospective on the material and has thought deeply about it. This comes through the most when he talks of the history of the field which is more the subject of Part V.

Part II: Trees and Π^0_1 Classes

A *Tree* is a subset of $\{0,1\{^*$ that is closed under prefix. This really is a tree in the way you usually think of it. A set of infinite strings is Π^0_1 if it is the branches of an infinite tree.

Why are Trees important in logic? Consider the following tree T:

1. Every number codes a statement using (say) the usual logic symbols and also $[+, \times, <, 0, 1]$. The quantifiers are over the natural numbers.

2. A string $\sigma = b_0 \cdots b_n$ codes the statement (1) all i such that $b_i = 1$ are true, and (2) all i such that $b_i = 0$ are false.

3. Fix an axiom system (e.g., Peano Arithmetic). A string σ is in T if none of the assertions of σ (as defined by point 2) have been proven false by an $|\sigma|$-step derivation.

Studying models of Peano Arithmetic is equivalent to studying branches of T. This tree is decidable. Hence studying decidable trees will yield theorems about models of Peano Arithmetic. For example, the low basis theorem states that every c.e. tree has a low branch. Hence there is a low model of Peano Arithmetic.

This chapter first studies trees and then applies them to PA. This is very nice since its good for any field of math to, once they prove something, go back to the original motivation. They prove far more about trees then they need for PA, but what they prove may be useful at some later time.

This chapter then discusses how trees, randomness, and measure interact. This is an immense topic that the author can only touch on briefly. We give one example without defining terms formally: if a Π^0_1 class has measure 0 then it has no Martin-Lof Random sets.

Part III: Minimal Degrees

This Part has two chapters. A degree is *minimal* if for any A in that degree (1) A is not decidable, and (2) all $B <_T A$ are decidable. In this chapter they show that there is a minimal degree that is $\leq_T 0''$ and then that there is a minimal degree $\leq_T 0'$. The presentation is very educational in that the author says, at every step of the way, why they are doing what they are doing. Frankly that is true for most of the book, but here it stands out more since they spend two chapters doing what other texts do in 5 pages.

Part IV: Games in Computability Theory

My wife has commented that math games are not fun games. This part will not change her mind.

Here is an example: Let A be a subset of $\{0,1\}^\omega$. The players are Alice and Bob. Alice goes first. Alice picks a bit a_0. Bob picks a bit b_0. Alice picks a bit a_1. Bob picks a bit b_1. etc. If the

string $a_0b_0a_1b_1\cdots$ is in A then Alice wins. If not then Bob wins. The question is: for which sets A does Alice have a winning strategy. Martin proved that all Borel sets have winning strategies. AD is the Axiom of determinacy which declares that all sets have a winning strategy. It turns out that AD is an interesting alternative to AC. However, that is not the direction that the book talks about. The book instead talks about how computable a strategy has to be in certain cases.

This chapter also talks about Lachlan games. In particular, finite injury priority arguments are rewritten as Lachlan games.

Part V: History of Computability

This is a history of computability from Hilbert (1900) until Posts problem was solved (1950's). It is masterful with enough details to be interesting but not so much as to be boring. The author is interested in correcting some misconceptions such as (1) Church's Thesis is not merely a Thesis its... more than that, (2) Recursive Function Theory is a terrible name for the field and historically inaccurate.

With regard to (1)–While people still call it *Church's Thesis* people really do believe its true. With regard to (2)–he has won the battle- people now call the field *computability*. But for the author this is not just a name change and not even just to clarify to others what we do (though that is part of it). It is a correct terminology based on history.

4 Opinion

Who should read this book? This question breaks down into several questions:

What level is this book at? Someone who knows no computability (but is good at math) could pick up this book, read it, and learn something. It would be tough going but they could. This could certainly be a classroom text on the topic though one would have pick which chapters to cover carefully.

Is the book well written? Absolutely yes. Robert Soare knows the topic, knows the origins, knows not just how the proofs go but why they go that way, and shares that with us.

People in field X should read this book. Fill in the X. In 1980 I would have said that logicians and computer scientists (especially theorists) should learn computability theory (I had a course from Michael Stob out of a preprint of Soare's old book in 1980.) What about in the year 2016?

1. *Computer Scientists:* Since 1980 computer science theory has shifted away from logic and towards combinatorics. Most theorists today have never seen a priority argument! I am *not* going to say *in my day we all did priority arguments while walking to school, uphill both ways, without shoes in the snow* In fact, I think its healthy for fields to evolve and change and not be stuck where it was 30 years ago. And the oracle results prove that the shift was needed. But see next note.

2. *People interested in Randomness:* This interest goes through logic, computer science, physics, and probably other fields. There has been a huge increase of interest in randomness. There is an 800 page book on the subject: *Algorithmic Randomness and Complexity* by Downey and Hirschfeldt. Computability theory is at the heart of the study of randomness. Hence people who read (or want to read) Downey/Hirschfeldt should also read Soare.

3. *Logicians:* Given a theorem whose proof is nonconstructive (defined in a variety of ways) is there a more constructive proof? How to measure this? The Reverse Math program deals with these issues and computability theory is an important part of that.