The Book Review Column[1]

by William Gasarch
Department of Computer Science
University of Maryland at College Park
College Park, MD, 20742
email: gasarch@cs.umd.edu

Editorial about Book Prices:

In the first draft of this column I had the following comment about Beck's book on Combinatorial Games:

The books is expensive and not online. I assume that the author is more interested in getting the material out to people who want to read it rather than in making money. Yet the book has a high price tag and is not online. The entire math-book-publishing business needs to be rethought.

The problem is not hypothetical. I mentor many high school and college students on projects. This book has a wealth of ideas for projects; however, I am not going to ask my students to shell out $65.00 for it. Hence I have not had any students work on this material.

When I emailed a link of the first draft to all authors-of-books, authors-of-reviews, and publishing-contacts Lauren Cowles of Cambridge Press (Beck's publisher) responded that (1) I didn't have prices listed for the other books (true at the time but since corrected), and (2) Beck's book is not bad in terms of price-per-page. (3) Most books are not online, including the other ones in the column.

I interpret her argument as *don't single out Beck's book since most books are overpriced and not online*. I agree with her point entirely. It would be easy for me to blame the industry; however, I do not know what the economics of this industry is. I can however suggest what WE as a community can do:

1. Only buy books used or at a cheaper than list price.

2. If you write a book then insist that the contract allow for it to be online for free. This is not as outlandish as it sounds: (1) *Blown to Bits* by Abelson, Ledeen, Lewis (which I reviewed in this Column) is *not* a book for a narrow audience like most books reviewed here. It is a popular book about computers and society. It is available online for free. (2) Some book companies of Math and Science books allow the author to have the book free on line.

3. When assigning a course textbook allow them to get earlier editions that are usually far cheaper.

End of Editorial about Book Prices

In this column we review the following books.

1. **Combinatorial Games: Tic-Tac-Toe Theory** by Jozsef Beck. Review by William Gasarch. Many theorems in Ramsey Theory can be made into games in several different ways. Each game gives rise to the following question: for which parameters does which player win? This book is a comprehensive study of this question. Many results here are the authors and appear here for the first time.

2. **Algorithmic Adventures: From Knowledge to Magic** by Juraj Hromkovič. Review by Antonio E. Porreca. This is an introduction to theoretical computer science which keeps the math and formalism to a minimum. Hence it can tell people without a math background about our field.

3. **Applied Algebra: Codes, Ciphers and Discrete Algorithms** by Darel W. Hardy, Fred Richman, and Carol L. Walker. Review by Yulai Xie. As the title suggests this book covers a lot of material. Its intended use is for a first course in cryptography; however, there is enough material for a year long course. It will be suitable for a variety of courses.

4. **Models of Computation - An Introduction to Computability Theory** by Maribel Fernández. Review by José de Oliveira Guimarães. The best description of the audience and content come from the preface: **to advanced undergraduate students, as a complement to programming languages or computability courses, and to postgraduate students who are interested in the theory of computation.**

5. **Handbook of Weighted Automata** Edited by Manfred Droste, Werner Kuich and Heiko Vogler. What happens if, in automata theory, you also assign to a word extra information (weights). What changes? What questions can you ask? What questions can you answer? What can you model? Read this book to find out.

6. **Matching Theory** by László Lovász and Michael D. Plummer. Reviewed by Haris Aziz. A Match in a graph is a set of edges that share no vertices. This is a very powerful and deep concept that permeates much of graph algorithms. This book is a comprehensive survey of graph matching theory.

7. **Introduction to Mathematics of Satisfiability** by Victor W. Marek. Review by Stephan Falke. We all know that SAT is NP-complete; however, SAT solvers seem to, as the name indicates, solve SAT problems quickly. This book provides a mathematically rigorous foundation for the theory and practice of SAT solvers.

8. **Elements of Automata Theory** by Jacques Sakarovitch. Reviewed by Shiva Kintali. There is much in Automata Theory that does not make it to the textbooks. This book is an attempt to include almost all that is known. It is mostly a success. As an example it has 750 pages on finite automata.

9. **Combinatorial Pattern Matching Algorithms in Computational Biology Using Perl and R** by Gabriel Valiente. Review by Anthony Labarre. The area of pattern matching has found applications and new problems from problems in biology. This book is an account of pattern matching algorithms, and the code for them, for biologists. The book contains no proof or theorems, but has references to the relevant literature, which should suffice.

10. **In Pursuit of the Traveling Salesman** by William J. Cook. Review by Haris Aziz. This book tells the tale of how TSP has captured the minds of various mathematicians and computer scientists in history and how the quest to solve TSP has led to many breakthroughs in the field of combinatorial optimization. It appears to be the layman's version of a more advanced co-authored by William Cook.

11. **Permutation Patterns** Edited by Steve Linton, Nik Ruškuc, Vincent Vatter. Review by Karolina Sołtys. Permutation pattern theory studies the classes of permutations containing or avoiding a certain pattern (a sub-permutation) or a set of patterns, usually with the goal of enumerating them or obtaining some structural relations between them. There are many surveys for the reader who is unfamiliar with the topic.

Review of[2] of
Combinatorial Games: Tic-Tac-Toe Theory
by Jozsef Beck
Cambridge Press, 2008
720 pages, Hardcover, Hardcover $160.00, Paperback $65.00
Review by
William Gasarch gasarch@cs.umd.edu

1 Introduction

Recall van der warden's theorem:
Theorem For all k, for all c, there exists $W = W(k, c)$ such that for all c-colorings of $\{1, \ldots, W\}$ there exists a monochromatic arithmetic sequence of length k. (Henceforth a mono k-AP.)

Now consider the following games with parameters k, W: In all of the games the players alternate coloring the numbers in $\{1, \ldots, W\}$ RED and BLUE. What differs is the win condition. We will assume that RED goes first.

1. The first player to create a mono k-AP in his color wins.
2. The first player to create a mono k-AP in his color loses.
3. If RED creates a mono k-AP then he wins. Otherwise BLUE wins.

Games of the last type are called *Maker-Breaker* games and are the main topic of this book, though it also covers some material on the other two types as well. Why are Maker-Breaker games studied instead of the other ones? Because very little is known about the other ones. The reasons for this are discussed in the book.

All of the games have a much more general form on hypergraphs. We define just the Maker-Breaker version.
Definition: Let H be a k-ary hypergraph. The *Maker-Breaker Game Associated with H* is the game where (1) players alternate coloring vertices (Player I uses RED, Player II uses BLUE), and (2) if any edge is monochromatic RED then Player I (Maker) wins, otherwise Player II (Breaker) wins.

Consider the Maker Breaker game with parameters k and W. where the board is the $W \times W$ grid and the hyperedges are the $k \times k$ grids. The following is a sample theorem from the book: For large W

- If $k \leq \lfloor \sqrt{\log_2 N} + o(1) \rfloor$ then Maker wins.
- If $k > \lfloor \sqrt{\log_2 N} + o(1) \rfloor$ then Breaker wins.

2 Summary

The book is divided into four parts each of which has several chapters. There are 49 chapters and 4 appendices.

[2] ©2012, William Gasarch

2.1 Part A: Weak Win and Strong Draw:

In the Maker-Breaker game it is Maker's goal to obtain a mono edge in his color. There are games where he does this but does not do this first— that is, Maker can win but not get the mono edge first. This is called a *Weak Win*. It is of some interest to determine when Maker gets a strong Win as opposed to a weak win.

Let S be a set of points in the plane. Maker and breaker play a game where they color points in the plane. Maker wants to get a RED set of points that is congruent to S. In this part they show that (1) Maker wins the game (and they give bounds for how many moves this takes), (2) there is a set S such that Breaker has a strategy so that he (Breaker) will get a blue copy of S first, though of course Maker will still win. This is an example of a weak win.

This part also look at multidim tic-tac-toe in the standard way: the first player to create a monochromatic line wins. In this game a player has a *strong draw* if he has colored a vertex in every edge, thus preventing his opponent from winning. Even though tic-tac-toe is not of Maker-breaker type, there is an extensive analysis of it.

Maker-breaker games based on Ramsey Theory are also discussed. Theorems from Ramsey Theory often are of the form *No matter how X is 2-colored then a mono substructure of type Y must exist.* In game terms you could say that if two players play, each wanting to get a mono Y, one of them will win *even if they are both very bad at the game.* In these cases player I has a win since coloring an edge first is an advantage. One can ask if X is smaller than the bounds given by Ramsey Theory, and the players play perfectly, then is it a draw?

Winning strategies are given using potential methods: every board position is assigned a potential based on how good it is for Maker. Maker tries to maximize the potential and Breaker tries to minimize it. For certain types of hypergraphs this leads to interesting results about who wins and how many moves it takes. One stands out: the Erdos-Selfridge theorem gives a criteria for when Breaker wins which works well for hypergraphs of low degree and where the hyperedges do not overlap much. This is the case for games based on van der Warden's theorem.

2.2 Part B: Basic Potential Techniques- Game Theoretic First and Second Moments

This part has many more examples of the potential method's use to see who wins. We give two examples from the book.

Maker and Breaker alternate taking subsets of $\{1, \ldots, N\}$. Maker wins if there is some $A \subseteq \{1, \ldots, N\}$ such that $|A| = k$ and *every* subset of A is RED.

Note that there is no Ramsey-theory here. There *are* 2-colorings of the set of all subsets of $\{1, \ldots, N\}$ such that for every set of size k there are subsets that are RED and there are subsets that are BLUE. Namely, color a set RED if it has an even number of elements and BLUE if it has an odd number of elements. Hence it is not at all clear that for large enough N Maker has a winning strategy. He proves that if $k = (1 + o(1)) \log_2 \log_2(n)$ then Maker has a winning strategy. This is done by actually proving a statement about a slightly different game.

2.3 Part C: Advanced Weak Win— Game Theoretic Higher Moments

This chapter uses more sophisticated mathematics to obtain theorems about the types of games discussed above and also some new games. We describe two of the games discussed here but not

in earlier chapters.

The *Picker-Chooser Game:* We give an example. The game is played with two parameters N, q. The board is the edges of K_N. In each round Picker picks two uncolored edges and then Chooser chooses one of them to be BLUE (the other one is colored RED). If ever there is a RED K_q then Picker wins; else Chooser wins. Fix N. The *achievement number* is the least q such that with parameters N, q Picker has a winning strategy. It is known that the achievement number is

$$\lfloor 2\lg(N) - 2\lg\lg(N) + 2\lg(e) - 1 + o(1) \rfloor .$$

It is clear how to generalize this game to any hypergraph.

The *Discrepancy Game:* We give an example. The game is played with three parameters N, q and $1/2 < \alpha \leq 1$. The board is the edges of K_N. This is a Maker-Breaker game except that instead of maker winning when he gets a mono K_q, he wins if there are q vertices such that $\alpha\binom{q}{2}$ of the edges are RED. Fix N. The *α-achievement number* is the least q such that with parameters N, q, α Maker has a winning strategy. It is known that the α achievement number is

$$\left\lfloor \frac{2}{1-H(\alpha)} \left(\lg N - \lg\log N + \lg(1 - H(\alpha) + \lg(e) - 1 \right) - 1 + o(1) \right\rfloor .$$

2.4 Part D: Advanced Strong Draw— Game Theoretic Independence

In this chapter games are decomposed into simpler ones leading to theorems about games from the Hales-Jewitt theorem, van der Warden's theorem, and others.

3 Opinion

Jozsef Beck has done a tremendous amount of work in this area. Many results appear in this book for the first time. This is a great book that brings many (all?) of the results in this field under one roof.

There is very little math prerequisite (some combinatorics and probability) yet the material is still difficult. The book is well written but still hard to get through. Once you get through it you will have learned a great deal of mathematics.

Who could read this book? With enough patience and time anyone with some mathematical maturity could. Realistically the audience is very very very smart high school students, very very smart math undergraduates, very smart grad students, and smart professors. The main prerequisite is that you care about this material.

Review of[3]
Algorithmic Adventures: From Knowledge to Magic
by Juraj Hromkovič
Springer, 2009, 376 pages, $59.95
363 pages, hardcover

Review by
Antonio E. Porreca (porreca@disco.unimib.it)
Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy

1 Introduction

Algorithmic Adventures: From Knowledge to Magic by Juraj Hromkovič is an introductory book to theoretical computer science. The author describes some of the most important areas of the discipline (and some less basic ones) while keeping the use of formalism and mathematics to a minimum whenever possible, and focuses on unexpected and fascinating results. The main goal of the author is to show that computer science is not a dull, purely technical subject but instead, quoting his own words, it is *full of magic*.

2 Summary

The first chapter of the book, *A short story about the development of computer science or why computer science is not a computer driving license*, begins by listing some of the common misconceptions about the discipline, particularly arguing against the identification of computer science with a set of ICT skills. The author then discusses the importance of *definitions* and *axioms* (in the informal sense of "principles") for science, and of *implication* in proofs. The search for a procedure for deciding mathematical truths, as proposed by David Hilbert, is cited as the origin of the concept of *algorithm*, hence of computer science itself.

The second chapter *Algorithmics, or what have programming and baking in common?* describes what an algorithm is by using the classic recipe metaphor, which is analyzed in detail in order to show the differences between the two concepts. The importance of the choice of the basic operations allowed is stressed. Programming is then described as the activity of implementing algorithms; a simple programming language, together with a few complete programs, is presented for this purpose.

The third chapter, *Infinity is not equal to infinity*, covers enough basic set theory to be able to prove that different infinite cardinalities exist, as shown by Cantor's diagonal argument. Here Hromkovič discusses extensively why the existence of bijections (which he calls "matchings") between sets is the right way to compare their sizes, instead of a more naive approach based on the subset relation, which fails for infinite sets.

This discussion about cardinality has the consequence, described in the fourth chapter *Limits of computability*, that there exist uncomputable real numbers (i.e., their decimal expansion is not

the output of any program). Once again by diagonal argument, a concrete undecidable set is then described. Reductions between problems are also introduced, together with their use in proving existence or non-existence of algorithms (in particular, the undecidability of the halting problem is proved).

The fifth chapter is titled *Complexity theory or what to do when the energy of the universe doesn't suffice for performing a computation?*. Here the author explains that decidability of a problem is usually not enough, and that *efficient* solutions are needed. Then he introduces a simple time complexity measure for programs and analyzes some examples from this point of view. The importance of the notion of polynomial time is discussed, and the notion of NP-completeness is informally described. As an example, the author describes a polynomial time (Karp) reduction of the vertex cover problem to the problem of checking the existence of Boolean solutions to a set of inequalities. As a remedy against NP-hardness, approximation algorithms are introduced via the standard 2-approximation for vertex cover.

Chapter six, *Randomness in nature and as a source of efficiency in algorithms*, begins with a discussion about the historical views on randomness. Then, the author shows how randomness can be exploited to obtain efficient solutions where deterministic ones are impossible. The example is a $O(\log n)$ communication complexity randomized protocol for deciding equality, which can be repeated several times in order to exponentially decrease the probability of error. No deterministic protocol for equality has sublinear communication complexity.

Cryptography, or how to transform drawbacks into advantages, the seventh chapter of the book, introduces "a magical science of the present time". After a brief historical excursion, including Kerckhoffs' principle, the author describes one-time pad encryption and why it is often impractical although unbreakable. Protocols for secret sharing are then introduced, via the usual padlock metaphor. The author shows that these *cannot* be implemented in a secure way by using the XOR operation with two secret keys (a simple example of cryptanalysis); the Diffie-Hellman protocol is presented as a working solution. Public-key cryptosystems based on one-way functions are informally described, together with digital signatures.

The last three chapters discuss relatively recent areas of computer science, or areas which are unusual for an introductory book. The eighth one is *Computing with DNA molecules, or biological computer technology on the horizon*. Here Hromkovič describes the structure of DNA, how it can be used to encode information and which basic operations (useful from a computational standpoint) can be performed in a laboratory setting. Finally, he describes Adleman's experiment, which provided a concrete implementation of an algorithm for finding Hamiltonian paths in graphs.

The ninth chapter *Quantum computers, or computing in the Wonderland of particles* informally describes some counter-intuitive properties of particles, the double-slit experiment, how qubits work in terms of probability amplitudes and how they collapse into a classic state when measured. The operations on qubits are described as unitary matrix transformations. Due to the modest mathematical background assumed, no concrete quantum algorithm is described.

The final chapter, *How to make good decisions for an unknown future or how to foil an adversary*, deals with online algorithms. These are algorithms which receive their input in pieces during the execution, instead of having it all available before starting. Since they need to choose a strategy based on partial information, an adversary (a model of the worst-case input) may exploit this weakness by making the algorithm perform poorly. A computer memory paging problem is shown not to possess online algorithms with a good competitiveness (i.e., the ratio between the cost of the solution it finds and the cost of the optimal solution); once again, randomness is shown to be

useful in finding a good solution with high probability, for the problem of allocating work stations in a factory to a multi-step task.

3 Opinion

I think the author is successful in identifying a set of intriguing topics, which clearly show that computer science is full of surprises. In particular, undecidability, which arose during the very first steps of the discipline itself, and cryptography are well-known sources of "magic" results. Of course, the book is not meant to be a comprehensive catalog of all aspects of computer science, not even as far as the more theoretical side is concerned, but the selection of topics is ample enough, and likely to light a spark of interest in the mathematically-minded readers.

The ideal audience of *Algorithmic Adventures* probably consists of undergraduate students with an interest in learning the fundamentals of computer science; the later chapters of the book feel a bit too technical for the general public, particularly since the number of references provided is limited.

I completely agree with the author on the necessity to explain what computer science is really about, and on his enthusiasm about the "magic" results which have been obtained (an enthusiasm which clearly shows through in his writing), and I believe this book is a step in the right direction. I really liked the presence of a chapter on set theory, which alone proves the existence of unsolvable problems, and I find that several portions of this books may also be useful to graduate students or researchers trying to familiarize themselves with an unknown subject: for instance, having only a vague understanding of communication complexity, I found the description of the randomized protocol for equality an excellent introduction. The single chapters of the book, which are also available in electronic format, might also be used as introductory reading material for courses on the specific topics.

Although my overall opinion is positive, I found some negative points about this book. The bibliography is very limited, and some topics are presented without references; hence, the reader interested in learning more about set theory, computability or complexity is mostly left on his own. The chapter on quantum computing is not very successful in conveying the fundamental ideas about the topic, which probably require a lot more space and a more mathematical and physical treatment in order to be tackled effectively. Finally, I found the historical and philosophical remarks scattered around the chapters a bit sketchy and imprecise. Nonetheless, this is a good "tourist guidebook" for those who want to take a deeper look at computer science.

Review of [4]
Applied Algebra: Codes, Ciphers and Discrete Algorithms [5]
Author: Darel W. Hardy, Fred Richman, and Carol L. Walker
Publisher: CRC Press, 2009, 424 pages, $107.95
ISBN 978-1-4200-7142-9, PRICE US$ 89.90
Second Edition

Reviewer: Yulai Xie (xie@cse.buffalo.edu, xie.yulai@yahoo.com)

1 Overview

This book is an introduction to two topics: data security and data integrity. Data security, i.e. Cryptography, is the study of making messages secured and visible only to the designated people even though the messages could be captured, intercepted or read by a third party. Data integrity is how to minimize data corruption which is introduced in data transmission. Issues of data integrity are handled by error-control codes.

In data security and data integrity, there are several topics:

1. code: a systematic way to represent words / symbols by other words / symbols.
2. cipher: a method for disguising message by replacing each letter by another letter / a number / some other symbol.
3. cryptography: the art of disguising messages so that only designated people can read them.
4. error-control code: how transmission errors can be detected and even corrected efficiently.

Data security and data integrity have been studied for centuries by mathematicians, physicists, and computer scientists. They have been extensively used in military, astronautics, governments, commerce and other fields. In order to get basic knowledge on these two fields, this book is recommended to readers.

2 Summary of Contents

This introductory text develops the discrete algorithms in Coding Theory and Cryptography. By covering some required knowledge on Number Theory, Linear Algebra and Abstract Algebra, the authors are able to introduce various ideas and algorithms early without the need for preliminary chapters developing sophisticated machinery. Some algebraic concepts are used as the primary tool in developing the theory. The authors present many practical methods for solving problems in data security and data integrity.

The content of this book covers the following topics:

1. Basic knowledge in Cryptography, Error-control Codes, Chinese Remainder Theorem, Fermat Little Theorem, Euler's Theorem, Finite Fields, Galois Fields, Rings,Polynomials,Groups, Polynomial Interpolation and others needed for this field.

[4] ©2012 Yulai Xie
[5] © Yulai Xie, 2010

2. Some widely used codes and their applications, including ACSII code, Morse Code, Bar Code, Hamming Code, BCH Codes and others;

3. some well-known ciphers and their applications, including some substitution and permutation ciphers like Block Ciphers, Unbreakable Ciphers (one-time pad), Enigma Machine, and some advanced ciphers including RSA Ciphers, Knapsack Ciphers and Rijndael Block Ciphers;

4. some high level encryption algorithms and protocols, including Electronic Signatures, Digital Signature Standard, Security Multiparty Protocols, Kronecker Interpolation, Fast Fourier Interpolation and others.

The authors have tried to build each chapter of the book around some simple but fundamental algorithms. Thus, Chapter 1 covers some very basic mathematical concepts and notation which are used in the rest of this book. Chapter 2 introduces some different kinds of codes which are mostly used in the transfer of information. Chapter 3 mainly develops the Euclidean algorithm. Chapter 4 introduces basic aspects of cryptography and covers some well-known substitution and permutation ciphers. Chapter 5 shows the importance of error-control codes to detect and even correct transmission errors. Chapter 6 is built on the Chinese Remainder Theorem and its usage to calculate greatest common divisor of polynomials. Chapter 7 is about two well-known and classical theorems in number theory: Fermat Little Theorem and Euler Theorem, which are very useful in Public Key protocols and Finite Fields. Chapter 8 introduces the famous public key ciphers RSA and Knapsack and Digital Signature standard.

Starting from Chapter 9, the concept and application of Finite Fields (i.e. Galois Fields) are studied. Chapter 9 unifies the basic knowledge on Galois field, ring of polynomials and multiplicative group; the author tried to provide the reader for the preliminary background on advanced encryption standard in chapter 11 and 12.

Chapter 10 covers the BCH error-correcting codes to correct multiple errors, which is a polynomial code built on finite fields. Chapter 11 is mainly for the advance encryption standard in addition to the introduction of Rijndael Block Cipher.

Near the end of the book in Chapter 12, some interpolation algorithms are shown to lead naturally to the secure multiparty protocols, while the efficient Fast Fourier Interpolation is also studied.

Finally, in Appendix A, the required mathematical background knowledge for this book are thoroughly reviewed. One of the main tool used in developing theory is that of number theory, especially the use of Mod arithmetic, Chinese Remainder theory, and Euler and Fermat theory. This seems the most natural tool for studying ciphers as well as the one most familiar to undergraduate students of mathematics and computer science. Basic knowledge on Field, Ring and Group are also covered.

Solutions to some exercises are also provided for the reader's convenience to verify their understanding of the text. An interactive version on CD-Rom is available for interactive examples and self-tests.

3 Style

The most distinguished feature of this book is that the authors develop just enough of the mathematical knowledge to solve practical problems. The authors do not intend to provide a general

study on abstract algebra and number theory but mainly focus on their applications in cryptography and error-control codes. As for data security, some basic discrete algorithms on cryptography are studied; as for data integrity, some knowledge on error-control codes are introduced. This presentation differs from the traditional ones by a more extensive use of the only required knowledge on elementary number theory, linear algebra and abstract algebra, and by a certain emphasis placed on basic applications, rather than on machinery or random details.

To maintain the proper balance between ideas and facts, the authors have presented a large number of examples that are computed in detail. Furthermore, a reasonable supply of exercises is provided. Some factual materials of classical discrete algorithms in cryptography and coding founds their places in these exercises. A good point for self-learners is that hints or answers are given for some of the exercises for self-tests.

The authors tried to prepare this book so that it could be used in more than one type of applied algebra course. Each chapter starts with an introduction that describes the material in the chapter and explains how this material will be used in reality or later in the book.

Although there is enough material in the book for a full-year course (or a topic course), the author tried to make the book suitable for a first course on cryptography and coding for students with some background in linear algebra and abstract algebra.

4 Opinion

This book, which draws significantly on students' prior knowledge of linear algebra and abstract algebra, is designed for a one-year course at the senior or graduate level in mathematics and computer science. The preferred prerequisites for reading this book are linear algebra, number theory and abstract algebra. Student completing linear algebra, at least operations on matrix, and abstract algebra, at least group and field theory, and number theory, a certain familiarity with Mod arithmetic, now have a fair preliminary understanding of the content of this book. It should be apparent that courses on the junior level or senior level by using this textbook, even with majors in mathematics, should draw upon and reinforce the concepts and skills learned in abstract algebra. Although all the required mathematical knowledge are reviewed in this book, students without learning them before still possibly find hard to understand and grasp the critical features of these concepts which can significantly help the students to go through the relating applications.

There is probably more material here than can be covered comfortably in one semester except by students with unusually strong backgrounds. For a short one-quarter or one-semester course (10-14 weeks), we suggest the use of the following material: Chapter 1,2,3,4, and 5, each with 1.5 weeks; Chapter 6 and 7, 2.5 weeks for each, and Chapter 8 - 1.5 weeks. The advanced part of this book can be introduced in a separate semester: Chapter 9 - 3 weeks, 10 - 2 weeks, 11 - 2 weeks, and 12 - 3 weeks, in addition to the coverage of some prerequisite knowledge. The 10-week program above is on a pretty tight schedule. A more relaxed alternative is to allow more time for the first two chapters and to present survey lectures, and on the last week of the course, on Chapter 3, 4, and 5. In one semester course (normally 14 weeks), it can be taught more leisurely and the instructor could probably include additional material (for instance, Chapter 6 and 7).

In summary, this is a good book on introduction of cryptography and error-control codes with the algorithms which have been practiced successfully.

Review of[6]
Models of Computation - An Introduction to Computability Theory
by Maribel Fernández
Springer, 2009, 196 pages, $49.95
184 pages, Softcover

Review by
José de Oliveira Guimarães, josedeoliveiraguimaraes@gmail.com

1 Introduction

Classical computability is the study of functions from the natural numbers to the natural numbers [4]. These functions are expressed through *models of computation*, which describe formally how the computation of a function should take place. The best known models of computation are the Turing machine, the Lambda Calculus, and the partial recursive functions. This last model has its origin in the work of Dedekind [2], which was the first to show satisfactorily how to define a function using recursion. The idea is equal in form to the principle of induction. A function f can be defined by specifying: a) the value of $f(0)$ and b) how to compute $f(n+1)$ using the value of $f(n)$. In the 20s and 30s of the previous century several logicians tried to define "effective computability", what can be really computed using only a finite number of steps and finite values. Among these logicians are Skolen, Herbrand, Gödel, Church, and Turing. At the same time Hilbert launched his *program* with the goal of formalizing Mathematics using finite means in such a way that all the mathematics could be derived from a consistent logical system consisting of axioms and rules. However, it was not clear what he wanted to say by "finite means". As an example, he did not define a model of computation to be used in the rules of logical deductions (which rules are valid?). Hilbert main goal was to create an algorithm for his logical system that, given a formula, would answer whether it was provable or not. This is called Entscheidungsproblem or decision problem and it was posed without first defining "algorithm".

Some of the definitions of "effective computability" available in 1936 were a) representability in formal systems b) recursive functions d) Lambda Calculus, and f) Herbrand-Gödel computability. However, there were doubts whether these systems really captured the intuitive notion of "effective computability", a computation that ends after a finite number of steps and that could be carried by finite means. Then in 1936 Turing [5] presented his wonderful machine that settled the matter: effective computability means computable by a Turing machine. This is called the Church-Turing Thesis. By the way, Turing and Church also showed that, if we assume the Church thesis, there is no algorithm to decide whether a first-order formula is provable or not. And it was proved that every function computed by a Turing machine can also be computed by any other of the proposed models.

2 Summary

The book, as the title says, is about models of computation, which are formal definitions on how to make a computation. It is divided in two parts. The first treats what the author calls

"the traditional models of computation": Turing machines and automata, the Lambda Calculus, recursive functions, and logic-based models. The second part presents models based on objects, interactions, concurrency, and a final chapter on more recent models such as quantum computing. Let us describe each chapter in detail.

Chapter 1, Introduction. It is the introduction that defines "model of computation" and gives examples of non-computable functions.

Chapter 2. Automata and Turing Machines. This chapter is an introduction to automata theory including finite state machines, pushdown automata, and Turing machines. It has far less material than usually taught in courses on formal languages and automata. It has many examples but few theorems — the presentation is far more informal that in an automata book, which is nice given the objectives of the book. The pumping Lemma for finite automata is proved but the pumping Lemma for push-down automata is only informally stated. The many examples make this chapter easy to read, even to undergraduates (hopefully). Unfortunately only half a page is given to the universal Turing machine. Its importance demands much more considering that universality is of fundamental importance when studying models of computation. The chapter ends with a section on Imperative programming. It concludes that modern computers are implementations of Turing machines (although vastly different, modern computers are close to Turing machines than to recursive functions or Lambda Calculus functions).

Chapter 3. The Lambda Calculus. A nice introduction to the subject with many examples and a careful explanation of the details of this calculus. This is very important because the apparent simplicity of Lambda calculus hides a confusing syntax which is rather difficult to master to undergraduate students. The chapter explains the syntax, definition of free variables, α equivalence, β reduction, substitution, arithmetic functions (representability of numbers, successor, and addition), booleans, the fixed-point theorem, and the relation of the Lambda calculus to functional programming. The theorems are stated informally within the text. The relation of Lambda calculus with functional languages is commented but the book does not try to precisely relate a functional language to the calculus. Typed Lambda calculus is not presented.

Chapter 4. Recursive Functions. This chapter describes the partial recursive functions starting with the primitive recursive functions (zero, successor, projection, composition, and primitive recursion). Examples of several primitive recursive functions are given, including sum, multiplication, predecessor, and bounded minimization. The Ackermann function is cited as an example of a function that is not primitive recursive but no proof is given (that would not be so simple for an undergraduate book). The text presents the minimization operator and the Kleene normal form theorem, although it seems that this theorem could be better explained. The final section relates, in four pages, recursive functions with functional programming languages. Here some small examples of fragments of programs are given which explain types, Curryfication, call by name and value, and lazy evaluation. My impression is that this section should appear in Chapter 3 on Lambda calculus.

Chapter 5. Logic-Based Models of Computation. This chapter basically describes the programming language PROLOG. There is only a small formal part that is used to explain the unification algorithm. The text defines clauses (facts, rules, and goals) and then the unification

algorithm (the non-deterministic application of six transformation rules). PROLOG is only slightly related to logic. After explaining the principle of Resolution (SLD), the final section shows the limits of the resolution algorithm that PROLOG uses. The whole chapter is full of examples and it is easy to understand.

Chapter 6. Computing with Objects. Abadi and Cardelli [1] created the Object Calculus, in which objects play role similar to functions in Lambda calculus. This chapter describes this calculus and relates it to object-oriented programming — there are some examples in Java that are associated to some examples in Object Calculus. The first sections present the language and reduction rules of Object Calculus with some relationships with object oriented languages. Then it is proved that this calculus has the same computational power as the Lambda calculus. The rest of the chapter gives more examples and tries to combine objects and functions.

Chapter 7. Interaction-Based Models of Computation. This is the longest and most challenging chapter of all. It describes interaction nets, a graphical model of computation specified by a set of agents and a set of interaction rules. The chapter describes interaction nets and how they can represent the arithmetical operations (addition and multiplication). The Turing completeness of the model is proved using combinatory logic. Finally the text presents a textual calculus for the model and explains its properties.

Chapter 8. Concurrency. The concurrent model is somehow different from the other models since the computation may not produce a result. In these cases the computation is only useful because there are interactions with the environment. Think in an operating system, for example. It does not produce a result but it interacts with programs, mouse, monitor, HD, Internet, etc. This chapter describes a process calculus based in CCS (calculus of communicating systems) [3]. It initially shows the characteristics of concurrent systems (non-determinism, interference among different processes, communication among processes). Then the text describes formally the calculus for process communication. Real examples are only shown in the beginning of the chapter. Examples in the other sections would be welcomed and they would make the text easier to understand.

Chapter 9. Emergent Models of Computation. This chapter gives a brief overview of emergent models of computation: bio-computing and quantum computing. The bio-computing section explains in three pages two classes of models: membrane calculi (based on biological membranes) and protein interaction calculi. The quantum computing section explains in two pages the very basics of quantum mechanics and quantum computing.

Every chapter has many exercises, many of them with answers at the end of the book. However, it seems that some exercises demand more skills than those brought by the book.

3 Opinion

In the preface, the author writes to whom the book is addressed: "to advanced undergraduate students, as a complement to programming languages or computability courses, and to postgraduate students who are interested in the theory of computation". If used as indicated, this is a magnificent book. Most of the chapters have exactly the amount of material that should be taught in an undergraduate course. And in a style that students can understand, with plenty of examples and some relationships between theory and practice. Only chapters "Interaction-Based Models

of Computation" and "Concurrency" are more demanding — they are theoretical and with few examples that are attractive to undergraduate students.

References

[1] Abadi, M. and Cardelli, L. *A Theory of Objects*. monographs in Computer Science, 1996.

[2] Biraben, Rodolfo Ertola. *Tese de Church (Churchs Thesis)*. Coleo CLE, Unicamp, 1996.

[3] Milner, R. *Communicating and Concurrency*. Prentice-Hall, 1989.

[4] Odifreddi, Piergiorgio. *Classical Recursion Theory — The Theory of Functions and Sets of Natural Numbers*. Elsevier Sience Publishers B. V., 1989.

[5] Turing, Alan. On Computable Numbers, With an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 42 (2), 1936.

Review of[7]
Handbook of Weighted Automata
Edited by Manfred Droste, Werner Kuich and Heiko Vogler
Springer, 2009, 625 pages, $189
626 pages, Hardcover

Review by Michaël Cadilhac michael@cadilhac.name

1 Introduction

When considering a language (i.e., a set of words over a finite alphabet), one may be interested in adding some information to each word. Mappings from Σ^* to a *weighting* structure S are thus of interest. When defining recognizers for those mappings, a natural algebraic structure for S emerges: that of a semiring — think of a base set with addition and multiplication, together with neutral and absorbing elements. Mappings from Σ^* to a semiring are called *formal power series*; this book studies some of their classes, generalizations and applications.

A natural thing to do when considering adding weights to a recognizer, say to a nondeterministic finite automaton, is to put them on its transitions. Now, what is the weight associated with a word w recognized by the automaton? (That is to say, what is the series described by the automaton?) This is where semirings become natural: for a successful run labeled w on the automaton, the weight of this run is defined as the product of the weights of the transitions. Then the weight of w is the sum of the weights of all successful runs labeled w.

For instance, suppose the underlying semiring of a weighted automaton is the so-called tropical semiring $\langle \mathbb{N} \cup \{\infty\}, \min, +, \infty, 0\rangle$, where min is the addition and $+$ the multiplication. If each transition has weight 1, then the weight associated with a word w is the length of the shortest successful run on w (or ∞ if none exists). Weights can model numerous notions, from probability of apparition to ambiguity in a grammar, in this very framework.

This book covers extensively both theory and applications of formal power series, focusing on both classical and state-of-the-art results.

2 Summary

This heavy book is divided into four parts; Part I is focused on the mathematical foundations of the theory — not to be confused with an *introduction* to it; Part II deals with the notion of weighted automata, focusing on (algebraic and logical) characterizations, and algorithms; Part III studies other notions of weighted structures (e.g., weighted pushdown automata) and the weighting of elements other than words; Part IV gives applications of the theory to compression, fuzzy languages, model checking and natural language processing. Most of the chapters are self-contained, and the few dependencies are explicit in the text. In the following, S usually denotes a semiring, M a monoid and Σ a finite alphabet containing at least a and b.

Part I Foundations In Chapter 1, Manfred Droste and Werner Kuich introduce the basic notions needed thorough the handbook, in a formal way. The concepts of *semirings* (in particular Conway semirings) and *formal power series* are presented in a limpid, pleasant style, based on a very few requirements (in general algebra). A *semiring* is a set S together with two binary operations $+$ and $\cdot$, and two constants 0 and 1 such that $\langle S, +, 0\rangle$ is a commutative monoid, $\langle S, \cdot, 1\rangle$ is a monoid, the obvious distributivity laws hold, and 0 is absorbing. Now, *formal power series* are mappings from Σ^* to a semiring S, which *weight* words on Σ by elements of S. The chapter then goes on to prove that the formal power series form a semiring which inherits most of the properties of S. Then an overview of the resolution of a particular kind of linear equation on formal power series is given.

In Chapter 2, Zoltán Ésik gives an extended overview of the theory of fixed points of functions. Studies of both the conditions of existence and the resolution techniques are presented. This is one of the most involved chapters, and a complete understanding requires some knowledge in set theory. The aim of this chapter is to present fixed points as a tool from which many results on weighted languages can be succinctly deduced. Although some inconsistency can be detected (some concepts being used before being defined), this chapter offers a comprehensive view, very adapted for a handbook. It may not be needed to read it thoroughly to understand the rest of the book.

Part II Concepts of Weighted Recognizability In Chapter 3, Zoltán Ésik and Werner Kuich introduce the definition of finite automata on semirings. The first half is dedicated to produce Kleene-Schützenberger type theorems, i.e., that recognizable (by an automaton) is equivalent to rational (expressed by a rational expression). A finite automaton over an arbitrary power series semiring is an automaton with transition labels in $S\langle\Sigma\cup\{\epsilon\}\rangle$, the set of polynomials with coefficients in S, i.e., series mapping only a finite number of words to non-zero values. For instance, a transition of an automaton working on $\mathbb{N}\langle\{a, b, \epsilon\}\rangle$ can have $3a + 5ab$ as a label, which is a series mapping a to 3, ab to 5 and any other word to 0. The behavior of such automaton is given as a series associating each word with the sum of the product of the weights over all its paths. The rational languages are defined as series obtained from the elements of $S\langle\Sigma^*\rangle$ by finitely many applications of sum, product and star. It is then proved that languages recognized by those automata are essentially the same as rational languages. The second half of this chapter is dedicated to infinite words. A generalization of the Büchi automaton is presented after the necessary algebraic structures have been defined and investigated. A Kleene-Büchi type theorem is given, proving the equality of rational languages with recognizable languages in the context of infinite words. The chapter ends on the study of a generalization of regular grammars in this context.

In Chapter 4, Jacques Sakarovitch gives a self-contained presentation of rational and recognizable series — for a slightly different definition of recognizable. The contents here basically go in 60 pages through a few chapters of Sakarovitch's book [Sak09] that span more than two hundred pages. The presentation focuses on a different view of rational series, defined under different assumptions. In order for some basic operation on series to be well-defined (namely the Cauchy product), the previous chapter made a natural assumption on the type of semiring S used. Here, the motto is that no restriction should be put on the semiring, but rather on the set of words (viewed as a monoid). The proposed constraint is that the monoid M should embed a homomorphism $M \to \mathbb{N}$, which is a *length* — rest assured that the usual Σ^* monoids have one of these. The text then proceeds to show that this restriction is well-suited, i.e., that the series over such a monoid accept a Cauchy product and a star operation. This in turn leads to the definition of rational series and

rational expressions, and the proof that the two express the same series. The author then defines weighted automata, and gives some fundamental theorems — among which, that automata and expressions denote the same series (a Kleene-like theorem). After an overview on morphisms of weighted automata, focus is shifted to another formalism for *recognizing* series, by means of morphisms from M to the semiring of matrices with entries in S. Under the assumption that M is a free monoid Σ^*, the Kleene-Schützenberger theorem, equating rational series with those recognized by morphisms, is proved. This section is concluded with the study of other notions of product on recognizable series, leading to a presentation of series over a product of monoids, with weighted generalizations of results on the morphic image of rational sets. The author then focuses on series on free monoids, and gives results on so-called *reduced* representations of rational series (which are not unique). This leads to one of the main goals of this chapter, the study of the decidability of the equivalence of two series. The chapter is concluded with results on the classification of the *support* of a series (i.e., the elements with a non-zero weight).

In Chapter 5, Manfred Droste and Paul Gastin present the links between weighted automata and logics, in the same flavor as Büchi's and Elgot's theorems equating regular (and ω-regular) languages with languages definable in monadic second-order logic (MSO). The chapter begins with a few facts on MSO and weighted automata, mainly for completeness. A logic for series (with free monoids and arbitrary semirings) is then devised in a syntactical way, for both finite and infinite words, and great details on the technicality of its definition are given. Two perks of this logic are that, first, it *extends* MSO, in the sense that if the underlying semiring is the Boolean semiring then the logic is the same as MSO; second, the classical corollary that MSO is the same as $\exists$MSO carries through this weighted logic. After a normal form of formulae (so-called *unambiguous*) is presented, it is shown that weighted MSO is effectively equivalent to rational series. Then larger logics are inspected, and it is shown that, for particular semirings, no expressive power is gained. The two last sections focus on infinite words, and translate the results of the previous sections in this framework. The chapter is closed with open questions.

In Chapter 6, Mehryar Mohri presents the algorithms associated with weighted automata and weighted transducers (defined in a natural way). These algorithms are the weighted counterparts of the classical unweighted ones, but their devising requires either a whole new technique or the study of the conditions of application. The first section presents the definitions, in particular of transducers, where each transition carries some weight in addition to the input and output labels. Then the focus is put on shortest-path algorithms, giving different approaches according to the properties of the semiring. The goal is to find, for two states p and q of the automaton, the word with minimal weight from p to q. The section presents algorithms for both the all-pairs and the single-source problems. Closure properties of weighted transducers are then investigated. The next sections focus on quite a lot of algorithms: transducer composition, intersection, difference, epsilon removal, determinization, weight pushing, and minimization. A thorough study of the complexity of these algorithms is given.

Part III Weighted Discrete Structures In Chapter 7, Ion Petre and Arto Salomaa present a theory of algebraic power series and weighted pushdown automata. Recall that algebraic language is just another name for context-free language. The first section presents algebraic power series in terms of algebraic systems of polynomial equations of a special form. Some characterizations and normal forms for algebraic power series are given. The section presents close connections between algebraic systems and the standard notion of context-free grammars and languages, namely that

there is a natural correspondence between the two, and that a language is context-free if and only if it is the support of an algebraic series (with a natural underlying semiring). Also, with $\mathbb{N}$ as the underlying semiring, algebraic systems naturally weight a word with its degree of ambiguity, leading to some neat results in grammar ambiguity. The next section gives an overview of weighted algebraic transductions, and investigates some normal forms and some links with rational transductions. The last section presents a theory for weighted pushdown automata, and, after giving some normal forms on the automata, shows that they express exactly the algebraic series.

In Chapter 8, Juha Honkala presents a generalization of Lindenmayer systems (L-systems), in which Σ^* is replaced by $S\langle\Sigma^*\rangle$; as a particular case, if S is the Boolean semiring, then this is precisely the usual definition of L-systems. Those systems (more precisely, their deterministic version) are a way to study free-monoid morphims or substitutions, and their iterations; the goal of this chapter is to translate those techniques in the context of series, thus allowing the study of endomorphisms of $S\langle\Sigma^*\rangle$. The first section gives the basic definitions of (classical deterministic) L-systems and draws connections with rational power series. The next section introduces formally the concepts of (weighted) Lindenmayerian algebraic (and rational) systems and series, and investigates the relationship between those classes. The last two sections study the series defined by *deterministic* L algebraic systems and related decidability questions, and consider other restrictions of L algebraic systems.

In Chapter 9, Zoltán Fülöp and Heiko Vogler present a theory for weighted *tree* automata and transducers. This self-contained and very complete chapter defines a tree series to be, as previously, a mapping from (finite and ranked) trees to weights, and studies weighted versions of the usual structures for tree recognition and transduction. Again, the consistency check resides in the fact that, when the underlying semiring is the Boolean semiring, the structures are to be the same as their classical unweighted counterparts. The main goal of this chapter is to lift up most of the results in the classical theory to the weighted theory. After giving the required definitions, in particular of tree series, the authors present closure properties, behaviors of supports of (recognizable) tree series, determinization of automata, pumping lemmas, decidability results, finite algebraic characterizations of recognizable tree series and MSO-definable tree series. Moreover, a study on other models of recognizability (akin to those used for usual tree automata) and a list of further results conclude this extensive tour. The last two sections focus on weighted tree transducers, including (de)composition results and a study of the related hierarchy. Again, this study is concluded with a presentation of other models for weighted tree transducers, and a list of further results.

In Chapter 10, Ina Fichtner, Dietrich Kuske and Ingman Meinecke present a theory for weighted concurrent computations, and consider weighted automata on pictures (defined as two-dimensional words) and on more general objects. In the first section, the model for weighted concurrency is chosen to be asynchronous cellular automata, and the relevant definitions are given, in particular three different but closely related notions of behavior. Then a logic is devised, inspired from the weighted MSO of Chapter 5, and it is proved to be effectively of the same expressiveness as weighted asynchronous cellular automata. Some more effective characterizations are presented, in particular a notion close to rational expressions. The next section presents a way to account for the execution time of the parallel events, by way of using a second, compatible semiring and defining weighted branching automata. An investigation of the expressive power of this model and a characterization in term of so-called series-rational expressions over bisemirings is given. The last section focuses on pictures by adding weighting capabilities to the (quadrapolic) picture automaton, an automaton on two-dimensional words which does a one-way read of the image, going in the four directions at

once, and reaching (possibly) accepting states on the borders. Again, characterizations by means of a logic and a formalism of expressions are given.

Part IV Applications In Chapter 11, Jürgen Albert and Jarkko Kari propose digital image compression as a natural application of weighted automata. If the weight of a word is taken to be its *color*, then automata can describe fractal-like grayscale images. In this framework, inference algorithms exist for constructing weighted automata representations of given images, and standard minimization techniques can be used to minimize the number of states. The first section presents the formalism in which images are described. As the usual underlying semiring for this task is $\mathbb{R}$, considerations about the *resolution* of the image are then discussed. The next two sections focus on the problem of going from a weighted automaton to an image, and the converse problem: finding inference algorithms. The authors then give a practical image compression technique, which is compared to the standard JPEG compression routine. One of the perks of those techniques is then investigated: some image transformations can directly be done on the compressed image, by means of weighted transducers. The chapter ends with a study of the fractal nature of this model and some open questions.

In Chapter 12, George Rahonis presents generalizations of fuzzy languages in the framework of power series. Recall that a fuzzy language L is a fuzzy subset of a free monoid M, i.e., $L : M \to [0, 1]$, in other words, a fuzzy language is a formal power series over the semiring $\langle [0, 1], \max, \min, 0, 1\rangle$. This chapter generalizes fuzzy languages to series over bounded distributive lattices. After the basic definitions are given, the author defines multi-valued automata and derives a special case of the Kleene-Schützenberger theorem for fuzzy languages. A few other characterizations are given, including a MSO logic, and a Myhill-Nerode-like theorem equating recognizable fuzzy languages with fuzzy languages having a finite number of classes by the Nerode congruence. Fuzzy languages over infinite words are also investigated. The author then studies the links between the properties of a semiring and the fuzzy languages defined by series over such semiring. The chapter ends with two applications to fuzzy languages, in pattern recognition and event systems.

In Chapter 13, Christel Baier, Marcus Größer and Frank Ciesinski present model checking of probabilistic systems in the light of weighted automata. The motivation for this study is that a Markov decision process (MDP) can be seen as an instance of a weighted automaton where the weights are elements of $]0, 1]$. This allows both a quantitative and qualitative analysis of such processes, and gives a mathematical background for the study of infinite words on such processes. The chapter begins with the basic definitions, in particular of MDPs, and a few examples. The authors then carry out a study of regular and ω-regular properties — expressed as (ω-)automata, and present an approach to derive probabilities for a property from a small fragment of a MDP. The chapter is concluded with the introduction and study of *partially observable* MDPs.

In Chapter 14, Kevin Knight and Jonathan May propose natural language processing as an application of weighted (tree and word) automata — where the weights represent some probabilities. In the first section, examples of the use of weighted (word) transducers are given, including weighting a correspondence between an English word and a Katakana word, and word-for-word translation techniques. The next section deals with other examples, including phrase-for-phrase translations, speech recognition and summarization. The last section focuses on weighted tree transducers, in particular in the context of translation. The chapter is closed with a few open questions.

3 Opinion

This book is an excellent reference for researchers in the field, as well as students interested in this research area. The presentation of applications makes it interesting to researchers from other fields to study weighted automata. The chapters usually require only some familiarity with automata theory and a taste for algebra. One of the main arguments in favor of this handbook is the completeness of its index table — usually a faulty section in such volumes. The chapters are globally well-written and self-contained, thus pleasant to read, and the efforts put to maintain consistency in vocabulary thorough the book are very appreciable.

It is worthy to make some comparisons with [Sak09], a recent book offering a coverage of the same theory. One good thing is that there is not much overlap between the two books, except, of course, for Sakarovitch's chapter (Chapter 4). Their contents are pretty much orthogonal, as Sakarovitch's book starts with usual automata theory, and deepens rational concepts (transductions, relations, etc.), while this handbook starts with foundations in pure mathematics and puts an emphasis on *adding weighting capabilities* to structures, weighting objects other than words, and real world applications.

Finally, only a very few overlaps with the *Handbook of Formal Languages* [RS97] can be found: in Vol. 1, Chap. 9, Werner Kuich introduces formal power series, and in Vol. 3, Chap. 10, Karel Culik II and Jarkko Kari cover the same subject as in Chap. 11.

References

[Sak09] Jacques Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, New York, NY, USA, 2009.

[RS97] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of formal languages, vol. 1-3.* Springer-Verlag New York, Inc., New York, NY, USA, 1997.

Review of [8]
Matching Theory
László Lovász and Michael D. Plummer
AMS Chelsea Publishing, 2009, 547 pages, $79.00
547 pages, Hardcover

Review by
Haris Aziz aziz@tcs.ifi.lmu.de
Ludwig-Maximilians-University Munich, 80538 Munich, Germany

1 Introduction

Matching is a general term with varying interpretations such as in game theory, social choice theory, and mechanism design [9] and also in pattern matching and stringology [2]. This book concerns matchings strictly in the graph theoretic sense where a matching is a set of edges without common vertices. Matching theory has been a fundamental topic in graph theory [3] and combinatorial optimization [8] with deep connections with other topics in the field such network flow theory. The book in question attempts to provide a comprehensive survey of graph matching theory.

2 Summary

In Chapters 1 and 2, classical results concerning matchings in bipartite graphs are presented and their close relation with network flows is described. Results covered include the following closely related theorems: *Konig's Minimax Theorem*, *Hall's Theorem*, *Frobenius' Marriage Theorem*, *Dilworth's Theorem*, *Menger's Theorem* and *Max-Flow Min-Cut Theorem*. The *Marriage Theorem* characterizes bipartite graphs which have perfect matchings (which match all the vertices of the graph).

In Chapter 3, fundamental results concerning non-bipartite graphs are discussed. These include the following related results: *Tutte's Theorem*, *Gallai's lemma* and *Berge's formula*. Tutte's theorem provides a characterization of graphs with perfect matchings. The chapter focuses on the *Gallai-Edmonds Structure Theorem* which states that any graph can be decomposed into a canonical decomposition where the vertices of the graph are partitioned into 1) D, vertices not in every perfect matching, 2) A vertices adjacent to D, and 3) C the remaining nodes, which have a perfect matching. The graph induced by C is *elementary* if the union of all its perfect matchings is a connected subgraph. In Chapter 4, elementary bipartite graphs are examined and '*ear decompositions*' are introduced. This discussion is carried over to elementary general graphs in Chapter 5.

Chapter 6 generalizes results concerning matching to those of *2-matchings* (subgraphs having all degrees at most 2). In Chapter 7, matching related problems are formulated as linear programs. Also, standard polytopes related to matchings are studied. Chapter 8 takes an algebraic graph theoretic perspective to matchings and presents various results in which properties of graph matrices are related to graph matchings. Various generating functions are also considered in which the constants in the polynomials represent properties such as the number of k-element matchings.

Chapter 9 presents and analyzes *Edmond's Matching Algorithm* which is a combinatorial method to compute maximum matchings of a general graph and also provides a subroutine to compute maximum weight matchings of general weighted graphs. Other approaches are also discussed.

Chapter 10 and 11 consider different extensions. Chapter 10 generalizes perfect marching to f-factors which are spanning subgraphs having a given degree f for each vertex. Chapter 11 extends matching theory to matching in matroids. In the last chapter (Chapter 12), maximum matchings are related to maximum independent sets and minimum vertex covers. Hypergraph-matchings are also considered.

3 Opinion

The book not only presents key results in matching theory in an organized fashion but also manages to clearly explain other topics in combinatorics and theoretical computer science along the way. An important characteristic of the book is the algorithmic perspective on problems in matching theory. Throughout the book, comparisons are drawn between algorithmic problems within matching theory and other optimization problems. This gives a wider perspective to the reader.

The book provides a window into the history of developments in combinatorial optimization [10]. This trend is evident from the offset, when the authors trace the developments in matching theory from the contributions of Peterson [1] and König [6], show how the *Hungarian Algorithm* (to compute maximum weight matching in bipartite weighted graphs) was a milestone in combinatorial algorithms and later discuss the impact of advances in linear programming. The clearly written text is aided by the occasional self contained 'Boxes' which contains short explanatory notes on relevant topics. The topics of the boxes include matroids, NP-completeness, minimizing submodular functions and probabilistic methods in graph theory.

Since the book is the second edition, I wanted to compare it with the original edition [7]. The new edition is an exact reprint of the original one with the same page numbering but with an additional appendix which contains an overview of further developments in matching theory. The appendix covers fifteen new developments including randomized approaches to counting matchings. A word of warning to the readers: the part from the first edition is untouched so that even errors from the first edition are not removed. However, errata of the first edition are included at the end of the book.

The book is a solid effort and contains most of the important facts and ideas concerning matching theory. I only had a few quibbles. The topic of hyper-graph matching is not given too much coverage with only a small section in Chapter 12. The appendix could have longer discussions. I also found Edmond's Matching Algorithm to be more easily explained in [4]. Since the book considers an algorithmic flavor, a notable omission is the *Gale-Shapley algorithm* for the *stable marriage problem*. This may have been unavoidable because the topic of stable marriage algorithms is itself vast [5]. In any case, the book is a tour de force in presenting a variety of results in a coherent and unified fashion.

The conclusion is that the book has been the important text on graph matching theory and with such a clear treatment of classic theorems and algorithms, it will remain the canonical text on such a fundamental and ever growing area.

References

[1] M. Christiansen, J. Lützenr, G. Sabidussi, and B. Toft. Julius Petersen annotated bibliography. *Discrete Math.*, 100(1-3):83–97, 1992.

[2] M. Crochemore and W. Rytter. *Jewels of stringology.* World Scientific Publishing Co. Inc., River Edge, NJ, 2003.

[3] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics.* Springer-Verlag, Heidelberg, 2005.

[4] A. Gibbons. *Algorithmic graph theory.* Cambridge University Press, Cambridge ; New York, 1985.

[5] D. Gusfield and R. W. Irving. *The stable marriage problem: structure and algorithms.* MIT Press, Cambridge, MA, USA, 1989.

[6] D. König. *Theorie der endlichen und unendlichen Graphen.* Chelsea Publ. Co., New York, USA, 1936.

[7] L. Lovász and M. D. Plummer. *Matching Theory (North-Holland mathematics studies).* Elsevier Science Ltd, 1986.

[8] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[9] A. E. Roth and M. A. O. Sotomayor. *Two-sided Matching: A study in game-theoretic modeling and analysis.* Cambridge University Press, 1990.

[10] A. Schrijver. On the history of combinatorial optimization (till 1960). *Handbook of Discrete Optimization*, pages 1–68, 2005.

Review[9] of
Introduction to Mathematics of Satisfiability
by Victor W. Marek
CRC Press, 2009, 364 pages, $89.95
350 pages, Hardcover

Review by
Stephan Falke (falke@iti.uka.de)

1 Introduction

Propositional logic is a simple formal system whose formulas are built from propositional variables by means of the unary operator $\neg$ (negation) and the binary operators $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (equivalence). A distinguishing feature of propositional logic is that the propositional variables do not have any inner structure and are only placeholders for the truth values $\top$ (true) and $\bot$ (false).

An important problem is that of satisfiability in propositional logic, i.e., the question of whether the propositional variables of a given formula can be assigned truth values in such a way that the formula evaluates to $\top$ (such an assignment is a satisfying assignment for the formula). Here, the operators $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$ have their usual meaning. The problem of determining satisfiability of a propositional formula is commonly abbreviated as SAT, and a decision procedure for SAT is called a SAT solver.

Stephen Cook showed in 1971 that SAT is an NP-complete problem (in fact, SAT was the first problem shown to be NP-complete). Nonetheless, the past 15 years have seem tremendous progress in algorithmic solutions and modern SAT solvers do not only serve as a decision procedure but also produce a satisfying assignment if one exists. As exhibited by annual competitions, modern SAT solvers are successful and efficient even on formulas with several thousand variables. Thus, SAT solvers have found practical applications in electronic design automation, hard- and software verification, combinatorial optimization, and many other areas.

2 Summary

Introduction to Mathematics of Satisfiability aims at providing a mathematically rigorous foundation for the theory and practice of satisfiability as it is applied in SAT solvers. In encyclopedic breadth, a large number of well-known and little-known properties of propositional logic in general and satisfiability in propositional logic in particular are surveyed. With few exceptions, proofs for all propositions and theorems are provided.

The book consists of a preface, 15 chapters, references, and an index. Each chapter ends with a collection of exercises of varying difficulty.

Chapter 1: Sets, lattices, and Boolean algebras. This chapter reviews basics about sets, posets, lattices, and Boolean algebras. Topics covered include Zorn's lemma, well-founded relations, and the Knaster-Tarski fixpoint theorem.

Chapter 2: Introduction to propositional logic. The second chapter introduces the syntax and semantics of propositional logic and proves fundamental properties of propositional logic. It furthermore introduces the concept of autarkies. Autarkies are partial assignments to the propositional variables that do not change the satisfiability status of a given formula. The author emphasizes autarkies, and they are used in almost all remaining chapters of the book. The concept of duality is briefly touched upon, and the deduction theorem is shown.

Chapter 3: Normal forms of formulas. In many cases, it is convenient to restrict attention to syntactically restricted classes of propositional formulas. For many of these classes, an arbitrary propositional formula can be converted into an equivalent formula in this class (a normal form of the given formula). This chapter introduces the following normal forms: negation normal form (NNF), disjunctive normal form (DNF), conjunctive normal form (CNF), implication normal form (INF), and if-then-else normal form.

Chapter 4: The Craig lemma. The goal of the fourth chapter is to prove two versions of Craig's interpolation theorem. For a tautological formula $\varphi \to \psi$, an interpolant is a formula ϑ containing, at most, the variables common to φ and ψ such that $\varphi \to \vartheta$ and $\vartheta \to \psi$ are tautologies. In its simple version, Craig's interpolation theorem shows that interpolants always exist in propositional logic, while a stronger version of the theorem makes it possible to put further restrictions on the interpolant.

Chapter 5: Complete sets of functors. This chapter investigates the question of whether the choice of the operators $\neg$, $\wedge$, $\vee$, $\to$, and $\leftrightarrow$ in the definition of propositional logic was arbitrary or whether other logical operators might have been chosen instead. This question is answered by an investigation of complete sets of operators, i.e., sets of operators that make it possible to represent all Boolean functions. Several examples of complete sets of operators are given. Furthermore, a complete characterization of complete sets of operators is provided.

Chapter 6: Compactness theorem. The compactness theorem is the most important model-theoretic result about propositional logic. It states that an infinite denumerable set of formulas is satisfiable if and only if every finite subset of the infinite denumerable set is satisfiable. This chapter provides a proof of the compactness theorem that makes use of König's lemma.

Chapter 7: Clausal logic and resolution. The seventh chapter discusses clausal logic (essentially a syntactic variant of formulas in CNF) and introduces resolution as a method to determine satisfiability in clausal logic. It is shown that resolution is both sound and complete for this purpose. In addition to unrestricted resolution, semantic resolution is introduced as a restriction that is still sound and complete.

Chapter 8: Testing satisfiability, finding satisfying assignment. This chapter presents four methods that can be used to determine whether a formula is satisfiable: truth tables, tableaux, the Davis-Putnam (DP) algorithm, and the Davis-Putnam-Logemann-Loveland (DPLL) algorithm. While truth tables and tableaux can be applied to arbitrary formulas, the DP and DPLL algorithms can only be applied to formulas in CNF. Most modern SAT solvers are based on refinements of the DPLL algorithm.

Chapter 9: Polynomial cases of SAT. The algorithms presented in Chapter 8 have exponential complexity in the worst case. There are, however, syntactically determined fragments of propositional logic for which dedicated algorithms with polynomial worst-case complexity can be devised. This chapter presents the following classes and discusses polynomial time algorithms for them: positive formulas, negative formulas, Horn formulas, dual Horn formulas, 2-SAT, affine formulas, and DNF formulas.

Chapter 10: Embedding SAT into integer programming and into matrix algebra. This chapter shows how the satisfiability problem in clausal logic can be reduced to integer programming or to a problem in matrix algebra. Through these reductions, techniques developed in the respective fields can be applied in the context of SAT.

Chapter 11: Coding runs of Turing machines, NP-completeness and related topics. Recall that the algorithms from Chapter 8 have exponential worst-case complexity. This chapter presents a proof that SAT is NP-complete, i.e., asymptotically better algorithms are most likely not possible. NP-hardness is shown by encoding polynomially-bounded runs of Turing machines in propositional logic. Furthermore, this chapter investigates the complexity of the fragments of propositional logic that arise from combining two (or more) fragments with polynomial worst-case complexity from Chapter 9.

Chapter 12: Computational knowledge representation with SAT–getting started. The twelfth chapter begins with examples of how SAT solvers can be used to solve problems that can be encoded in propositional logic. Next, it is shown that formulas in predicate logic over finite domains can be reduced to formulas in propositional logic. This makes it possible to use SAT solvers in the context of knowledge representation using the more convenient framework of (restricted) predicate logic. Finally, cardinality constraints, weight constraints, and monotone constraints are introduced as extensions of propositional logic.

Chapter 13: Computational knowledge representation with SAT–handling constraint satisfaction. Constraint satisfaction problems (CSPs) can be seen as a generalization of SAT. After introducing the needed terminology and properties of CSPs, this chapter shows how SAT can be used for solving CSPs.

Chapter 14: Answer set programming, an extension of Horn logic. This chapter discusses answer set programming (ASP), a form of declarative programming that is based on the stable model (answer set) semantics of logic programming. It is shown that ASP with propositional programs can be reduced to SAT and how ASP can be used for knowledge representation.

Chapter 15: Conclusions. The book closes with concluding remarks.

3 Opinion

Introduction to Mathematics of Satisfiability is a comprehensive review of the theoretical aspects of satisfiability in propositional logic. Assuming only some general mathematical maturity as a

prerequisite, it provides an encyclopedic treatment of well-known and little-known properties of propositional logic in general and satisfiability in propositional logic in particular.

Practical aspects of SAT solving are largely ignored. While the fundamental DPLL algorithm is presented as a high-level pseudo code, none of the implementation techniques and improvements of the basic algorithm that have been developed in the past 15 years are discussed.

The strong theoretical flavor and the amount of material make it a challenge to read the book from cover to cover. Furthermore, the lack of illustrating examples makes it unsuitable as a textbook. However, the encyclopedic nature of the book makes it suitable as a reference book on the theory of satisfiability in propositional logic.

Review of[10]

Elements of Automata Theory

Author: Jacques Sakarovitch, Translator (from French) Reuben Thomas

Cambridge University Press, 2009, 782 pages, $164.00

Reviewer: Shiva Kintali

1 Introduction

During my undergrad I often found myself captivated by the beauty and depth of automata theory. I wanted to read *one* book on automata theory and say that I *know* automata theory. Couple of years later I realized that it is silly to expect such a book. The depth and breadth of automata theory cannot be covered by a single book.

My PhD thesis is heavily inspired by automata theory. I had to read several (fifty year old) papers and books related to automata theory to understand several fundamental theorems. Unfortunately, the concepts I wanted to learn are scattered in multiple books and old research papers, most of which are hard to find. When I noticed that Prof. Bill Gasarch is looking for a review of *Elements of Automata Theory*, I was very excited and volunteered to review it, mainly because I wanted to increase my knowledge about automata theory. Given my background in parsing technologies and research interests in space-bounded computation I wanted to read this book carefully. This book is around 750 pages long and it took me around one year to (approximately) read it. It is very close to my expectations of the *one* book on automata theory.

First impressions : Most of the books on automata theory start with the properties of regular languages, finite automata, pushdown automata, context-free languages, pumping lemmas, Chomsky hierarchy, decidability and conclude with NP-completeness and the P vs NP problem. This book is about "elements" of automata theory. It focuses *only* on finite automata over different mathematical structures. It studies pushdown automata only in the context of rational subsets in the free group. Yes, there is 750 pages worth literature studying only finite automata.

This book is aimed at people enthusiastic to know the subject rigorously and not intended as a textbook for automata theory course. It can also be used by advanced researchers as a desk reference. There is no prerequisite to follow this book, except for a reasonable mathematical maturity. It can be used as a self-study text. This book is a direct translation of its french original.

2 Summary

This book is divided into five major chapters. The first three chapters deal with the notions of *rationality* and *recognizability*. A family of languages are *rationally closed* if it is closed under rational operations (union, product and star). A language is recognizable if there exists a finite automata that recognizes it. The fourth and fifth chapters discuss rationality in relations. *Chapter 0* acts as an appendix of several definitions of structures such as relations, monoids, semirings, matrices, graphs and concepts such as decidability. Following is a *short* summary of the five major chapters. There are several deep theorems (for example, Higman's Theorem) studied in this book. I cannot list all of them here. The chapter summaries in the book have more details.

Chapter 1 essentially deals with the basic definitions and theorems required for any study of automata theory. It starts with the definitions of states, transitions, (deterministic and nondeterministic) automaton, transpose, ambiguity and basic operations such as union, Cartesian product, star, quotient of a language. Rational operators, rational languages and rational expressions are defined and the relation between rationality and recognizability is established leading to the proof of Kleene's theorem. String matching (i.e., finding a word in a text) is studied in detail as an illustrative example. Several theorems related to star height of languages are proved. A fundamental theorem stating that 'the language accepted by a two-way automaton is rational' is proved. The distinction between Moore and Mealy machines is introduced.

Chapter 2 deals with automata over the elements of an arbitrary monoid and the distinction between rational set and recognizable set in this context. This leads to a better understanding of Kleene's theorem. The notion of morphism of automata is introduced and several properties of morphisms and factorizations are presented. Conway's construction of universally minimal automaton is explained and the importance of well quasi-orderings is explained in detail. Based on these established concepts, McNaughton's theorem (which states that the star height of a rational group language is computable) is studied with a new perspective.

Chapter 3 formalizes the notion of "weighted" automata that count the number of computations that make an element be accepted by an automaton, thus generalizing the previously introduced concepts in a new dimension. Languages are generalized to formal series and actions are generalized to representations. The concepts and theorems in this chapter makes the reader appreciate the deep connections of automata theory with several branches of mathematics. I personally enjoyed reading this chapter more than any other chapter in this book.

Chapter 4 builds an understanding of the relations realized by different finite automata in the order they are presented in chapters 1, 2 and 3. The Evaluation Theorem and the Composition Theorem play a central role in understanding this study. The decidability of the equivalence of transducers (with and without weights) is studied. This chapter concludes with the study of deterministic and synchronous relations. *Chapter 5* studies the functions realized by finite automata. Deciding functionality, sequential functions, uniformisation of rational relations by rational functions, semi-monomial matrix representation, translations of a function and uniformly bounded functions are studied.

There are exercises (with solutions) at the end of every section of every chapter. These exercises are very carefully designed and aid towards better understanding of the corresponding concepts. First time readers are highly encouraged to solve (or at least glance through) these exercises. Every section ends with *Notes and References* mentioning the corresponding references and a brief historical summary of the chapter.

3 Opinion

Overall I found the book very enlightening. It has provided me new perspectives on several theorems that I assumed I understood completely. Most of the concepts in this book are new to me and I had no problems following the concepts and the corresponding theorems. The related exercises made

these topics even more fun to learn. It was a joy for me to read this book and I recommend this book for anyone who is interested in automata theory (or more generally complexity theory) and wants to know the fundamental theorems of theory of computing. If you are a complexity theorist, it is worthwhile to look back at the foundations of theory of computing to better appreciate its beauty and history. This book is definitely unique in its approach and the topics chosen. Most of the topics covered are either available in very old papers or not accessible at all. I applaud the author for compiling these topics into a wonderful free-flowing text.

This book is nicely balanced between discussions of concepts and formal proofs. The writing is clear and the topics are organized very well from the most specific to the most general, making it a free-flowing text. On the other hand, it is very dense and requires lots of motivation and patience to read and understand the theorems. The author chose a rigorous way of explaining rationality and recognizability. Sometimes you might end up spending couple of hours to read just two pages. Such is the depth of the topics covered. Beginners might find this book too much to handle. I encourage beginners to read this book *after* taking an introductory automata theory course. This is definitely a very good reference text for researchers in the field of automata theory.

In terms of being used in a course, I can say that a graduate level course can be designed from a carefully chosen subset of the topics covered in this book. The exercises in the book can be readily used for such a course.

This is an expensive book, which is understandable based on the author's efforts to cover several fundamental topics (along with exercises) in such a depth. If you think it is expensive, I would definitely suggest that you get one for your library.

Review of[11]

Combinatorial Pattern Matching Algorithms in Computational Biology Using Perl and R

by Gabriel Valiente
Chapman & Hall/CRC Press, 2009, 368 pages, $92.95

Review by
Anthony Labarre `Anthony.Labarre@cs.kuleuven.be`

Department of Computer Science
K. U. Leuven
Celestijnenlaan 200A - bus 2402
3001 Heverlee
Belgium

1 Introduction

Gabriel Valiente's book is about *combinatorial pattern matching*, a field whose object of study consists of problems of the form: "find one, k, or all occurrence(s) of a given pattern in another given object". The nature of the objects that can be considered varies greatly, encompassing among others texts, trees, graphs, or images, all of which are relevant in the context of computational biology, which is the motivation for and the field of application of this book.

The usefulness of efficient algorithms for searching texts is obvious to anyone who has ever used a computer. This is also the case in biology, where the advent of genome sequencing has led to a dizzying number of databases containing the full sequences of a huge number of organisms. Text search algorithms are particularly relevant in that setting, and are routinely used in discovering patterns in sequences or finding common subsequences in order to gain more insight into the evolution of living beings and their relationships, as well as into the relationships between genotypes (the "source code" of an organism) and phenotypes (the set of physical traits and characteristics of an organism).

But the relevance of combinatorial pattern matching to computational biology goes far beyond text algorithms. The field also has applications in phylogenetics, which focuses on representing the evolution of a set of species using a tree (or sometimes a network) whose leaves are the species under consideration and whose internal nodes represent putative unknown common ancestors. Many techniques for reconstructing phylogenetic trees are known, and there is an obvious need to be able to compare the results these methods yield, which can be achieved by identifying common substructures in the resulting trees. The applications explained above also serve as motivations for having techniques that allow to find patterns in graphs, and to express the similarities between them; indeed, graphs are also used to model evolution in the same way as phylogenetic trees, and model chemical reactions, proteins, RNA structures, protein interaction networks, and metabolic pathways to name but a few.

2 Summary

The book is organized into three large parts, according to the type of structure under consideration: Part I deals with sequences, which constitute the simplest, the most natural and also the most studied setting. Part II is concerned with trees, whose study is motivated by phylogenetic reconstruction methods and RNA structure comparison, while Part III considers graphs, which are useful in representing evolution and biochemical reactions.

Each part is structured in the same way and consists of three chapters: the first chapter states some basic definitions and facts about the structures that are being investigated in that part, introducing simple algorithms which are used as building blocks in the sequel, as well as implementation notes on how to represent the data structures under consideration, listing already implemented options whenever available. The algorithms and operations introduced in that introductory chapter are not about pattern matching, but are part of the general background and are used later on; examples of tasks that are carried out in this chapter include traversing trees and graphs, complementing DNA sequences, generating structures and counting them.

The second chapter introduces some of the building blocks of pattern matching in the context of the given structures. These methods include the fundamental task of finding an occurrence, or several occurrences of a word (resp. a tree, or a graph) in a given text (resp. in a graph), and how to compare two structures. Comparison of sequences is achieved by aligning them, while in the case of graphs it is carried out by computing various statistics on the graphs under comparison, like the set of distances between two terminal nodes or the sets of paths in both graphs, and then comparing those statistics.

Finally, the third chapter examines more advanced tasks, like those of finding common substructures in several objects. This differs mostly from what is tackled in the second chapter by the nature of the information that is extracted. For instance, in the case of graphs, one can look for partitions induced by the removal of an edge, or the subgraphs induced by a particular set of terminal nodes, and then comparing the sets obtained from both structures to be compared. Other more advanced tasks include subgraph isomorphism and maximum agreement common subgraphs.

Valiente concludes the book with two appendices, where the basics of Perl and R are presented in a very didactic way to people who know little or nothing about those languages. The author walks the reader through the design of two simple scripts, offers a brief overview of both languages, and presents what he calls "quick reference cards", which are concise lists summarizing the syntax and usage of essential functions in these languages.

3 Opinion

This book seems geared mostly towards people who have had little previous exposure to computer science or programming and who are, or want to become active in Bioinformatics. More specifically, I would imagine the primary audience of this book to be biologists getting to know about programming, or undergraduate students in computer science getting started in Bioinformatics. It is also very light from a mathematical point of view.

Valiente clearly decided to approach the subjects he treats in his book in breadth rather than in depth, with an emphasis on actually *teaching* those subjects. This kind of book is especially helpful in the context of interdisciplinary fields, of which computational biology is a perfect example, where professionals with a particular background are led to learn aspects from a field that is very different

from their initial training. People who are already familiar with combinatorial pattern matching may benefit from the source code included in the book[12], but if they want to deepen their knowledge of the subjects it covers, I would recommend them to have a look at Gusfield's book [1] (especially for Part I), Valiente's previous book [3] for more advanced topics and algorithms (for Parts II and III), which also contains implementations (although in C++), and Huson et al.'s book [2] for more information on phylogenetic networks, which Valiente discusses in Part III.

As hinted above, the author focuses primarily on basic tasks in combinatorial pattern matching, particularly on those that are most relevant to computational biology. I like the "hands-on" approach this book offers, and the very pedagogical structure it follows, by first clearly stating the problems and their relevance to biology, explaining how they will be solved, giving the algorithm in pseudocode, and then in actual Perl and R source code. The book also has tons of examples, thoughtfully chosen and beautifully laid out, illustrating the various definitions, concepts and problems under consideration. Of course, some basic algorithms have already long been implemented in Perl and R, but it is worth explaining them nonetheless. For other tasks, Valiente informs the reader about existing formats and libraries (the famous BioPerl[13] module for instance) so that he or she does not reinvent the wheel. There is a judicious mention of things that do *not* exist too, which also serves as a justification for showing how things could be done using the aforementioned libraries. This is also useful in making a choice between the two languages that are suggested to you. A lot of alternative options are of course available, including Python and its well-furnished Biopython module[14], for which it is not difficult to adapt the source code snippets presented in this book.

The book contains no proof or theorems, but has references to the relevant literature, should the reader be eager to know more mathematical facts about the objects being dealt with. This is consistent with the approach chosen by Valiente, but it makes me wonder why some counting problems, while interesting, are considered, or why the sequences they yield are not linked to the corresponding entries in OEIS[15]. I was a little bit let down by the index, which could have been thicker. This is a minor issue, but it suggests that you should not take too long to read a particular chapter, otherwise retrieving the definition of that particular term that you forgot may take you some time. Other than that, the book is very well-written and accessible, undoubtedly written by an author who takes great care in preparing his manuscripts and teaching about an area he enjoys working on.

References

[1] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.

[2] Daniel H. Huson, Regula Rupp, and Celine Scornavacca. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press, 2011. http://www.phylogenetic-networks.org/.

[3] Gabriel Valiente. *Algorithms on Trees and Graphs*. Springer, 2002.

[12] Also freely available from the author's webpage at `http://www.lsi.upc.edu/~valiente/comput-biol/`.

[13] See `http://www.bioperl.org/wiki/Main\_Page`.

[14] See `http://biopython.org/wiki/Main\_Page`

[15] The On-Line Encyclopedia of Integer Sequences, see `https://oeis.org/`.

Review of [16]
In Pursuit of the Traveling Salesman
William J. Cook
2012
272 pages, $27.95, Hardcover

Review by
Haris Aziz (aziz@in.tum.de)
Technische Universität München, 85748 Garching, Germany

> "The traveling-salesman problem is one of the great unsolved problems in mathematics, capturing notions of complexity that are at the core of the information age."— W. J. Cook [4].

1 Introduction

The traveling salesman problem (TSP) is one of the most fundamental and ubiquitous computational problems. W. J. Cook has written a book which tries to explain the intellectual challenge of coming up with an efficient way to solve TSP. This book tells the tale of how TSP has captured the minds of various mathematicians and computer scientists in history and how the quest to solve TSP has led to many breakthroughs in the field of combinatorial optimization. It appears to be the layman's version of [3] which is a comprehensive book covering technical aspects in much more detail.

2 Summary

The book consists of twelve chapters.

Chapter 1 ("*Challenges*") introduces the challenge of the TSP and explains the combinatorial difficulty of solving the problem without having to enumerate and compare an exponential number of tours.

In Chapter 2 ("*Origins of the Problem*"), the author explores the origin of the problem. He describes how even before mathematicians examined the problem formally, salesmen, lawyers, and preachers encountered the TSP in their travels. This is followed by research on how TSP was formalized as a scientific problem.

In Chapter 3 ("*The Salesman in Action*"), the modern-day applications of the TSP are outlined which includes mapping genomes, industrial machines, organizing data, scheduling jobs etc.

Chapter 4 ("*Searching for a Tour*") initiates the formal study of the TSP and describes the fundamental approaches to searching for optimal tours.

In Chapter 5 ("*Linear Programming*"), linear programming is introduced and it is explained how LP relaxations are useful in finding good tours efficiently.

In Chapter 6 ("*Cutting Planes*") the cutting-plane method is discussed in detail.

Chapter 7 ("*Branching*") then explores branching in the use of combinatorial optimization.

Chapter 8 titled *"Big Computing"* covers the algorithmic engineering aspect of the TSP. Various practical milestones are mentioned and historical world records in solving big TSP problems are highlighted.

Chapter 9 titled *"Complexity"* consists of an exposition on formal computational complexity including Turing machines; the desirability of polynomial-time algorithms; polynomial-time reductions; NP-complete problems etc.

Chapter 10 and 11 discuss miscellaneous aspects. In Chapter 10 (*"The Human Touch")*, the use of the human mind is compared to the computer. The problem solving strategies of animals are also covered. Chapter 11 ("*Aesthetics*") examined the aesthetic and artistic aspects of the TSP and especially how TSP has been adopted in several works of modern art.

The book concludes with Chapter 12 with anticipation about the next episodes in the unfolding story of the TSP.

The back matter consists of notes, bibliography, and an index.

3 Opinion

TSP is a fundamental problem in combinatorial optimization and theoretical computer science and a number of books have been written which are dedicated solely to TSP [see for example, 3, 6, 5]. This book is unique in the sense that it tries to convey interesting technical ideas to the general public. Passages from the book could easily be used as an accessible introductions to topics in combinatorial optimization. For example, a part of Chapter 5 is a mini-tutorial on linear programming and Chapter 9 presents a gentle introduction to computational complexity.

Not only does Cook make an excellent effort to convey elegant ideas from theoretical computer science to the public but does so in an upbeat and engaging manner. In this effort, Cook helps his cause by conveying the human side of the research story. As in sports and arts, the human aspects of the performance adds an extra layer of public interest. Cook manages to bring his passion for mathematical research to the fore. There are mini-biographies or at least mention of most of the actors involved in story of the TSP. The list includes but is not restricted to Euler, William Hamilton, Menger, Hassler Whitney, Jack Edmonds, Richard Karp, Stephen Cook, Dantzig, Nicos Christofides, Shen Lin, Brian Kernighan, Robert Tarjan, and David Johnson. In this sense, the book is similar in flavor to popular science books by Simon Singh which discuss interesting scientific problems or questions and tell the story of scientific problems which spans decades, continents, and a number of characters and ideas [see for example, 8]. Along with the text, one also notices the photographs of various mathematicians and computer scientists associated with the story of TSP.

Although the book contains many ideas, it is suitable for leisurely reading. Each chapter starts with a well-chosen quotation and focuses on one aspect of TSP. The text in *"In Pursuit of the Traveling Salesman"* is accompanied by figures and pictures. One also encounters humor. For example when the author discusses Stephen Cook, he quips "No relation to me, although I have enjoyed a number of free dinners due to mistaken identity." There is a companion website which further captures the excitement around the TSP [2]. It is evident that a lot of research has gone into retracing the history of TSP and identifying crucial breakthroughs. It would have been appealing had the author made a timeline of the various milestones in the challenge of TSP just as Plummer [7] did for matching theory. I feel that the book serves well for the outreach and dissemination for theoretical computer science. Incidentally a movie titled "Travelling Salesman" but independent

from the book (to the best of my knowledge) is also planned for 2012 which will further help the cause [1].

The book is highly recommended to any one with a mathematical curiosity and interest in the development of ideas.

References

[1] http://www.travellingsalesmanmovie.com/, April 2012.

[2] http://www.tsp.gatech.edu/, April 2012.

[3] D. L. Applegate, R. E. Bixby, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, 2007.

[4] W. J. Cook. The problem of the traveling politician. *NYT Blogs*, 2011.

[5] G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations, Springer,*. Springer, 2006.

[6] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

[7] M. D. Plummer. Matching theory - a sampler: from dénes könig to the present. *Discrete Mathematics*, 100:177–219, 1992.

[8] S. Singh. *Fermat's Last Theorem: The story of a riddle that confounded the world's greatest minds for 358 years*. Fourth Estate, 2002.

Review of[17]
Permutation Patterns
Edited by Steve Linton, Nik Ruškuc, Vincent Vatter
The London Mathematical Society, 2010
352 pages, $83.00, softcover

Review by
Karolina Sołtys ksoltys@students.mimuw.edu.pl

1 Introduction

Permutation pattern theory studies the classes of permutations containing or avoiding a certain pattern (a sub-permutation) or a set of patterns, usually with the goal of enumerating them or obtaining some structural relations between them. For example, the permutation 679238541 contains the pattern 4132 (we can consider the subsequence 9256) and avoids the pattern 1234.

To get some intuition behind the study of permutation patterns, let us consider the sequences that can be sorted using a single stack (with a single pass over the sequence). Knuth showed in 1968 that these are exactly the 231-avoiding permutations (which also are counted by the Catalan numbers), which result actually gave the initial motivation for the area.

Some other early results are the theorem stating that the 123-avoiding permutations are counted by the Catalan numbers and the Erdős–Szekeres theorem, stating that every permutation of length at least $ab + 1$ must contain either the pattern $1, 2, 3, ..., a + 1$ or the pattern $b + 1, b, ..., 2, 1$.

The area has applications to computer science and biomathematics.

2 Summary

Permutation Patterns are the proceedings of the fifth conference of the same name, held in 2007, and are comprised of fifteen articles and a short chapter with notes from a problem session. Eight articles at the beginning are surveys, the others represent several current research directions in the subject. The book covers all the most important aspects of the area.

The first survey, *Some general results in combinatorial enumeration*, while only partially related to permutation patterns, does an excellent job in formalizing several important notions used in combinatorial enumeration, and then goes on to survey the enumeration results in several classes of structures, including, of course, pattern-avoiding permutations.

The second survey, *A survey of simple permutations*, studies various results concerning *simple* permutations, that is the permutations not containing intervals (sets of contiguous indices) such that the value set in the interval is contiguous. The article hints at some applications of intervals and simple permutations in biomathematics, but unfortunately does not elaborate on this point.

The third survey, *Permuting machines and permutation patterns*, will probably be the most interesting for a computer scientist. It relates permutation patterns to the behavior of various data structures used to sort a sequence. This research direction dates back to the work of Knuth

described in the introduction. The article studies the behavior of other so called *permuting machines* and their compositions, using automata-theoretic methods.

The fourth survey, *On three different notions of monotone subsequences*, is a very comprehensive guide to the enumeration results concerning permutations avoiding monotone patterns under three definitions of avoidance.

The fifth and sixth surveys, *A survey on partially ordered patterns* and *Generalized permutation patterns – a short survey*, study generalizations of permutation patterns, which provide an elegant and uniform notation for various combinatorial structures arising in permutations

The seventh survey, *An introduction to structural methods in permutation patterns*, presents the reader with some combinatorial methods for obtaining a generating function enumerating the permutations avoiding certain patterns. One of these methods is based on the interesting relation between permutation patterns and regular languages.

The eighth survey, *Combinatorial properties of permutation tableaux*, covers the relation between permutations and permutation tableaux and the ways to use the properties of permutation tableaux in the study of permutation patterns.

Articles nine to fifteen explores such areas in permutation patterns as enumeration schemes, probability distributions, descents and rises in permutation, algorithmic descriptions of permutation classes, packing density and the relation between permutations and token passing networks.

The final chapter, *Problems and conjectures presented at the problem session*, briefly states several important problems in the area of permutation patterns, and provides for each of them sufficient motivation and references.

3 Opinion

The articles are consistently well written, the proofs are usually explained in enough detail (although it is worth noting that in the surveys most of the results are only stated, with a reference given to the proof), the typographical errors are rare and do not interfere with understanding of the text. The only thing that I seriously lacked while reading the book was the motivation for the results in a broader mathematical context. In that respect, the third survey is especially worthy of attention, because it motivates its results with interesting real-life (or rather, computer science related) examples well enough even to serve as an introductory text to the area of permutation patterns.

I would certainly recommend this book as a reference for researchers in the area. The surveys could also serve as supplementary reading material for an advanced enumerative combinatorics course. However, one should already have some background in permutation patterns to profit from this book; an undergraduate using this book for self-study would probably find the results unmotivated, and therefore not interesting.