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`

Welcome to the Book Reviews Column. We hope to bring you at least two reviews of books every month. In this column six books are reviewed.

1. The following three books are all reviewed together by William Gasarch. **Descriptive Complexity Theory** by Neal Immerman, **Finite Model Theory** by Heinz-Dieter Ebbinhaus and Jorg Flum, and **Descriptive Complexity and Finite Models (Proceedings from a DIMACS workshop)** edited by Neil Immerman and Phokion Kolaitis. These books deal with how complicated it is to describe a set in terms of how many quantifiers you need and what symbols are needed in the language. There are many connections to complexity theory in that virtually all descriptive classes are equivalent to the more standard complexity classes.

2. **Theory of Computing: A Gentle Introduction** by Efim Kinber and Carl Smith, reviewed by Judy Goldsmith. This book is a textbook aimed at undergraduates who would not be happy with the mathematical rigour of the automata books of my youth, such as Hopcroft and Ullman's book.

3. **Microsurveys in Discrete Probability (Proceedings from a DIMACS workshop)** edited by David Aldous and James Propp is reviewed by Hassan Masum. This is a collection of articles (not all surveys) in the area of probability.

4. **Term Rewriting and all that** by Franz Baader and Tobias Nipkow, reviewed by Paliath Narendran. This is intended as both a text and a reference book on term rewriting.

I am looking for reviewers for the following books

If you want a FREE copy of one of these books in exchange for a review, then email me at gasarch@cs.umd.edu. If you want more information about any of these books, again, feel free to email me. Reviewing a book can be a great way to learn a field. I have gotten research ideas that resulted in papers from books I have reviewed.

1. *Parameterized Complexiy* by Downey and Fellows.

2. *A=B* by Petkovsek, Wilf, and Zeilberger. Since the title is not informative, I'll quote from the ad for the book: "This book shows how several recently developed computer algorithms can master the difficult job of simplifying complex summations and if there is no such simplification they will prove this to be the case. The authors present the underlying mathematical theory of these methods, the principle theorms and proofs, and include a package of computer programs that can do these tasks."

3. *Proof, Language, and Interaction (Essay in honor of Robin Millner)* Edited by Gordon Plotkin, Colin Stirling, and Mads Tofte.

4. *Control Flow Semantics* by Jaco de Bakker and Erik de Vink

[1]© William Gasarch, 2000.

5. *Complexity and Real Computation* by Blum, Cucker, Shub, and Smale.
6. *The Clausal Theory of Types* by Wolfram.
7. *Model Checking* by Clarke, Grumberg, and Peled.
8. *Data Refinement: Model-Orientd Proof Methods and their Comparisons* by de Roever and Engelhardt.
9. *Communication and Mobile systems: the π-calculus* by Milner.
10. *Automatic Algorithm Recognition and Replacement: A new approach to program optimization* by Metzger and Wen.

The following are DIMACS workshop books which are collections of articles on the topic in the title.

1. Randomization Methods in Algorithm Design.
2. Multichannel Optical Networks: Theory and Practice.
3. Networks in Distributed Computing.
4. Advances in Switching Networks.
5. External Memory Algorithms.
6. Mobile Networks and Computing.
7. Robust Communication Networks: Interconnection and Survivability.

Reviews [2] of THREE books on Descriptive Complexity Theory
Descriptive Complexity Theory
by Neil Immerman
Published by Springer 1998, 268 pages
Hardcover, $55.00
ISBN number 0-387-98600-6

AND

Finite Model Theory
by Heinz-Dieter Ebbinghaus and Jorg Flum
Published by Springer-Verlag
Perspectives in Mathematical Logic series
Hardcover, $119.00
ISBN number 3540149X

AND

Descriptive Complexity and Finite Models
Edited by Neil Immerman and Phokion Kolaitis

[2]© William Gasarch 2000

Published by AMS 1997, 268 pages
DIMACS series in Discrete Mathematics and Theoretical Computer Science
Hardcover, $55.00
ISBN number 0-8218-0517-7

Reviews by
William Gasarch
University of Maryland At College Park
gasarch@cs.umd.edu

1 Overview

The *Complexity of a Problem* usually means how much time or space is needed to solve it; however, there are other measures of complexity. The three books under review deal with *Descriptive Complexity*– how hard is it to *describe* the problem. For example, the set of graphs $G = (V, E)$ that are not connected can be described as NOT-CONN =

$$\{G \mid (\exists A \subseteq V)(\exists x \in V)(\exists y \in V)[x \in A \wedge y \notin A \wedge (\forall u)(\forall v)[u \in A \wedge E(u,v) \rightarrow v \in A]]\}.$$

Note that this description uses one existential second order quantifier and several first order quantifiers. Is there a simpler description? This question needs refinment. A description is *first order* (henceforth FO) if all the quantifiers range over V. A description is *second order monadic* (henceforth SO(monadic)) if the quantifiers range over V or subsets of V. We showed above that NOT-CONN $\in$ SO(monadic). The following questions arise.

1. Is NOT-CONN $\in$ FO? (No.)
2. What is the weakest descriptive class for NOT-CONN. (The above shows that NOT-CONN can be described with one existential second order monadic quantifier, the rest first order quantifiers, and no additional numeric relations. This is essentially best possible.)
3. How can we prove results about non-expressability? (See below.)
4. Why do we care? (See below.)

1.1 Non-expressibility

Non-expressibility results are proven using Ehrenfeucht-Fraïssé games (henceforth E-F games). The E-F game with m moves on graphs G and H is played as follows.

1. SPOILER puts a pebble a_1 (b_1) on a vertex of G (H).
2. DUPLICATOR put a pebble b_1 (a_1) on a vertex of H (G).
3. Repeat the above two steps $m - 1$ times.
4. If the subgraph of G induced by $\{a_1, \ldots, a_m\}$ is isomorphic to the subgraph of H induced by $\{b_1, \ldots, b_m\}$ via an isomorphism that maps a_i to b_i then DUPLICATOR wins. Otherwise SPOILER wins.

One can prove that if DUPLICATOR can win the m-move E-F game on G and H then no m-quantifier first order formula can distinguish G from H. Hence if, for all m, there exists G and H such that G is connected, H is not connected, and DUPLICATOR wins then neither CONN nor NOT-CONN is in FO. This is indeed the case.

Variants of E-F games exist to prove properties about second order languages, languages with limits on the number of variables, and languages with other operations.

1.2 Why do we care?

There are two main reasons to care about descriptive complexity. Firstly, and somewhat underated, there is curiosity. It *seems* like NOT-CONN $\notin$ FO but one is curious to find out. Secondly, the more cited reason, is the connection between descriptive complexity and well known classes. Fagin [2] showed that NP is equivalent to a descriptive class, and since then virtually every complexity class has an exact characterization as a descriptive class. For example, (1) NP is equal to ($\exists$)SO which is the set of languages that can be described with one second order existential quantifier, and (2) P is FO[LFP] which means FO appended with a second order function that can be obtained as a least fixed point of an operator.

A plausible attack on P vs. NP is to find some set in NP and show by E-F games that it is not in FO[LFP], and hence not in P. This plan of attack is plausible for any classes that have exact descriptive complexities, which is most classes. As of now no separations have been obtained this way; however, there do not seem to be any mathematical obstacles such as relativization. Ironically, there has been a *collapse* of classes proven by thinking about descriptive complexity: NSPACE($S(n)$) is closed under complementation. This was proven independently by Immerman [3] and Szelepcsényi [8]. Immerman proved it while trying to determine if two descriptive classes were the same. Szelepcsényi proved it while looking at inductive counting arguments. The proofs are essentially the same and can be understood without knowing descriptive complexity theory.

2 Review of Immerman's book Descriptive Complexity

Neil Immerman has been one of the leaders in descriptive complexity theory since 1979; hence, he is ideally suited to write a text in the area. His book starts from the beginning and assumes the reader to be mathematically mature but ignorant.

Chapters 1 and 2 are elementary. They cover the basics and clarify some of the implicit assumptions about the languages used throughout the book.

Chapters 3,4,5,7, and 10 prove that many of the usual complexity classes are exactly descriptive classes. The following classes, and others, are equivalent to descriptive classes: CRCW[O(1)] (concurrent read, concurrent write, poly number of processors, constant time), P, NP, PSPACE. The descriptive classes these are equivalent to are, for the most part, natural. These seem to be the kind of theorems that are hard to come up with in the first place, but once you think they are true the proofs are not hard. For example, the proof that a set is in NP iff it can be defined with a second order existential quantifier is proven by a coding of Turing machines into formulas in the obvious way.

Chapters 6,8, and 13 use E-F games to obtain lower bounds on expressibility. Here are some:

1. PATH$_k$ is the set of graphs with distinguished vertices s and t such that there is a path of length $\leq 2^k$ from s to t. PATH$_k$ can be expressed with k quantifiers and 3 variables. PATH$_k$ cannot be expressed with $k-1$ quantifiers (any number of variables) or 2 variables (any number of quantifiers). (Chapter 6.)

2. Hanf's theorem is proven which states a very general condition under which DUPLICATOR has a winning strategy. From this we easily obtain 2-COL and CONN are not in FO. (Chapter 6.)

3. CONN $\notin$ ($\exists$)SO(*monadic*) (without any numeric predicates or ordering). This requires defining a new kind of E-F game for second order existential. (Chapter 8.)

4. REACH is the problem of, given a graph with two distinguished vertices s and t, can you reach s from the t. For undirected graphs REACH is in ($\exists$)(SO)(*monadic*). They show that for directed graphs REACH is not in that class. This required a general theorem and then the graphs used were shown to exist via the probabilististic method. (Chapter 8)

5. For strings, the PARITY predicate is not in FO. This is proven by Hastad's Switching lemma. A full proof is given.

6. A *colored graph* is one whose vertices are colored. There is no constraint on the coloring. Two colored graphs are isomorphic if there is an isomorphism that preserves color. Let CC_k be the set of colored graphs where at most k vertices share a color. Let L^m be the set of first order sentences about colored graphs that have only m variables.

   (a) If $G, H \in CC_1$ and agree on all L^1 sentences then $G \simeq H$.
   (b) If $G, H \in CC_3$ and agree on all L^3 sentences then $G \simeq H$.
   (c) There exists $c > 0$ such that for all n there exists $G, H \in CC_4$ such that G, H agree on all L^{cn} sentences but $G \not\simeq H$.

Chapters 9, 11, and 12 are about tradeoffs of parameters. Chapter 9 has the famous result that NSPACE($S(n)$) is closed under complementation. Chapter 14 contains applications to databases, dynamic complexity, and model checking. Chapter 15 points to future directions. Due to space limitations the book can't spend too much time on any of these topics, but a flavor for them is given.

Neil Immerman has done the community a great favor by compiling all the information into one book that can actually be read. He claims that it can be used as text for advanced undergrads and grad students. This seems reasonable if the students have enough mathematical maturity.

My only objection to the book is in the material it does not cover:

1. Arora and Fagin [1] have a clearer (though longer) proof that REACH is in not in ($\exists$)(SO)(*monadic*) for directed graphs.

2. Turan [9] has shown that AUT (the set of graphs that have a nontrivial automorphism) is not in (SO)(*monadic*). Since this is one of the few results about non-expressibility on (SO)(*monadic*), and it's simple, it should have been included.

3. Kolatis and Vaananen [4] have used sophisticated combinatorics (e.g., Van der Waerden's Theorem) to obtain results about counting classes and infinitary languages.

4. Papadimitriou and Yannakakis have [5] made connections between descriptive complexity and approximation of optimization problems.

Since it is not possible to include everything, Immerman's choice of topics makes sense.

3 Review of Ebbinghaus-Flum book on Finite Model Theory

Finite model theory (henceforth FMT) is a superset of Descriptive Complexity Theory (henceforth DCT) Logicians were working on FMT before the connections to complexity theory were established. The following kind of theorem is part of FMT but not DCT: "Every sentence of a certain type that has a model has a finite one."

The book under review is on FMT and is for mathematicians. As such its choice of topics differs from Immerman's book. There is some overlap in content but no overlap in mentality. I'll first describe the contents then compare the two books.

Chapter 1 is on E-F games for first order. They proof $CONN \notin FO$ and Hanf's theorem. Chapter 2 extends E-F games to second order logic, infinitary logics, and counting quantifiers. A sample application: PARITY cannot be expressed with an infinite formula using only a finite number of variables.

Chapter 3 is on 0-1 laws. The number of structures (e.g., graphs) on n elements is finite. If P is some property then some of the structures have property P and some do not. Hence the question of "how many" or "what fraction" have property P can be asked. For many properties P

$$\lim_{n\to\infty} \frac{\text{Number of graphs on } n \text{ vertices that have property } P}{\text{Number of graphs on } n \text{ vertices}} \in \{0,1\}.$$

This phenomenon is called a 0-1 law. In this chapter they prove that all FO properties satisfy the 0-1 law, there are monadic second order sentences that do not, and other theorems about this phenomena.

Chapter 4 is on finite model properties. Here is a sample theorem and why it is important. If ϕ is a first order sentence with at most 2 variables in a relational vocabulary (no function symbols) then the following, called the finite model property, is true: If there exists M such that $M \models \phi$ then there is a finite such M. Normally the validity question "Given ϕ is it true in all model M (including infinite ones)?" is undecidable. However, if ϕ is in some restricted class of sentences that satisfies the finite model property then the question is decidable. Let $VALID$ be the set of valid sentences. By the completeness theorem $VALID$ is the set of provable ones, hence $VALID$ is a c.e. set. [3] However, $\overline{VALID}$ is c.e. since $\phi \in \overline{VALID}$ iff $(\exists M)[M \text{ finite } \wedge M \models \neg\phi]$. Since $VALID$ and $\overline{VALID}$ are both c.e., both are computable.

Chapter 5 is on finite automata and logic. The following are proven: (1) a language L is regular iff $L \in \mathrm{MSO}[S,<]$ (Monadic second order with symbols for successor and $<$), (2) a language L is star-free iff $L \in \mathrm{FO}[S,<]$. Much more is known about this topic, but not proven here (see the next review, the chapter by Straubing).

Chapter 6 links Descriptive classes to Complexity classes. As such it rephrases questions in complexity theory as questions in logic. Chapter 7 discusses fixed point operators and proves FO(posTC) = FO(TC), more commonly known as "WOW!, NSPACE($S(n)$) is closed under complementation!"

Chapter 8 is on logic programming. A sample theorem: a class is expressible with a stratified Datalog program iff it is is in FO(BFP) (BFP is Bounded Fixed Point). Chapter 9 is a short chapter on connections between DCT and approximating optimization problems. Chapter 10 is on different types of quantifiers such as "$(\exists)$ exactly n"

Comparisons to Immerman's book:

[3] Logicians have recently switched from r.e. (recursively enumerable) to c.e. (computably enumerable). See [6] for the essay that inspired the change.

1. Both books cover E-F games and the equivalence of complexity classes to descriptive classes. Immerman does more equivalences.

2. Topics in Ebbinghaus-Flum that are not in Immerman: infinitary logic, finite model properties, finite automata, and optimization problems. The last two topics get a very brief treatment in Ebbinghaus-Flum

3. Topics in Immerman that are not in Ebbinghaus-Flum: number of variables as a measure of descriptive complexity, lower bounds on expressibility of reachability, tradeoff's between variables and quantifier depth, circuits, lip service to applications like database.

4. Ebbinghaus-Flum requires a higher level of mathematical maturity. There is little introductory material and it is in theorem-proof style. There are not that many examples. (Through there are some, unlike most math books at this level.)

5. Ebbinghaus-Flum is written for mathematicians while Immerman is written for computer science theorists. A startling example: I looked in the table of contents of both books for the results NSPACE($S(n)$) is closed under complementation. In Immerman there was a section named "NSPACE($S(n)$) = coNSPACE($S(n)$)". The proof is in terms of FO(posTC) = FO(TC) (as it should be for this book) but the connection to space is both in the introduction to that chapter and at the end of the chapter and spelled out explicitly. In Ebbinghaus-Flum there is a section called "FO(posTC) and normal forms." In this section is a proof that FO(posTC) = FO(TC). There is no mention of space classes.

4 Review of DIMACS Workshop Proceedings

From January 14–17 in 1996 there was a DIMACS Workshop at Princeton University on Descriptive Complexity and Finite Models. The book under review is one of the results of that workshop. It consists of seven articles by different authors.

4.1 Ronald Fagin's article *Easier Ways to Win Logical Games*

There are three very general theorems that can be used to show that DUP has a winning strategy in an E-F game. The three theorems are stated, compared, and used. They are not proven, and the applications are sketched (not proven). This article inspired me to read [1].

4.2 Bruno Courcelle's article *On the Expression of Graph Properties*

This article contains a chart of several graph properties and which fragment of Monadic Second-Order Logic (henceforth MS) can be used to express it. For example 3-colorability is expressible by a formula of the form $(\exists X_1, \ldots, X_n)[\phi(X_1, \ldots, X_n)]$ where X_i ranges over sets of vertices and ϕ is first order. MS is important since if a property A is in MS then, when restricted to "tree-structured graphs", there exists a tree automata for A. Hence, the set of tree-structured graphs with property A is in $DTIME(n)$. This article contains proofs and many pointers to the literature.

4.3 Howard Straubing's article *Finite Models, Automata, and Circuit Complexity*

This article is about using descriptive complexity on sets of strings. A sample theorem: a language L is regular iff $L \in \text{MSO}[S, <]$ (Monadic second order with symbols for successor and $<$). What about first order? A language L is star-free iff $L \in \mathrm{FO}[S,<]$. What about subsets of $\mathrm{FO}[S,<]$? If $L \in \mathrm{FO}[S,<]$ then L is certainly regular. Let $A = (Q, \Sigma, \delta, F)$ be the minimal DFA for L. δ can be extended to act on strings. For all $w \in \Sigma^*$ let $f_w : Q \rightarrow Q$ via $f_w(q) = \delta(q, w)$. $M(L)$ is the monoid $\{f_w \,:\, w \in \Sigma^*\}$ under composition. The algebraic properties of this monoid can be used to classify descriptions. Sample theorem: $L \in \mathrm{FO}[<]$ iff $M(L)$ is aperiodic and finite. Other algebraic theorems are also shown. These theorems make it possible to prove non-expressibility without E-F games. Circuits are connected to this material via the following theorem: $AC^0 = FO[ALL]$, which is first order with all numerical predicates. Reading this chapter made me want to reread Straubing's book on this material [7].

4.4 Victor Vianu's article *Databases and Finite-Model Theory*

This article surveys the connections between database theory and finite model theory. It claims (correctly) that database theory can be a source of *new* questions for finite model theorists. The main difference between the two, and hence the richest source of new questions, is that database theory deals with *dynamic* aspects such as update languages whereas finite model theorists have mostly studied complexity classes which are static. (See Immerman's book, Chapter 14.2, for some work on dynamic classes.)

4.5 Moshe Vardi's article *Why is Modal Logic so Robustly Decidable?*

Modal logic was invented to model necessity and possibility, but can also be used to model time. This paper looks at propositional modal logic. The well-formed-formulas are (1) propositional variables, (2) if ϕ_1 and ϕ_2 are wff then $\neg\phi_1$, $\phi_1 \wedge \phi_2$, and $\Box\phi_1$ are wff. The symbol $\Box\phi$ means that ϕ is *necessarily true.* An *interpretation* M for a formula ϕ is a (possibly infinite) directed graph G together with, for every vertex s of G, an assignment to all the variables of ϕ. We can define what it means for ϕ to be true at vertex s of interpretation M, denoted $(M, s) \models \phi$ as follows:

1. If ϕ is a propositional variable then $(M, s) \models \phi$ iff M assigns ϕ TRUE at s.
2. $(M, s) \models \phi_1 \wedge \phi_2$ iff $(M, s) \models \phi_1$ and $(M, s) \models \phi_2$.
3. $(M, s) \models \neg\phi$ iff $(M, s) \not\models \phi$.
4. (This is the interesting one.) $(M, s) \models \Box\phi$ iff for all t such that (s, t) is an edge in G, $(M, t) \models \phi$.

A formula ϕ is *satisfiable* if there is some interpretation M and some vertex s such that $(M, s) \models \phi$. Since M could be infinite the question of decidability is of interest. However, the following is known:
Theorem: If ϕ is satisfiable then there exists an interpretation M and a vertex s such that $(M, s) \models \phi$ and M is of size $\leq 2^{|\phi|}$. Hence satisfiability is decidable. (This is called the Finite Model Property.)

The paper under review probes the underlying reasons why this problem (and harder ones) are decidable. Modal logic is closer to first order logic (where SAT is not decidable) then to propositional since the $\Box$ operator is an implicit $\forall$. So why should SAT be decidable?

Modal logic can be embedded in a small fragment of first order logic, called FO^2, where SAT is decidable via a Finite Model Property. The authors could have stopped here, satisfied (the authors that is, not the formulas) that they have drawn a distinction between FO and $MODAL \subseteq FO^2$. However, they point out that a more complicated modal logic still has a decidable SAT problem, while there does not seem to be an analogous subset of FO. In particular Computation Tree Logic (henceforth CTL) where you can quantify over branches is decidable. CTL is decidable by showing it has a Tree Model Property— if a formula is satisfiable then there exists an interpretation (a tree) where it is true (the tree might be infinite but this is not a problem).

This article gives a good survey of Modal Logic with the theme of decidability and, to a lesser extent, complexity. There are few proofs; however there are many references. This article, along with several of the references, could be the content of a graduate course or two.

4.6 E. Allen Emerson's article *Model Checking and the Mu-calculus*

(The reader of this review should read the review in the last subsection first.)

A *reactive system* is a system that has many unpredictable external inputs, such as air traffic control. Temporal Logic (a variety of Modal Logic) seems to be a good way to reason about reactive systems. The question of most interest is to proof correctness of such a system (e.g., there is never a time when you want to land two airplanes on the same runway). This can be formalized as the model-checking problem:

Given a finite state transition graph M, an initial state s_o of M, and a temporal logic specification formula f, does $(M, s_0) \models f$?

The article under review defines CTL and the μ-calculus rigorously, states theorems about expressiveness and model-checking algorithms. There are a variety of model-checking algorithms because there are two size parameters of interest: the size of the model and the size of the formula. The authors own words best summarize his point "The μ-calculus and associated temporal logics such as CTL provide a good handle on precisely stating just what behavior is to occur when, at a variety of levels of detail. The fully automated type of reasoning provided by model checking provides a convenient tool for both verifying correctness and for automatic debugging. Moreover, a number of interesting mathematical problems arise in connection with model checking in the μ-calculus."

4.7 Toniann Pitassi's article *Algebraic Propositional Proof Systems*

The following problem is fundamental: given $C_1 \wedge \cdots \wedge C_m \notin SAT$, give a proof that $C_1 \wedge \cdots \wedge C_m \notin SAT$. There are many approaches to this problem. For several of them it is known that there exists, for all n, a formula on n variables that *requires* $2^{\Omega(n)}$ long refutations.

This paper presents an algebraic proof system. Given a formula $\phi(\vec{x})$ (n variables, m clauses) one can (easily) produce equations $Q_1(\vec{x}), \ldots, Q_{n+m}(\vec{x})$ such that $\phi \in SAT$ iff $(\exists \vec{a})[Q_1(\vec{a}) = \cdots = Q_{n+m}(\vec{a}) = 0]$. A refutation of ϕ is a prime p together with a set of polynomials $F_1, \ldots, F_{n+m} \in Z_p[\vec{x}]$ such that $\sum_{i=1}^{n+m} Q_i(\vec{x})F_i(\vec{x}) \equiv 1 \pmod{p}$. The existence of such polynomials proves that $\phi \notin SAT$. The question of interest is the *size* of $F_1, \ldots, F_{n+m}$. The following are proven: (1) If there are always poly-bounded polynomials then $PH = \Sigma_2^p$. (2) If there are not always poly-bounded polynomials then extended Frege systems do not have poly-bounded refutations (This is a conjecture that has resisted attempts to prove it.)

Several upper and lower bounds are given for particular formulas such as the pigeon-hole principle. The upper bounds are easy. The lower bounds are difficult and not proven; however, some of the methods to prove them are discussed. Connections to Frege systems are made precise.

This article is a nice survey. It can be used to guide the readers to the literature.

5 Final Comments

Let CS be computer scientists who do not work on theory. Let A denote computer scientists who work on algorithms. Let NA be computer scientists who work on theory but not on algorithms (complexity theory, database theory, semantics, etc.)

Let IMM stand for Immerman's book. Let $EFLUM$ stand for the Ebbinghaus-Flum book. Let DIM stand for the DIMACS book.

Let NN denote that there is No Need to have this book in your library. Let L denote that a book should be in your school library. Let S denote that a book should be on your own shelf. Let P denote that a book should be under your pillow at night. Then we have $P \rightarrow S$ and $S \rightarrow L$.

The following table is my advice for people in $\{C, A, NA\}$ with regard to whether they should $\{L, S, P\}$ the books in $\{IMM, EFLUM, DIM\}$.

	CS	A	NA
IMM	L	S	P
$EFLUM$	NN	L	S
DIM	L	L	S

References

[1] S. Arora and R. Fagin. On winning strategies in E-F games. *Theoretical Comput. Sci.*, 174, 1997.

[2] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of Computation, Richard Karp (editor), SIAM-AMS Proc.*, 7, 1974.

[3] N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 1988. Earlier version in Structures 1988.

[4] Kolaitis and Vaananen. Generalized quantifiers and pebble games on finite structures. *Annals of Pure and Applied Logic*, 1995.

[5] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and Systems Sciences*, 43:425–440, 1991.

[6] R. Soare. Computability and recursion. *Bulletin of of Symbolic Logic*, 27, 1996.

[7] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Progress in Computer Science and Applied Logic. Birkhäuser, Boston, 1994.

[8] R. Szelepcsenyi. The method of forced enumeration for nondeterminisitc automata. *Acta Informatica*, 1988. Earlier version in BEATCS in 1987.

[9] G. Turán. On the definability of properties of finite graphs. *Discrete Mathematics*, 49:291–302, 1984.

Review[4] of

Theory of Computing: A Gentle Introduction

[4]© Judy Goldsmith 2000

Authors: Kinber and Smith
Publisher: Prentice-Hall, 2001
Paperback: ISBN 0-13-027961-7, $XX
207 Pages

Reviewer: Judy Goldsmith
University of Kentucky
Department of Computer Science

I have the pleasure of reviewing an important book here. Kinber and Smith's new automata theory and formal languages textbook is a much-needed addition to our shelves. It covers material that is taught in classes that are mandatory or encouraged in many CS departments (automata and formal languages, computability, and a brief intro to complexity focussed on P and NP), yet there are few textbooks that are accessible to our less mathematically inclined undergraduates. The only other comparable book I've used is John Martin's **Introduction to Languages and the Theory of Computation.** Smith and Kinber's book is most similar in tone to Martin's; the next closest that I found was the second edition of Lewis and Papadimitriou's book, **Elements of the Theory of Computing.**

Unfortunately, many undergraduates are intimidated by formal mathematical presentations. This book seeks to introduce the fundamentals of the theory of computing without too much stress. It succeeds admirably.

I have not yet tested this book in the classroom. However, there were (last I checked) two reviews available on amazon.com, presumably from University of Maryland students. Both reviewers gave the book four stars (out of five); one of them said, "My theory of computer languages class was taught with this book and I must say it is a very good introduction." In the many years I have taught automata and formal languages, it has been rare for students to actually *like* an automata theory book.

1 Content Summary

Chapter 1 is the usual intro to sets, etc. It is serviceable and succinct. It contains almost everything that the students might be expected to know about strings and sets. It does not attempt to cover basic graph theory, which would be helpful, for instance when the authors refer to breadth-first search. If a student has had graph theory, one might hope she has also been introduced to sets. Given that the authors chose not to introduce graph theory, the succinctness seems appropriate. The one topic which I often cover in this course that is not introduced is countability and uncountable sets.

Chapter 2 introduces finite automata, nondeterminism, regular expressions, nonregular languages, and closure properties. The equivalence of NFAs and DFAs is sketched, rather than proved. The details are left to the reader, or outlined in exercises. The theorem that regular expressions describe the class of languages that are accepted by FAs is implicit in the discussion at the introduction of regular expressions. Students are likely to miss that, and be puzzled by the algorithm to create a regular expression from an automaton. (The theorem is stated after the algorithms.)

The pumping lemma is stated and used, somewhat informally, to show several languages are not regular. The Myhill-Nerode Theorem is not mentioned.

Chapter 3 introduces context-free grammars first, and gives a few examples of a grammar for a fragment of English; a few sentences are generated. The discussion of parsing mentions leftmost and rightmost derivations, removal of ambiguity, and inherent ambiguity, but does not go into great detail. Pushdown automata are introduced as an extension of FAs by an extra "data structure," namely, the stack, and shown to recognize exactly the context-free languages. Closure properties are discussed, and the pumping lemma for context-free languages is stated and applied. The statement and application are more formal than those of the pumping lemma for regular languages. Chomsky Normal Forms are discussed, as are deterministic CFLs.

Chapter 4 introduces Turing machiness as accceptors and transducers, considers variants of TMs such as multi-headed TMs and RAMs, and shows that they are all equivalent. There's a clear discussion of Church's thesis and a brief introduction to decidability and semi-decidability (what the old-fashioned types call recursive and recursively enumerable sets).

Chapter 5 covers undecidability, introducing the notion of universal TMs and showing that the diagonal set and the halting set are both undecidable. It ends with Rice's Theorem: any property of sets corresponds to a set of programs (in any reasonable system) that is either trivial or undecidable. [The wording is the reviewer's.]

Chapter 6 introduces time complexity and the classes P and NP. It introduces polynomial-time reductions and NP-completeness, and gives several examples, more than in most automata theory or algorithms books. However, it does not mention any other complexity classes or measures (such as space complexity).

Instructor's Manual All of the missing or sketched proofs from Chapter 2 appear here, as well as solutions to all of the exercises.

I understand that the instructor's manual will not be available from bookstores, but only on the request of an instructor. Thus, in theory, it will not fall into students' hands.

2 Style

The overall style of the book is fairly informal. For instance:

> We call finite automata a type of **program.** They hardly look like programs in any conventional programming language. On the other hand, the reader probably has no doubt that they can be simulated by software programs. As our vending machine example shows, they can also be directly implemented on the hardware level.

However, the degree of formality varies through the book. In particular, the chapter on regular languages is presented much less formally than that on CFLs, although formal proofs of all the results for regular languages are given in the instructor's manual. There is no discussion of this choice. Was it to ease the students into the material, or because (for some reason) a student is more likely to need the formal apparatus of CFGs?

Living in a culture of "More is better," it is startling to find a textbook that covers the material for a 15-week semester course, and no more. The exercises seem reasonable and sufficient; it is likely that I would assign at least 50% of them per semester. There are several levels of difficulty, all labeled by zero, one, or two diamonds (the two-diamond ones are the hardest).

This book is written for computer science students, not mathematicians. There is an emphasis on connecting the material to work in other areas of computer science and to real-world programming. The hard-core, "who needs this theory stuff?" crowd will not be persuaded by this presentation, but more of the middle-of-the-road computer science students may actually grasp some of the material and its connections to the rest of the curriculum.

On the other hand, non-native speakers of English may be somewhat put off by the wordinees and an occasional floridity of style. Italics are used both for *emphasis* and for *definitions,* making it a bit more difficult to locate the latter on a page. This will not be a theorist's reference book;

what will make it useful to a student in the years following a course is that it is short, and the table of contents is actually useful in locating results, theorems, and constructions. However, it would have been useful to have some sort of chapter summary or other tool for reference.

3 Opinions

There were several things I disliked about this book. The style sometimes gets a bit florid, and "obviously" and some of its synonymous phrases occur occasionally. (One of the authors assures me that this will be fixed in later editions.) A few topics that I think are important are omitted: the Myhill-Nerode theorem, the notion of space complexity, and at least a mention of other complexity classes, perhaps as a teaser for a complexity theory course. And some of my favorite exercises about semi-decidable sets (show a set is the range of a partially computable function if and only if it is the domain of a partially computable function) appear in the text.

While this book does not do as thorough a job on non-regular languages as Martin's book does, it does a better job on CFLs. An earlier edition of Martin's book had a mildly offensive example of an English sentence in the discussion of parsing natural languages; Kinber and Smith give several remarkably clunky sentences. Is there a reason that they chose to omit articles, leaving them with such ringing examples as "John bought big car"? I think it's important to mention the connection to historical computational linguistics. Perhaps the clunkiness serves the purpose of indicating that CFLs are still not adequate for generating natural languages.

On the other hand, this book is at the right level for my undergraduate students. In particular, it has an excellent introduction to nondeterminism at the automaton level. The exercises look like they have a good range, from completely straightforward to challenging. Having all the solutions in an instructor's manual should make the course more accessible to non-theorist instructors as well.

I like that this is really a one-semester book. I like the gentleness of the text, and its technical correctness. It is not so wordy that it obscures the content, as some recent books have been, but it does take the time to put the material in the bigger picture of computer science. It is likely that (at least some) students will actually read the entire book, and thus will get exposure to and practice with the technical details of the area, and more of the bigger picture than they'd get by skimming a longer textbook.

Review of *Microsurveys in Discrete Probability*
Edited by David Aldous and James Propp
Published by AMS 1998
DIMACS series in Discrete Mathematics and Theoretical Computer Science Volume 41
Hardcover, $39.00
ISBN number : 0-8218-0827-3

Review by
Hassan Masum
Carleton University, Ottawa, Canada
hmasum@ccs.carleton.ca

1 Overview

Microsurveys in Discrete Probability is a collection of articles from a DIMACS Workshop on themes in discrete probability. The main themes include distributional estimates, dynamical percolation, Markov chains, Poisson processes, random graphs, and random sampling.

For those who are not specialists in the above areas, "Microsurveys" may be a misnomer, since many articles are relatively specific journal-format articles instead of more accessible survey articles. From my point of view as a nonspecialist, the articles of most interest are: Dynamic Percolation, Mixing Times, Uniform Spanning Trees, Coupling from the Past, and Perfectly Random Sampling. Each of these articles is discussed briefly below.

2 Summary of some of the articles

Dynamical percolation: Early results and open problems by Olle Haggstrom.

(Static) bond percolation theory imposes a probability measure on the edges of an infinite (but locally finite) connected graph, such that each edge is independently closed or open, and then looks at the properties of the resulting structure. Dynamical percolation extends this model to the time-varying case by defining a Markov process on the edges, so that each edge can now independently flip state at a given rate. The article looks at some dynamical percolation results for trees, cubic lattices, and more general graphs. Percolation can also be studied in a continuous space without being confined to a lattice or graph; in this vein, the author looks at a dynamical version of the continuum Poisson Boolean model.

Mixing times by Laszlo Lovasz and Peter Winkler.

A very interesting research paper on various ways of defining "mixing time", which is the number of steps a Markov chain is expected to take to reach its stationary distribution. (Aside from theoretical interest, this is a very practical issue in many simulation situations where a Markov process must be sampled.) Proofs are given for the equivalence up to constant factors of many of the different mixing time definitions.

A bird's-eye view of uniform spanning trees and forests by Russell Lyons.

A "uniform" spanning tree simply denotes choosing one such tree from the set of all spanning trees of a graph, with uniform probability distribution. An elegant algorithm for choosing such trees on a (connected finite) graph is to take a random walk on the graph and add only those edges which do not complete cycles. The idea can be extended in a natural way to infinite graphs, where one speaks of uniform spanning forests. The author examines the properties of such forests on several graph classes, particularly integer lattices, and discusses applications to electrical networks; several interesting open questions are then posed.

Coupling from the past: A user's guide by James Propp and David Wilson.

In the same vein as the previous article on mixing times, the authors describe a general method called "Coupling From the Past" which allows perfectly random samples from Markov chains; for the method to be applicable, some convergence conditions must be met. More specifically, in applicable cases the final output sample from the probability distribution is independent of the initial state of the chain, with expected running time a fixed multiple of the mixing time. One interesting point the authors make is that sampling schemes with random execution times are vulnerable to a subtle bias, due to long execution runs being prematurely terminated by impatient experimenters.

Annotated bibliography of perfectly random sampling with Markov chains by David B Wilson.

A useful survey of many different methods for sampling perfectly (or arbitrarily close to perfectly, depending on the algorithm) from Markov chains. Pointers to simulations are given, along with a glossary of related concepts.

3 Titles of the other articles

The following is a list of all the other articles in this volume.

1. *Tree-valued Markov chains and Poisson-Galton-Watson distributions* by David Aldous.
2. *On the central role of scale invariant Poisson processes on (0, ∞)* by Richard Arratia.
3. *Beyond the method of bounded differences* by Anant P Godbole and Pawel Hitczenko.
4. *Distinguishing and reconstructing sceneries from observations along random walk paths* by Harry Kesten.
5. *Enumerations of trees and forests related to branching processes and random walks* by Jim Pitman.
6. *Couplings for normal approximations with Stein's method* by Gesine Reinert.

4 Opinion

This book would benefit greatly from the addition of a Preface or Introduction chapter, with an overview of the importance and techniques of the area, links to other related areas, and a short explanation of the significance of each article. As it is, the book essentially reads like a collection of independent papers as opposed to a unified exposition; although this is understandable since the book consists of the proceedings from a DIMACS workshop, a "semantic overview layer" would add a lot of value. An Index would also be useful.

I would suggest the following three recent books as useful background or supplementary reading for those who are not discrete probabilists: 1) Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues. Pierre Bremaud; Springer-Verlag, 1999. 2) Percolation (2nd ed). Geoffrey Grimmett; Springer-Verlag, 1999. 3) Random Graphs. Janson et al; Wiley-Interscience, 2000.

Microsurveys is a worthwhile random sampling from recent developments in discrete probability. "Discrete Probability"is probably an indiscreetly broad phrase to use in the title, since the articles are clustered in the areas of Markov and Poisson processes and random graphs. Nevertheless, those who are interested in these areas and ready to provide their own orientation will find some interesting material here.

Review of *Term Rewriting and all that*
by Franz Baader and Tobias Nipkow
Published by Cambridge Univ. Press, 313 pages
Softcover, $27.95
ISBN number 0-521-779200

Reviewer: Paliath Narendran
State University of New York at Albany (SUNY-Albany)
Dept. of Computer Science

A term rewriting system is a collection of one-directional rewrite rules between (well-formed) terms over a signature. (A well-known example of a rewrite rule is the cancellation rule $x \circ x^{-1} \rightarrow 1$ for groups, over the signature $\{\circ, ^{-1}, 1\}$.) The area of term rewriting has seen considerable progress in the last two decades. Though there are several surveys of the field (e.g., [DerJou88]), until now no book was available in English on this topic. (There is a small monograph-size book by Jürgen Avenhaus [Aven95], published in 1995, but it is in German.) The book under review more than fills that void.

As the authors say in the introduction, this book is meant both as a textbook and as a reference for researchers in the field. The daunting task of writing such a book did not faze the authors who have "pulled it off" admirably. (It helps a great deal that the authors are themselves very active, researchers in the field.) The outcome is a cogent presentation of the important topics in a clear and at the same time rigorous way.

The first chapter, "Motivating examples," sets the stage by presenting situations (e.g. symbolic differentiation) where term rewriting is applicable. The second one lays the foundations rigorously by stating and proving properties of abstract reduction systems. Key concepts like *confluence,* and *termination* are introduced. Huet's elegant proof, using well-founded induction, that local confluence and confluence are equivalent for terminating systems is presented.

The third and fourth chapters introduce equational reasoning. Birkhoff's completeness proof, that semantic (model-theoretic) and syntactic notions of validity coincide for equational logic, is presented in detail in the third chapter. The fourth chapter deals with congruence closure (which is important when the axioms are over ground, or variable-free, terms) and syntactic unification.

Chapter 5, "Termination," is a crucial one (and a very good one too!) especially for me since I feel that this is one area where term rewriting has made a significant contribution to Theoretical Computer Science in general. Kruskal's theorem on tree embedding is proved following Nash-Williams. This important result is the underpinning for many an ordering that is used in automated reasoning systems (e.g., see [KZ95]).

Chapter 6 deals with the property of confluence (which roughly means that any two diverging sequences of rewrites can be made to converge). The idea of *completion,* where one adds enough new rules to make a non-confluent system confluent, is treated in the next chapter. Correctness of Huet's completion procedure is proved.

Chapter 8 is a delightful bonus for the reader. Buchberger's algorithm for computing *Gröbner bases* for polynomial ideals is presented. This is also a "completion procedure" in the above sense and crucial to computational algebraic geometry.

The remaining chapters deal with advanced topics. Chapter 10, "Equational Unification," is an especially good one, which is not very surprising since one of the authors (Baader) is an authority in this area. The chapter contains a detailed exposition of *associative-commutative unification* (or "AC-unification"), namely unification where some of the operators are associative and commutative. This is an important computational problem since associative and commutative operators (e.g., the logical $\wedge$ operator) turn up in many situations.

Two appendices — one covering background material on orders and the other on the language ML — and a detailed bibliography round out the book. The index seems carefully done and is very helpful. Congratulations to the authors on a job well done.

And now to nitpick: I feel the authors should have paid some attention to the (sub)field of string rewriting. References [228] and [250] in the book ([Sen96] and [ZG95] respectively) pertain to that area in any case. At the very least, a reference to the volume by Book and Otto [BoOt93] should have been there. Also, despite what the authors suggest in the Preface about prerequisites being minimal, I suspect that a course using this book can be handled only by students who have had a fairly sophisticated course on combinatorics, logic and set theory.

References

[Aven95] J. Avenhaus, *Reductionssysteme,* Springer, 1995.

[BoOt93] R.V. Book, and F. Otto, *String-Rewriting Systems*, Springer, New York, 1993.

[DerJou88] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[KZ95] D. Kapur and H. Zhang. An Overview of Rewrite Rule Laboratory (RRL), J. of Computer and Mathematics with Applications, 29, 2, 1995, 91-114.

[Sen96] G. Senizergues. On the termination problem for one-rule semi-Thue systems, In H. Ganzinger, editor, *Rewriting Techniques and Applications* LNCS 1103, pages 302–316. Springer-Verlag, 1996.

[ZG95] H. Zantema and A. Geser. A complete characterization of termination of $0^p1^q \to 1^r0^s$. In J. Hsiang, editor, *Rewriting Techniques and Applications* LNCS 914, pages 41–55. Springer-Verlag, 1995.