**The Book Review Column**[1]

by William Gasarch and Samir Khuller
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 three books are reviewed. This particular column has a guest co-editor Samir Khuller because one of the books reviewed is by the usual editor William Gasarch. Samir Khuller handled that review (e.g., decided if it was in scope, found a reviewer, and proofread the final review). William Gasarch will not see the review until he reads it in SIGACT NEWS.

1. **Computational geometry in C (Second Edition)** , by Joseph O'Rourke. Reviewed by: Michael Dekhtyar. This book is an introductory text in Computational Geometry. The algorithms are explained on a variety of levels including actual code.
2. **Bounded Queries in Recursion Theory** by William Gasarch and Georgia Martin. Reviewed by Lance Fortnow. This book looks at the complexity of a function in terms of how many queries to some oracle are needed to compute it. The context is recursion-theoretic.
3. **Logic For Applications (Second Edition)** , By Anil Nerode and Richard A. Shore. Reviewed by Alexander Dekhtyar. This book contains both logic and logic programming, and the connections between the two.

Review of[2]
**Computational geometry in C**
**Second Edition, 1998**
**Author: Joseph O'Rourke**
**Softcover $29.95, ISBN 0521649765**
**Hardcover $69.95, ISBN 0521640105**
**Publisher: Cambridge University Press**
**424 Pages**

Reviewer: Michael Dekhtyar
Department of Computer Science
Tver State University
Tver, 170000, Russia
Michael.Dekhtyar@tversu.ru

# 1 Overview

The book under review is the second edition of an introductory textbook in computational geometry. This area has seen intense development during the last two decades. Machine graphics, pattern recognition, robot motion are the main sources of the problems in this area. For many of these problems "simple" solutions can be found by brute force algorithms. But their efficient, practically implementable solutions need the deep insight into geometrical structures as well as the delicate

analysis of computational peculiarities. Throughout the eight chapters that constitute the main part of the book, the author leads the reader by a typical way from the informal or semi-formal specifications of a problem to its efficient solution by consecutively (a) formalizing the problem, (b) developing geometrical structures and establishing their properties, (c) constructing more or less straightforward algorithms, then (d) transforming them into efficient ones, (e) and finally implementing them in C. Here we list examples of typical problems considered in chapters 1-8:

1. How many stationary guards are needed to guard an art gallery which has a form of polygon?
2. Find a partition of a polygon into the (minimal number of) convex pieces.
3. Find the smallest area rectangle that encloses a polygon.
4. Find the smallest three-dimensional box surrounding an object in space.
5. Suppose one would like to locate a new grocery store in an area with several existing, competing grocery stores. Where should the new store be located to optimize its sales?
6. Given a set of flat, opaque, colored polygons in three-dimensional space, produce an image of their appearance from a particular viewpoint.
7. Given a partition of a workstation screen into polygons and coordinates of a point where the mouse is clicked, find the polygon which includes the point.
8. Suppose that an environment of impenetrable obstacles (polygons in two and polyhedra in three dimensions) is fixed. Robot $R$ is at some initial point $s$, and the task is to plan motions that will move it into some final position $t$. Find a shortest path for $R$ from $s$ to $t$.

Of course this list is not exhaustive and solutions of numerous other problems are presented as well. The second edition extends the first with about fifty pages, more than dozen new algorithms, fifty new exercises, and eighty bibliographic references. Also all the C code had been rewritten and many programs run faster.

The more detailed summary of the book is provided below.

## 2 Summary of Contents

The book consists of 9 chapters.

Chapter 1 **Polygon triangulation** begins with discussion of problem 1 from the list above. *The Art Gallery Theorem*, which says that $\lfloor n/3 \rfloor$ guards are sufficient and necessary to guard any polygonal gallery is proved. This problem naturally leads to the problem of dividing of a polygon into triangles. The almost "obvious" statement that every polygon of $n$ vertices may be partitioned into $n-2$ triangles using $n-3$ diagonals (theorem 1.2.3, lemma 1.2.4) leads to a brute force triangulation algorithm of complexity $O(n^4)$. At the end of the chapter a more efficient triangulation algorithm of complexity $O(n^2)$ is developed and implemented in C. It is preceded by solutions of some simple but important computational tasks: area of triangle, cross product, area of convex and nonconvex polygon, volume in three dimensions, and segments intersection. Procedures for these tasks are used later in the book.

In Chapter 2, **Polygon partitioning**, other types of polygon partitioning are considered. The first three: partitioning into "monotone" polygons, into trapezoids, and into "monotone mountains" allow to get partition algorithms of complexity $O(n \log n)$. In the construction of trapezoidalization,

a technique called a "plane sweep" (or "sweep line"), which is one of the main technical tools of computational geometry is introduced for the first time. First a table which shows history of triangulation is presented. Then the best algorithms are sketched: the linear time Chazelle's algorithm and the almost linear randomized Seidel's algorithm. At the end of the chapter, problem 2 from the list above is discussed. It is shown that the Hertel-Melhorn algorithm based on triangulation solves the problem in linear time and provides a solution which is never worse than four-times optimal in the numbers of convex pieces. There is no C-code in this chapter but the reader can produce it herself easily enough from the presented pseudo-code algorithms.

Chapter 3, **Convex hulls in two dimensions**, is devoted to the problem of constructing given a finite set $S$ of $n$ points on the plane, the smallest convex polygon $P$ that encloses $S$. This problem has a number of applications, and one of them is problem 3 from the list above. The chapter starts with three naive algorithms of complexity $O(n^2)$. Then Graham's algorithm of complexity $O(n \log n)$ is carefully analyzed and implemented in C. It is shown that the sorting problem is reducible quickly to the convex hull problem, so the bound $O(n \log n)$ is the best possible. Two other optimal algorithms are presented at the end of the chapter. First one constructs the convex hull incrementally: given a convex hull of $n$-points set $S$ and a new point $x$ it efficiently constructs the convex hull of $S \cup \{x\}$. Second algorithm exploits "divide and conquer" technique and shows how to merge quickly convex hulls of two sets into the convex hull of their union.

In chapter 4, **Convex hulls in three dimensions**, the same problem is considered for finite sets of points in 3-dimensional space. At first the famous Euler's formula is deduced which shows that for any polyhedron with $n$ vertices, both the number of its edges and faces is $O(n)$. The author outlines the theoretically best algorithm of complexity $O(n \log n)$ which uses divide and conquer technique. He discusses his own troubles with implementation of this algorithm. The core of the chapter is an implementation and analysis of the incremental algorithm for constructing convex hull in three dimensions which takes about 30 pages. Its theoretical complexity is $O(n^2)$, but proposed implementation is efficient enough. An impressive result of the algorithm is the image on the cover of the book which shows the convex hull of 5000 random points near the surface of a sphere. The problem of constructing convex hull in higher dimensions is discussed briefly at the end of the section.

Chapter 5, **Voronoi diagrams**, introduces two dual geometrical structures which have a number of applications. One of the applications is problem 5 from our list. Some other problems, which include minimum spanning tree and traveling salesperson problems, are discussed in Section 5.5.

Given set of points $P = \{p_1, ..., p_n\}$ on a plane, a *Voronoi diagram* $\mathcal{V}(\mathcal{P})$ is the set of all points that have more than one nearest neighbor in $P$. This set partitions the plane into *Voronoi regions (polygons)* $V(p_i), 1 \leq i \leq n$, of points with nearest neighbor $p_i$. *Delaunay triangulation* $\mathcal{D}(\mathcal{P})$ is the dual graph whose nodes are points of $P$ and whose edges connect points with adjoining Voronoi regions. $\mathcal{D}(\mathcal{P})$ is in fact triangulation of convex hull of $P$. As the book notes, there have been numerous efforts to establish efficient algorithms to construct Voronoi diagrams. The author overviews four of them and presents some details of Fortune's plane-sweep algorithm which has optimal complexity $O(n \log n)$. Another approach is based on the following interesting geometrical fact: if we transform any point $p_i = (x_i, y_i)$ of $P$ into 3-dimensional point $p'_i = (x_i, y_i, x_i^2 + y_i^2)$ on the paraboloid $z = x^2 + y^2$ and construct the lower convex hull of the set $\{p'_1, \ldots, p'_n\}$ then the projection of this convex hull to the $xy$-plane gives us precisely the Delaunay triangulation of $P$. It allows us to use convex hull algorithm of the previous chapter to get efficient algorithms which construct Delaunay triangulation and Voronoi diagram in $O(n^2)$ steps.

Chapter 6, **Arrangements**, introduces an important geometrical structure consisting of a set of lines "arranged" in the plane. Any such set divides the plane into cells, edges, and vertices. The

author lists a number of problems in which this (at the first glance too abstract) structure appears naturally. One of them is a *Hidden surface removal* problem included as problem 6 in the list above, another example is *Ham-sandwich cuts* problem: given two sets of points on the plane, find a line that simultaneously bisects both sets. An incremental algorithm is presented (in pseudo-code) which given representation of lines of an arrangement maps into some clear representation of its cells (edges, vertices). It works in $O(n^2)$ steps. Some applications of the algorithm are illustrated. It is shown also that arrangements are closely related to Voronoi diagrams.

Chapter 7, **Search and intersection**, shows how to answer two kinds of questions: (i) how to locate a point in some geometrical structure and (ii) how to find an intersection between different geometrical objects. It starts with two constant time algorithms: first one finds the point of intersection of two segments and the second one finds the point of intersection between segment and triangle in three dimensions. Then two locations problems are considered: point in polygon and point in polyhedron. For both problems linear time algorithms are presented based on counting the number of times a ray cast from a point crosses the boundary of a polygon or a polyhedron. It's clear that the point is in iff this number is odd. An implementation of this simple idea nevertheless requires some accuracy since there are some special-case intersections of the ray with the boundary which should be handled separately. Then the focus of the book shifts again to intersection problems: for convex polygons, for sets of segments, and for nonconvex polygons. For the first problem a simple linear time algorithm, which had been developed by the author and three of his undergraduate students, is chosen For the second problem, the Bently-Ottman sweepline algorithm is presented. This algorithm constructs the intersection of $n$ segments in the plane in $O((n+k)\log n)$ time, where $k$ is a number of intersection points. It is explained how to adapt the same algorithm to find the intersection of nonconvex polygons. Problems of constructing extremal points of polygons and polytops are closely related to intersection problems. It's not difficult to use binary search to obtain the extreme point of a polygon in $O(\log n)$ time. The same problem seems much harder for three dimensions but Kirkpatrick's algorithm presented in section 7.10 allows, after $O(n)$ preprocessing, to detect polytope extreme points in $O(\log n)$ time each. At the end of the chapter problem 7 of our list above, called the planar point location problem, is considered and some algorithms to solve it are outlined. The most interesting one uses randomized trapezoidal decomposition and after $O(n \log n)$ preprocessing it takes expected $O(\log n)$ time per query.

Chapter 8, **Motion planning**, is devoted to the problem of moving geometrical objects in a given environment. The main problem considered is that of the robot moving from one point to another (problem 7 on the list above). First, a simple case when the robot is a point and the environment is a collection of disjoint polygonal obstacles with a total $n$ vertices is considered. In this case it is possible to construct the visibility graph and then to apply the well-known Dijkstra's shortest path algorithm. So the problem can be solved in $O(n^2)$ time. When the robot is a disk, a convex polygon, or a line segment even the problem of finding any path between two locations is hard enough. For the first two cases algorithms based on construction of Minkowski sums of the robot and polygon obstacles are sketched (and supported by C code). For the third case of "moving ladder" two general techniques are considered: the cell decomposition method by Schwartz and Sharir and the retraction method by Ó'Dúnlain and Yap. Then another kind of motion planning problems is considered. It is planning the motion of an anchored "robot arm" consisting of fixed-length connected segments. An interesting *Two links theorem* (theorem 8.6.5) shows that the reachability problem for $n$-link arm can be simply reduced to the problem for 3-link arm. It allows to implement an algorithm which solves reachability problem for $n$-link arm in time $O(n)$ (in fact, after $O(n)$ preprocessing only $O(1)$ time is needed to check any point). The

last separability problem considered in the chapter is: given a collection of disjoint polygons on the plane, can each of them be moved to "infinity" without disturbing the others? The answer is always "yes" if all polygons are convex (even under some severe conditions such as all motions are translations in the same direction, each polygon is moved only once, and only one polygon is moved at a time). Such separation can be done in time $O(n \log n)$. In the case of nonconvex polygons the problem is much harder. It is shown that well-known $NP$-complete problem $PARTITION$ is reducible to it, so the separability problem is $NP$-hard. Then an example of the collection of $n$ polygons is constructed by the "Towers of Hanoi" puzzle of height $n$. It is shown that this collection can be separated, but the number of moves should be exponential.

Chapter 9, **Sources**, was absent in the first edition of the book. It should help the reader to get access to different sources of information in the young but rapidly developing area of computational geometry. The chapter includes the lists of bibliographies, textbooks, book collections, monographs, journals, conference proceedings, and links to Web sites with geometrical software.

At the end of the book, an extensive bibliography of more than 250 references is presented.

## 3 Style

It seems that presentation strategy used by the author is the best possible for an introductory textbook with the main goal of teaching the reader to solve problems in a new field. Solutions of many problems are traced completely from the specification of a problem to the C program which implements an efficient algorithm solving it. Development of geometrical structures and detailed proofs of their properties used by the algorithms are also discussed in detail. But for some problems other presentation styles are used: for some geometrical problems proofs of their properties are omitted, for some others the solutions are presented by pseudo-code algorithms only and their implementation is left to the reader, some algorithms are only sketched, and some only listed. (In all cases the omissions are accompanied by the references to the bibliography). It allows the author to combine both deep and complete exposition of the main techniques used in the area with the wide range of the problem in it, and stress the best known and most recent results, and applications. There is plenty of exercises throughout the book, which vary from easy programming tasks to "open" problems.

An important quality of the book is the fact that it self-contained. Almost all needed elementary geometry, calculus, and linear algebra is explained in the text. Only basic knowledge of C is required to understand the programs. Explanations of the algorithms and programs are clear and all long programs are divided into smaller blocks of understandable procedures. Presented programs are ready to be used and are available (both C and Java versions) through the author's Web cite: *http://www.cs.smith.edu/∼orourke* (unfortunately the prefix "www" is omitted in the address presented three times in the book).

A number of illustrations improve the geometrical intuition of the reader and help them to catch the subtle details of algorithms.

The author pays a lot of attention to the history of the young area of computational geometry. All theorems, lemmas, methods, algorithms, and even ideas are thoroughly attributed to their authors and dates.

## 4 Opinion

This is an excellent book! It is well written and can and should be used as the basis of undergraduate as well as graduate courses in computational geometry. It can be studied by and will be useful for beginners and can be an excellent reference for the readers more familiar with the area. The improved second edition can draw a new army of researches in computational geometry. In the last line of the book the author said:"Enjoy!". I did. And I believe that so would its every reader.

Review of **Bounded Queries in Recursion Theory**[3]
William A. Gasarch and Georgia A. Martin
Progress in Computer Science & Applied Logic, Volume 16
Birkhäuser. Boston, Basel, Berlin, 1999, xiii + 353 pp.
Hardcover, ISBN: 0817639667, $49.50
Reviewer: Lance Fortnow, University of Chicago, fortnow@cs.uchicago.edu

## 1 Overview

Consider a set $A$ of strings. Suppose you have a list of a thousand strings. You want to know how many of these strings are in $A$. You are allowed to make queries to $A$ but you want to limit the number of queries allowed. How many queries to $A$ are necessary to determine how many of the thousand strings are in $A$?

The answer depends considerably on $A$. Clearly one thousand queries to $A$ suffice and if $A$ is random this is the best we can do. If $A$ is the halting problem then ten queries suffice (though not if we make them in parallel). If $A$ is computable we need not make any queries at all. Perhaps surprisingly there are computably enumerable sets where all thousand queries are necessary, even if we just want to know whether an odd or even number of them are in the set.

The book by Gasarch and Martin is full of results like these: results that are easy to state and proofs that for the most part are easy to follow. For example consider the following question: Given three Turing machines $M$, $N$ and $O$ which of them will halt on blank tape. You are only allowed two queries to halting set.

First query: Do at least two of the machines halt?

If no: Second query: Does at least one of the machines halt? If no we know that none of them halt. If so then simulate the machines in parallel until the one that halts halts.

If the answers to the first query is yes: Second query: Do at least three halt? If so we know they all halt. Otherwise simulate them in parallel until two of them halt.

Most of the results described in the book have more involved proofs but still rather easy to follow. There lies the beauty of the book.

## 2 Summary of Contents

Chapter one covers basic definitions and background in computability theory.

Chapter two covers some simple examples in bounded queries.

Chapter three covers most of the definitions used later in the book with some simple relationships between them. I found this chapter hard to read through–it might be better just to refer to the definitions as needed.

[3] 

Chapters four through nine cover the bulk of the results. The authors divide these chapters into groups entitled functions (chapters 4-5), sets (6-8) and miscellaneous (9).

Chapter four considers questions related to determining which of $n$ given strings are in $A$. This leads to the interesting notions of superterse and verbose.

Chapter five looks at counting how many of a given list of strings is in a specified set $A$. Chapter six looks at determining whether this number is odd or a multiple of some fixed $m$. A notably part of this chapter is three seemingly different proofs that for any computably enumerable noncomputable set $A$, determining whether the number of strings is odd cannot be done with $n-1$ nonadaptive queries to $A$.

Chapter seven focuses on convergence. It is possible that a reduction using queries to $A$ might not halt if the queries were not answered correctly. This chapter considers the effect of requiring the machines to halt no matter how the queries are answered.

Chapter eight looks at the relationship between classes of sets reducible to $n$ versus $n+1$ queries to a given set $A$. Chapter nine considers nondeterministic machines making the queries.

Chapter ten gives a very nicely annotated bibliography of the major papers in the area. For each paper a page is given describing the results defining appropriate notation as necessary. Looking through them one must note that a small group of researchers (Beigel, Gasarch, Kummer and Stephan) dominate most of the papers in the area.

Notably missing are results on bounded queries in resource-bounded models. There is a considerable amount of research here done by many of the same researchers often using the same techniques and yet it is completely ignored in this book.

## 3 Style

Good: The book requires some computability theory but not much more than is covered in an undergraduate theory class. Very few proofs require even as much as a priority argument or finite injury. When they do, such as the thousand queries is necessary result described in the overview, the authors go out of the way to give details making the arguments easy to follow. Overall the results are well presented and the proofs fully given.

Bad: The book is notation heavy. I found it quite difficult to keep in my head the meaning of the various symbols and terminology, much of which is unique to bounded queries. The good index helped find the definitions I needed, but understanding most proof required too much page turning.

The book uses the "recursion theory" terminology instead of the more intuitive "computability theory" terminology that has recently been gaining wide acceptance in the community.

## 4 Opinion

The book is ideal for an advanced undergraduate or beginning graduate student who has some exposure to basic computability theory and wants to see what one can do with it. The questions asked are interesting and can be easily understood and the proofs one can follow without a large amount of training in computability theory. I cannot think of another book which serves such a purpose.

It also makes a good reference for those interested in bounded queries though it is sometimes difficult to locate a specific result. The book does not give much in the way of applications for other areas of computability theory.

The book would not replace textbooks (such as Soare or Odifreddi) as graduate introduction to computability theory. Only techniques and result related to bounded queries are really discussed in this book. The book does not give the necessary background needed to do heavy duty research in computability theory.

Review of
**Logic For Applications**[4]
**Second Edition, 1997**
**Authors: Anil Nerode, Richard A. Shore**
**Publisher: Springer Verlag**
**Hardcover $49.95**
**ISBN 0387948937**
**456 Pages**

Reviewer: Alexander Dekhtyar
Department of Computer Science
University of Maryland
at College Park
dekhtyar@cs.umd.edu

# 1 Overview

Among the books on foundations of mathematics and theoretical computer science, textbooks on logic and logic programming are usually quite distinct. The former introduce propositional and predicate logic, study their properties and prove the soundness and completeness results together with some more advanced results, such as the NP-completeness of the satisfiability problem for propositional case and undecidability of this problem for first order logic. The proof systems discussed in these books are usually either Hilbert- or Gentzen-style. The latter category of the textbooks usually start with the introduction of the Horn fragment of propositional (and first order) logics. Herbrand models are described as the semantics and SLD-resolution – as the proof procedure for this fragment.

There is also a third category of textbooks, those devoted to non-classical logics, such as modal, intuitionistic, temporal or linear.

*"Logic for Applications"*, the book under review is a rare kind of a textbook which tries to combine together classical and non-classical logics and logic programming. This in fact was one of the two main reasons for writing the book as stated by the authors. The other reason (closely connected with the first one) was to provide a flexible textbook which could be used for teaching a course with a stress on logic programming as well as a pure introduction into classical mathematical logic or an introduction in non-classical logics.

The book starts with the description of propositional and first-order predicate logics, providing all the regular results (soundness, completeness, compactness) and introducing the concepts which are usually associated with logic programming (such as Herbrand models and resolution). Then a more formal and detailed description of logic programming and PROLOG follows. Two parts of the book that follow are devoted to two non-classical logical frameworks: modal logic and intuitionistic logic, with the remainder of the book occupied by a chapter devoted to the introduction into axiomatic set theory and a historical overview of logic starting from Ancient Greece and ending at present times.

[4]

## 2 Summary of Contents

The book consists of six chapters and two appendices, and also features a large bibliography at the end, broken according to the contents of the chapters.

Chapter I, *Propositional Logic* and Chapter II, *Predicate Logic* form the first part of the book, devoted to classical logic. As usual, the syntax of propositional and first-order predicate logics is described in the beginning of each chapter, followed by the description of their semantics and proof procedures. As a "standard" proof procedure, the authors use the Tableau proofs, which in their opinion are more intuitive than the Hilbert-style proof systems. The latter, however are also introduced, but only as optional material. The third proof procedure introduced is resolution. For each of the procedures soundness, completeness and compactness results are stated, but only for Tableaux and resolution the proofs are provided.

At the end of Chapter I, PROLOG, the logic programming language is introduced briefly, together with the underlying concepts of Horn fragment of propositional logic and SLD-resolution, with a detailed description deferred until Chapter III. Chapter II, together with soundness, completeness and compactness theorems discusses

- Skolem-Löwenheim theorem (existence of countable model for any satisfiable set of sentences);
- Prenex normal form and Skolemization theorem (existence, for any sentence of first order logic of a universal sentence in syntax extended with new function symbols which is equisatisfiable with it);
- Herbrand's Theorem (existence of an Herbrand model for any satisfiable set of sentences).

Creating a "bridge" toward the logic programming content that follows the classical logic chapters, Chapter II also introduces the concept of unification of terms.

Chapter III, *PROLOG* is devoted to a more detailed study of logic programming. While the logic programming related content of the first two chapters (Horn clauses, logic programs, resolution, SLD-resolution) concentrated on theoretical issues, some more practical issues, related to PROLOG are being discussed in detail here. Such issues involve, for example, backtracking in PROLOG, the order of clauses in a PROLOG program and the use of ! (Cut) to control the backtracking. Some attention is also devoted to the relatively recent results in termination of PROLOG programs (the general problem being undecidable, the question is for which reasonable subclasses one can decide this problem). Last but not least, the authors discuss two different approaches to handling negations in logic programming: traditional *negation as failure* approach and a more recent *stable semantics* of logic programs.

Chapters IV, V, *Modal Logic* and *Intuitionistic Logic* constitute the part of the book devoted to non-classical logics. Both chapters are presented in a fashion that makes them independent from each other in an attempt to provide an instructor who might use the book as a textbook more flexibility in the choice of the content to be taught in class.

Chapter IV follows the general structure of Chapters I, II to introduce modal logic. The authors introduce the modalities of necessity and possibility into the language of propositional logic and then define the semantics of the new logic via Kripke frames (structures). A Tableau proof procedure for modal case is introduced and its soundness and completeness is established. The authors then go on to consider the hierarchy of modal logics (K, T, S4, S5 systems), by introducing the appropriate axioms and showing which class of models satisfies them.

Due to the author's desire to make Chapter V independent from Chapter IV, it also start with the description of Kripke frames, this time complete with a $\leq$ relation. The authors then introduce

the semantics of intuitionistic logic on such frames and prove that all intuitionistically valid sentences are also valid in classical logic. Then, they provide a variety of examples of classically valid sentences, which are not valid in intuitionistic logic. An intuitionistic Tableau proof procedure is then introduced and soundness and completeness results are established. Then, the decidability and undecidability issues for the intuitionistic logic are discussed, including the finite model property theorem (every quantifier free sentence is forced in all model iff it is forced in all finite models), which leads to the decidability of quantifier-free intuitionistic logic and prenex fragment of intuitionistic predicate logic. The undecidability of first order intuitionistic logic is shown by translation of classical logic sentences into intuitionistic ones, the latter leading to an interesting conclusion that in intuitionistic logic not every sentence is equivalent to a sentence in a prenex form. At the end of the chapter, different aspects of intuitionistic and modal logics are compared and contrasted.

Chapter VI, *Set Theory* completes the book by offering a formal introduction into set theory including such topics as Axiom of Choice, set cardinality, ordinals and ordinal arithmetics and transfinite induction.

Appendix A of the book contains a rather comprehensive overview of the history of logic and foundations of mathematics, from Ancient Greece to Renaissance to present time.

## 3 Style

It can be concluded that the authors succeeded in their goal to provide a flexible, almost all-encompassing introduction into mathematical logic. The design of the book is quite elaborate, with sections intended for use in a logic programming oriented course preceded and followed by sections intended for a pure logic oriented course. However, the presentation of the material is always clear with multiple examples always illustrating and clarifying the key ideas and concepts.

The four logical systems introduced in the book: classical propositional and predicate logics, modal logic and intuitionistic logic are all described in a similar manner, first by introducing the language, following semantics, tableau proof procedure, soundness and completeness results and decidability results, and with an optional section on axiomatization. This structure makes the study of the latter, non-classical systems easier as it is relatively easy to compare and contrast them with classical logic.

We should also point out that the author's choice of Tableaux as the proof procedures for all their logics is somewhat unusual for a textbook in logic (most classical logic textbooks use Hilbert-style proof system, with yet some other, with a slant towards proof theory using Gentzen-style proof systems). The authors explain their choice by pointing out that creating a tableau that proves a sentence is a much more deterministic process than trying to prove the same sentence in a Hilbert-style proof system. This makes sense as the traditional Hilbert-style proof system for propositional logic (which has three axiom schemas and one inference rule schema and which allows sentences to contain only negation and implication connectives), while convenient for the soundness and completeness results proofs, makes search for the derivations rather hard.

The chapter devoted to PROLOG and applied logic programming strikes a good balance between the theory of logic programming and the realities of writing PROLOG programs, and spends considerable amount of space discussing the differences between PROLOG interpreter and the formal semantics. The discussion of negation in logic programming, despite its brevity is clear and is accompanied by numerous examples, which clearly illustrate the similarity and differences between negation as failure and stable semantics approaches.

The inclusion of the chapter on Set Theory and the overview of the history of foundations of

mathematics makes the book self-contained to a high degree (most of the non-logical concepts used in the book come either from a brief introduction on trees and other structures in the beginning of Chapter I or from the set theory). Putting the introduction into set theory in the book also added to its flexibility, as this chapter really can be used not only as a reference but as the material for the lectures.

As any textbook should, this book contains plenty of exercises, ranging from relatively simple ones, which test the understanding of key concepts studied in the course to rather elaborate and sophisticated ones, such as proofs of some important theorems.

# 4 Opinion

In our opinion the book achieves its goal of being a unified introduction into classical logic, logic programming and certain non-classical logics. While a broad scope of the book means that some more deep results had to be left out, the book succeeded in presenting a uniform framework for describing different logics. The author's thorough approach to describing logic programming, via introduction of resolution-based refutations and subsequent study of different kinds of resolutions allows the reader to gradually switch from the study of logic to the study of logical programming paradigm and provides a lot of intuition about the behavior of logic programs.

As such the book can be recommended both as a textbook for senior/graduate course in logic/logic programming, and as a reading or reference for graduate students in the areas related to discrete mathematics. The only possible disadvantage this book has compared to other mathematical logic textbooks is the choice of a primary proof procedure, which, while better than Hilbert (or even Gentzen) -style calculi for teaching purposes, is not as popular as the latter two in the literature.

This, however, is more than compensated by the book's clarity, efficient organization and self-containment.