TURING 图灵原版计算机科学系列

ELSEVIER 爱思唯尔

Computability, Complexity, and Languages

Fundamentals of Theoretical Computer Science

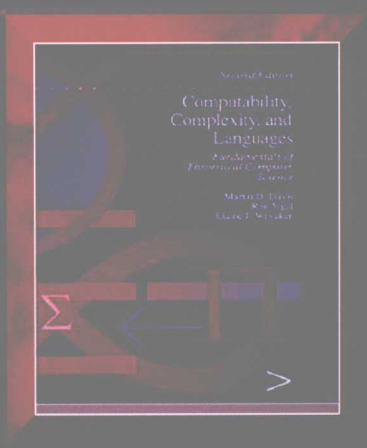
Second Edition

计算理论基础

可计算性、复杂性和语言

（英文版·第2版）

Martin D. Davis
[美] Ron Sigal 著
Elaine J. Weyuker



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵原版计算机科学系列

Computability, Complexity, and Languages

Fundamentals of Theoretical Computer Science

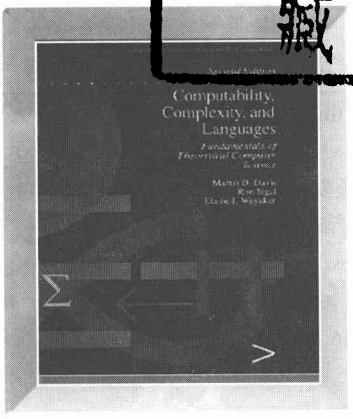
Second Edition

计算理论基础

可计算性、复杂性和语言

（英文版·第2版）

[美] Martin D. Davis / Ron Sigal / Elaine J. Weyuker 著



人民邮电出版社
北京

图书在版编目（CIP）数据

计算理论基础：可计算性、复杂性和语言＝Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science：第2版：英文/（美）戴维斯（Davis, M. D.），（美）西加尔（Sigal, R.），（美）韦约克（Weyuker, E. j.）著. —北京：人民邮电出版社，2009.5
（图灵原版计算机科学系列）
ISBN 978-7-115-19657-6

I. 计… II. ①戴… ②西… ③韦… III. 计算技术－理论－英文 IV. TP301

中国版本图书馆CIP数据核字（2009）第006831号

内容提要

本书是理论计算机科学领域的名作，是计算机科学核心主题的导论性教材。全书分为可计算性、文法与自动机、逻辑学、复杂性及语义学5个部分，分别讲述了可计算性理论、形式语言、逻辑学与自动演绎、可计算复杂性（包括NP完全问题）和编程语言的语义等主题，并展示了它们之间如何相互关联。

本书是计算机及相关专业高年级本科生和研究生的理想教学参考书，对于计算机领域的专业人士也是很好的技术参考书。

图灵原版计算机科学系列

计算理论基础：可计算性、复杂性和语言（英文版·第2版）

◆ 著 [美] Martin D. Davis Ron Sigal Elaine J. Weyuker
责任编辑 杨海玲
◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京顺义振华印刷厂印刷
◆ 开本：800×1000 1/16
印张：39.25
字数：754千字 2009年5月第1版
印数：1-2 000册 2009年5月北京第1次印刷
著作权合同登记号 图字：01-2008-5836号
ISBN 978-7-115-19657-6/TP

定价：79.00元

读者服务热线：（010）88593802 印装质量热线：（010）67129223
反盗版热线：（010）67171154

版权声明

ISBN: 978-981-272-313-0

Elsevier (Singapore) Pte Ltd.
3 Killiney Road
08-01 Winsland House I
Singapore 239519
Tel: (65)6349-0200
Fax: (65)6733-1817

First Published 2009
2009年初版

To the memory of Helen and Harry Davis
and to
Hannah and Herman Sigal
Sylvia and Marx Weyuker

Virginia Davis, Dana Latch, Thomas Ostrand
and to
Rachel Weyuker Ostrand

Preface

Theoretical computer science is the mathematical study of models of computation. As such, it originated in the 1930s, well before the existence of modern computers, in the work of the logicians Church, Gödel, Kleene, Post, and Turing. This early work has had a profound influence on the practical and theoretical development of computer science. Not only has the Turing machine model proved basic for theory, but the work of these pioneers presaged many aspects of computational practice that are now commonplace and whose intellectual antecedents are typically unknown to users. Included among these are the existence in principle of all-purpose (or universal) digital computers, the concept of a program as a list of instructions in a formal language, the possibility of interpretive programs, the duality between software and hardware, and the representation of languages by formal structures, based on productions. While the spotlight in computer science has tended to fall on the truly breathtaking technological advances that have been taking place, important work in the foundations of the subject has continued as well. It is our purpose in writing this book to provide an introduction to the various aspects of theoretical computer science for undergraduate and graduate students that is sufficiently comprehensive that the professional literature of treatises and research papers will become accessible to our readers.

We are dealing with a very young field that is still finding itself. Computer scientists have by no means been unanimous in judging which

parts of the subject will turn out to have enduring significance. In this situation, fraught with peril for authors, we have attempted to select topics that have already achieved a polished classic form, and that we believe will play an important role in future research.

In this second edition, we have included new material on the subject of programming language semantics, which we believe to be established as an important topic in theoretical computer science. Some of the material on computability theory that had been scattered in the first edition has been brought together, and a few topics that were deemed to be of only peripheral interest to our intended audience have been eliminated. Numerous exercises have also been added. We were particularly pleased to be able to include the answer to a question that had to be listed as open in the first edition. Namely, we present Neil Immerman's surprisingly straightforward proof of the fact that the class of languages accepted by linear bounded automata is closed under complementation.

We have assumed that many of our readers will have had little experience with mathematical proof, but that almost all of them have had substantial programming experience. Thus the first chapter contains an introduction to the use of proofs in mathematics in addition to the usual explanation of terminology and notation. We then proceed to take advantage of the reader's background by developing computability theory in the context of an extremely simple abstract programming language. By systematic use of a macro expansion technique, the surprising power of the language is demonstrated. This culminates in a universal program, which is written in all detail on a single page. By a series of simulations, we then obtain the equivalence of various different formulations of computability, including Turing's. Our point of view with respect to these simulations is that it should not be the reader's responsibility, at this stage, to fill in the details of vaguely sketched arguments, but rather that it is our responsibility as authors to arrange matters so that the simulations can be exhibited simply, clearly, and completely.

This material, in various preliminary forms, has been used with undergraduate and graduate students at New York University, Brooklyn College, The Scuola Matematica Interuniversitaria–Perugia, The University of California–Berkeley, The University of California–Santa Barbara, Worcester Polytechnic Institute, and Yale University.

Although it has been our practice to cover the material from the second part of the book on formal languages after the first part, the chapters on regular and on context-free languages can be read immediately after Chapter 1. The Chomsky–Schützenberger representation theorem for context-free languages in used to develop their relation to pushdown automata in a way that we believe is clarifying. Part 3 is an exposition of the aspects of logic that we think are important for computer science and can

also be read immediately following Chapter 1. Each of the chapters of Part 4 introduces an important theory of computational complexity, concluding with the theory of NP-completeness. Part 5, which is new to the second edition, uses recursion equations to expand upon the notion of computability developed in Part 1, with an emphasis on the techniques of formal semantics, both denotational and operational. Rooted in the early work of Gödel, Herbrand, Kleene, and others, Part 5 introduces ideas from the modern fields of functional programming languages, denotational semantics, and term rewriting systems.

Because many of the chapters are independent of one another, this book can be used in various ways. There is more than enough material for a full-year course at the graduate level on *theory of computation.* We have used the unstarred sections of Chapters 1–6 and Chapter 9 in a successful one-semester junior-level course, Introduction to Theory of Computation, at New York University. A course on *finite automata and formal languages* could be based on Chapters 1, 9, and 10. A semester or quarter course on *logic for computer scientists* could be based on selections from Parts 1 and 3. Part 5 could be used for a third semester on the theory of computation or an introduction to *programming language semantics.* Many other arrangements and courses are possible, as should be apparent from the dependency graph, which follows the Acknowledgments. It is our hope, however, that this book will help readers to see theoretical computer science not as a fragmented list of discrete topics, but rather as a unified subject drawing on powerful mathematical methods and on intuitions derived from experience with computing technology to give valuable insights into a vital new area of human knowledge.

Note to the Reader

Many readers will wish to begin with Chapter 2, using the material of Chapter 1 for reference as required. Readers who enjoy skipping around will find the *dependency graph* useful.

Sections marked with an asterisk (*) may be skipped without loss of continuity. The relationship of these sections to later material is given in the dependency graph.

Exercises marked with an asterisk either introduce new material, refer to earlier material in ways not indicated in the dependency graph, or simply are considered more difficult than unmarked exercises.

A reference to Theorem 8.1 is to Theorem 8.1 of the chapter in which the reference is made. When a reference is to a theorem in another chapter, the chapter is specified. The same system is used in referring to numbered formulas and to exercises.

Acknowledgments

It is a pleasure to acknowledge the help we have received. Charlene Herring, Debbie Herring, Barry Jacobs, and Joseph Miller made their student classroom notes available to us. James Cox, Keith Harrow, Steve Henkind, Karen Lemone, Colm O'Dunlaing, and James Robinett provided helpful comments and corrections. Stewart Weiss was kind enough to redraw one of the figures. Thomas Ostrand, Norman Shulman, Louis Salkind, Ron Sigal, Patricia Teller, and Elia Weixelbaum were particularly generous with their time, and devoted many hours to helping us. We are especially grateful to them.

Acknowledgments to Corrected Printing

We have taken this opportunity to correct a number of errors. We are grateful to the readers who have called our attention to errors and who have suggested corrections. The following have been particularly helpful: Alissa Bernholc, Domenico Cantone, John R. Cowles, Herbert Enderton, Phyllis Frankl, Fred Green, Warren Hirsch, J. D. Monk, Steve Rozen, and Stewart Weiss.

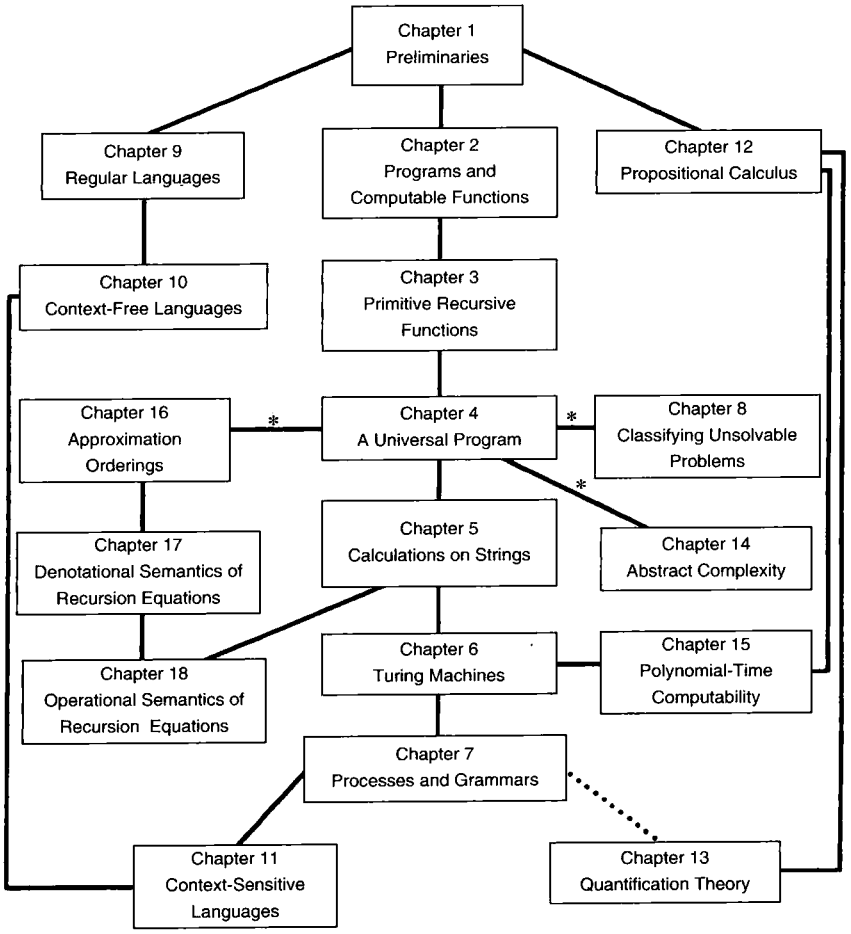
Acknowledgments to Second Edition

Yuri Gurevich, Paliath Narendran, Robert Paige, Carl Smith, and particularly Robert McNaughton made numerous suggestions for improving the first edition. Kung Chen, William Hurwood, Dana Latch, Sidd Puri, Benjamin Russell, Jason Smith, Jean Toal, and Niping Wu read a preliminary version of Part 5.

Acknowledgments to Reprint of Second Edition

We are grateful to the following people for their careful reading of the Second Edition: John Case, P. Klingsberg, Ken Klein, Eugenio Omodeo, David Schedler, John David Stone, and Lenore Zuck.

Dependency Graph



A solid line between two chapters indicates the dependence of the unstarred sections of the higher numbered chapter on the unstarred sections of the lower numbered chapter. An asterisk next to a solid line indicates that knowledge of the starred sections of the lower numbered chapter is also assumed. A dotted line shows that knowledge of the unstarred sections of the lower numbered chapter is assumed for the starred sections of the higher numbered chapter.

Contents

1

Preliminaries

1. Sets and *n*-tuples

We shall often be dealing with *sets* of objects of some definite kind. Thinking of a collection of entities as a *set* simply amounts to a decision to regard the whole collection as a single object. We shall use the word *class* as synonymous with *set*. In particular we write N for the set of *natural numbers* $0, 1, 2, 3, \ldots$. In this book the word *number* will always mean *natural number* except in contexts where the contrary is explicitly stated.

We write

$$a \in S$$

to mean that a belongs to S or, equivalently, is a member of the set S, and

$$a \notin S$$

to mean that a does not belong to S. It is useful to speak of the *empty set*, written $\varnothing$, which has no members. The equation $R = S$, where R and S are sets, means that R and S are *identical as sets*, that is, that they have exactly the same members. We write $R \subseteq S$ and speak of R as a *subset* of S to mean that every element of R is also an element of S. Thus, $R = S$ if and only if $R \subseteq S$ and $S \subseteq R$. Note also that for any set R, $\varnothing \subseteq R$ and $R \subseteq R$. We write $R \subset S$ to indicate that $R \subseteq S$ but $R \neq S$. In this case R