

Undergraduate Topics in Computer Science

Undergraduate Topics in Computer Science' (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

Also in this series

Iain D. Craig
Object-Oriented Programming Languages: Interpretation
978-1-84628-773-2

Max Bramer
Principles of Data Mining
978-1-84628-765-7

Hanne Riis Nielson and Flemming Nielson
Semantics with Applications: An Appetizer
978-1-84628-691-9

Michael Kifer and Scott A. Smolka
Introduction to Operating System Design and Implementation: The OSP 2 Approach
978-1-84628-842-5

Phil Brooke and Richard Paige
Practical Distributed Processing
978-1-84628-840-1

Frank Klawonn
Computer Graphics with Java
978-1-84628-847-0

David Salomon
A Concise Introduction to Data Compression
978-1-84800-071-1

David Makinson
Sets, Logic and Maths for Computing
978-1-84628-844-9

Alan P. Parkes

A Concise Introduction to Languages and Machines

 Springer

Alan P. Parkes, PhD
Honorary Senior Lecturer, Computing Dept., Lancaster University, UK

Series editor
Ian Mackie, école Polytechnique, France and University of Sussex, UK

Advisory board
Samson Abramsky, University of Oxford, UK
Chris Hankin, Imperial College London, UK
Dexter Kozen, Cornell University, USA
Andrew Pitts, University of Cambridge, UK
Hanne Riis Nielson, Technical University of Denmark, Denmark
Steven Skiena, Stony Brook University, USA
Iain Stewart, University of Durham, UK
David Zhang, The Hong Kong Polytechnic University, Hong Kong

Undergraduate Topics in Computer Science ISSN 1863-7310
ISBN 978-1-84800-120-6 e-ISBN 978-1-84800-121-3
DOI 10.1007/978-1-84800-121-3

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 0000000

© Springer-Verlag London Limited 2008

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licences issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

The use of registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

Springer Science+Business Media
springer.com

Contents

1. Introduction	1
1.1 Overview	1
1.2 What this Book is About	1
1.3 What this Book Tries to Do	6
1.4 What this Book Tries Not to Do	6
1.5 The Exercises	7
1.6 Further Reading	7
1.7 Some Advice	7
 Part 1 Language and Machines	
2. Elements of Formal Languages	11
2.1 Overview	11
2.2 Alphabets	11
2.3 Strings	12
2.3.1 Functions that Apply to Strings	12
2.3.2 Useful Notation for Describing Strings	13
2.4 Formal Languages	15
2.5 Methods for Defining Formal Languages	16
2.5.1 Set Definitions of Languages	17
2.5.2 Decision Programs for Languages	19
2.5.3 Rules for Generating Languages	20
2.6 Formal Grammars	25
2.6.1 Grammars, Derivations and Languages	26
2.6.2 The Relationship between Grammars and Languages	29

2.7	Phrase Structure Grammars and the Chomsky Hierarchy	30
2.7.1	Formal Definition of Phrase Structure Grammars	30
2.7.2	Derivations, Sentential Forms, Sentences and “ $L(G)$ ”	31
2.7.3	The Chomsky Hierarchy	35
2.8	A Type 0 Grammar: Computation as Symbol Manipulation	38
3.	Syntax, Semantics and Ambiguity	43
3.1	Overview	43
3.2	Syntax vs. Semantics	43
3.3	Derivation Trees	44
3.4	Parsing	47
3.5	Ambiguity	49
4.	Regular Languages and Finite State Recognisers	55
4.1	Overview	55
4.2	Regular Grammars	55
4.3	Some Problems with Grammars	57
4.4	Finite State Recognisers and Finite State Generators	58
4.4.1	Creating an FSR	58
4.4.2	The Behaviour of the FSR	60
4.4.3	The FSR as Equivalent to the Regular Grammar	64
4.5	Non-determinism in Finite State Recognisers	67
4.5.1	Constructing Deterministic FSRs	69
4.5.2	The Deterministic FSR as Equivalent to the Non-deterministic FSR	72
4.6	A Simple Deterministic Decision Program	77
4.7	Minimal FSRs	77
4.7.1	Constructing a Minimal FSR	79
4.7.2	Why Minimisation Works	83
4.8	The General Equivalence of Regular Languages and FSRs	86
4.9	Observations on Regular Grammars and Languages	88
5.	Context Free Languages and Pushdown Recognisers	93
5.1	Overview	93
5.2	Context Free Grammars and Context Free Languages	94
5.3	Changing G without Changing $L(G)$	94
5.3.1	The Empty String (ε)	95
5.3.2	Chomsky Normal Form	98
5.4	Pushdown Recognisers	104
5.4.1	The Stack	105
5.4.2	Constructing a Non-deterministic PDR	105
5.4.3	Example NPDRs, M_3 and M_{10}	107

5.5	Deterministic Pushdown Recognisers	112
5.5.1	M_3^d , a Deterministic Version of M_3	113
5.5.2	More Deterministic PDRs	115
5.6	Deterministic and Non-deterministic Context Free Languages.	116
5.6.1	Every Regular Language is a Deterministic CFL	116
5.6.2	The Non-deterministic CFLs.	118
5.6.3	A Refinement to the Chomsky Hierarchy in the Case of CFLs	120
5.7	The Equivalence of CFGs and PDRs.	121
5.8	Observations on Context Free Grammars and Languages	121
6.	Important Features of Regular and Context Free Languages.	125
6.1	Overview.	125
6.2	Closure Properties of Languages	126
6.3	Closure Properties of the Regular Languages	126
6.3.1	Complement	126
6.3.2	Union	128
6.3.3	Intersection	129
6.3.4	Concatenation	131
6.4	Closure Properties of the Context Free Languages	133
6.4.1	Union	133
6.4.2	Concatenation	135
6.4.3	Intersection	136
6.4.4	Complement	137
6.5	Chomsky's Hierarchy is Indeed a Proper Hierarchy	138
6.5.1	The "Repeat State Theorem"	139
6.5.2	A Language that is Context Free but not Regular	142
6.5.3	The "uvwxy" Theorem for Context Free Languages	143
6.5.4	$\{a^i b^i c^i : i \geq 1\}$ is not Context Free	150
6.5.5	The "Multiplication Language" is not Context Free	151
6.6	Preliminary Observations on the Scope of the Chomsky Hierarchy	153
7.	Phrase Structure Languages and Turing Machines.	155
7.1	Overview.	155
7.2	The Architecture of the Turing Machine.	155
7.2.1	"Tapes" and the "Read/Write Head"	156
7.2.2	Blank Squares	157
7.2.3	TM "Instructions"	158
7.2.4	Turing Machines Defined	159

7.3	The Behaviour of a TM	159
7.4	Turing Machines as Language Recognisers	163
7.4.1	Regular Languages	163
7.4.2	Context Free Languages	164
7.4.3	Turing Machines are More Powerful than PDRs	166
7.5	Introduction to (Turing Machine) Computable Languages	169
7.6	The TM as the Recogniser for the Context Sensitive Languages	170
7.6.1	Constructing a Non-deterministic TM for Reduction Parsing of a Context Sensitive Language	171
7.6.2	The Generality of the Construction	175
7.7	The TM as the Recogniser for the Type 0 Languages	177
7.7.1	Amending the Reduction Parsing TM to Deal with Type 0 Productions	178
7.7.2	Dealing with the Empty String	178
7.7.3	The TM as the Recogniser for all Types in the Chomsky Hierarchy	181
7.8	Decidability: A Preliminary Discussion	181
7.8.1	Deciding a Language	181
7.8.2	Accepting a Language	183
7.9	End of Part One	184

Part 2 Machines and Computation

8.	Finite State Transducers	189
8.1	Overview	189
8.2	Finite State Transducers	189
8.3	Finite State Transducers and Language Recognition	190
8.4	Finite State Transducers and Memory	191
8.5	Finite State Transducers and Computation	194
8.5.1	Simple Multiplication	194
8.5.2	Addition and Subtraction	195
8.5.3	Simple Division and Modular Arithmetic	199
8.6	The Limitations of the Finite State Transducer	200
8.6.1	Restricted FST Multiplication	201
8.6.2	FSTs and Unlimited Multiplication	204
8.7	FSTs as Unsuitable Models for Real Computers	204
9.	Turing Machines as Computers	209
9.1	Overview	209
9.2	Turing Machines and Computation	209

9.3	Turing Machines and Arbitrary Binary Multiplication	210
9.3.1	Some Basic TM Operations	210
9.3.2	The “ADD” TM	212
9.3.3	The “MULT” TM	215
9.4	Turing Machines and Arbitrary Integer Division	222
9.4.1	The “SUBTRACT” TM	222
9.4.2	The “DIV” TM	225
9.5	Logical Operations	226
9.6	TMs and the Simulation of Computer Operations	229
10.	Turing’s Thesis and the Universality of the Turing Machine	237
10.1	Overview	237
10.2	Turing’s Thesis	238
10.3	Coding a Turing Machine and its Tape as a Binary Number	240
10.3.1	Coding Any TM	241
10.3.2	Coding the Tape	244
10.4	The Universal Turing Machine	244
10.4.1	<i>UTM</i> ’s Tapes	245
10.4.2	The Operation of <i>UTM</i>	246
10.4.3	Some Implications of <i>UTM</i>	249
10.5	Non-deterministic TMs	249
10.6	Converting a Non-deterministic TM into a 4-tape Deterministic TM	250
10.6.1	The Four Tapes of the Deterministic Machine, <i>D</i>	251
10.6.2	The Systematic Generation of the Strings of Quintuple Labels	253
10.6.3	The Operation of <i>D</i>	257
10.6.4	The Equivalence of Non-deterministic TMs and 4-Tape Deterministic TMs	260
10.7	Converting a Multi-tape TM into a Single-tape TM	261
10.7.1	Example: Representing Three Tapes as One.	263
10.7.2	The Operation of the Single-tape Machine, <i>S</i>	265
10.7.3	The Equivalence of Deterministic Multi-tape TMs and Deterministic Single-tape TMs	266
10.8	The Linguistic Implications of the Equivalence of Non-deterministic and Deterministic TMs.	267
11.	Computability, Solvability and the Halting Problem	269
11.1	Overview	269
11.2	The Relationship Between Functions, Problems, Solvability and Decidability	270
11.2.1	Functions and Computability	270
11.2.2	Problems and Solvability	271

11.2.3	Decision Problems and Decidability	272
11.3	The Halting Problem	273
11.3.1	UTM_H Partially Solves the Halting Problem	274
11.3.2	<i>Reductio ad Absurdum</i> Applied to the Halting Problem	275
11.3.3	The halting problem shown to be unsolvable	277
11.3.4	Some Implications of the Unsolvability of the Halting Problem	280
11.4	Computable Languages	282
11.4.1	An Unacceptable (non-Computable) Language	283
11.4.2	An Acceptable, But Undecidable, Language.	285
11.5	Languages and Machines	286
12.	Dimensions of Computation	291
12.1	Overview.	291
12.2	Aspects of Computation: Space, Time and Complexity	292
12.3	Non-Deterministic TMs Viewed as Parallel Processors	294
12.3.1	Parallel Computations and Time	296
12.4	A Brief Look at an Unsolved Problem of Complexity	298
12.5	A Beginner's Guide to the "Big O"	298
12.5.1	Predicting the Running Time of Algorithms.	298
12.5.2	Linear time.	300
12.5.3	Logarithmic Time	302
12.5.4	Polynomial Time	305
12.5.5	Exponential Time	313
12.6	The Implications of Exponential Time Processes	315
12.6.1	"Is P Equal to NP ?"	316
12.7	Observations on the Efficiency of Algorithms	317
	Further Reading	319
	Solutions to Selected Exercises	323
	Chapter 2	323
	Chapter 3	325
	Chapter 4	326
	Chapter 5	328
	Chapter 6	329
	Chapter 7	330
	Chapter 8	330
	Chapter 9	330
	Chapter 10	331
	Chapter 11	331
	Chapter 12	331
	Index	335

Preface

Aims and Objectives

This book focuses on key theoretical topics of computing, in particular formal languages and abstract machines. It is intended primarily to support the theoretical modules on a computer science or computing-related undergraduate degree scheme.

Though the book is primarily theoretical in nature, it attempts to avoid the overly mathematical approach of many books on the subject and for the most part focuses on encouraging the reader to gain an intuitive understanding. Proofs are often only sketched and, in many cases, supported by diagrams. Wherever possible, the book links the theory to practical considerations, in particular the implications for programming, computation and problem solving.

Organisation and Features of the Book

There is a short introductory chapter that provides an overview of the book and its main features. The remainder of the book is in two parts, *Languages and Machines* and *Machines and Computation*.

Part 1, Languages and Machines, is concerned with formal language theory as it applies to Computer Science. It begins with an introduction to the notation and concepts that support the theory, such as strings, the various ways in which a formal language can be defined, and the Chomsky hierarchy of formal language. It then focuses on the languages of the Chomsky hierarchy, in each case also introducing the abstract machines associated with each type of language.

The topics are regular languages and finite state recognisers, context free languages and pushdown recognisers, and context sensitive and unrestricted languages and the Turing machine. Many important theoretical properties of regular and context free languages are established. The more intellectually demanding of these results are mostly confined to a single chapter, so that the reader can focus on the general thrust of the argument of Part 1, which is to demonstrate that the Chomsky hierarchy is indeed a proper hierarchy for both languages and machines, and to consider some of the implications of this.

In the first part of the book the finite state machine, the pushdown machine and the Turing machine are considered as language recognisers, though many hints are given about the potential computational properties of these abstract machines.

Part 2, Machines and Computation, considers the computational properties of the machines from Part 1 in more detail. The relationship between finite state machines and the digital computer is explored. This leads us on to the need for a more powerful machine to deal with arbitrary computation. This machine is shown to be the Turing machine, introduced in Part 1 as a language processor.

The Turing machine is used to explore key aspects of computation, such as non-determinism, parallel processing and the efficiency of computational processes. The latter is considered in the context of a brief introduction to algorithm analysis, using big O notation.

The book also considers the limitations of computation, both in terms of language processing (simply defined formal languages that cannot be processed by any machine) and computation (the halting problem and related issues).

The book contains numerous illustrative figures, and proofs are often partly accomplished through diagrammatic means.

From Chapter 2 onwards, each chapter concludes with exercises, some of which are programming exercises. Solutions and hints for many of the exercises appear at the end of the book.

The book also contains a list of recommended reading material.

For Students

I wrote this book partly because when I studied this material as part of my own Computing degree, I had to work really hard to understand the material, a situation which arose not because the material is too difficult, but because it was not well presented and the books seemed to assume I was a pure mathematician, which I am not.

This book is primarily for undergraduate students of computing, though it can also be used by students of computational linguistics and researchers, particularly those entering computer science from other disciplines, who find that they require a foundation or a refresher course in the theoretical aspects of computing.

Some aspects of the book are certainly clearer if the student has some experience of programming, though such experience is not essential for understanding most of the book.

The reader is advised where especially demanding material can be omitted, though he or she is encouraged to appreciate the implications of that material, as such an appreciation may be assumed later in the book.

For Instructors

I wrote this book partly because when I taught this material to undergraduates, as a Computer Science lecturer in a UK university, I had to work really hard to present the material in an appropriate way, and at the appropriate level, a situation which arose mainly because the available books seemed to assume the students were pure mathematicians, which mostly they are not.

This is intended to be not only a book to promote understanding of the topics for students, but also a recommendation of the core material that should be covered in a theoretical computer science module for undergraduates. I suggest that to cover all of the material would require at least a module in each semester of one year of the degree course. If possible, the linguistic aspects should precede a compiler course and the computational aspects should precede an algorithm analysis course. If the student has programming experience the material is much more accessible. This should influence the way it is presented if it appears early on in the degree scheme.

There is, of course, much more theory relating to programming language design, compiler writing, parallel processing and advanced algorithm analysis than presented in this book. However, such additional theory is best covered in the modules that concern these subjects, some of which are often optional in a computing degree scheme.