

Texts in Computing

Volume 22

Languages, Machines, and Classical Computation

Volume 9 Logic for Artificial Intelligence & Information Technology Dov M. Gabbay

Volume 10 Foundations of Logic and Theory of Computation Amílcar Sernadas and Cristina Sernadas

Volume 11 Invariants: A Generative Approach to Programming Daniel Zingaro

Volume 12 The Mathematics of the Models of Reference Francesco Berto, Gabriele Rossi and Jacopo Tagliabue

Volume 13 Picturing Programs Stephen Bloch

Volume 14 JAVA: Just in Time John Latham

Volume 15 Design and Analysis of Purely Functional Programs Christian Rinderknecht

Volume 16 Implementing Programming Languages. An Introduction to Compilers and Interpreters Aarne Ranta, with an appendix coauthored by Markus Forsberg

Volume 17 Acts of the Programme *Semantics and Syntax*. Isaac Newton Institute for the Mathematical Sciences, January to July 2012. Arnold Beckmann and Benedikt Löwe, eds.

Volume 18 What Is a Computer and What Can It Do? An Algorithms-Oriented Introduction to the Theory of Computation Thomas C. O'Connell

Volume 19 Computational Logic. Volume 1: Classical Deductive Computing with Classical Logic Luis M. Augusto

Volume 20 An Introduction to Ontology Engineering C. Maria Keet

Volume 21 A Mathematical Primer on Computability Amílcar Sernadas, Cristina Sernadas, João Rasga and Jaime Ramos

Volume 22 Languages, Machines, and Classical Computation Luis M. Augusto

Texts in Computing Series Editor lan Mackie

mackie@lix.polytechnique.fr

Languages, Machines, and Classical Computation

Luis M. Augusto

© Individual author and College Publications 2019. All rights reserved.

ISBN 978-1-84890-300-5

College Publications Scientific Director: Dov Gabbay Managing Director: Jane Spurr

http://www.collegepublications.co.uk

Cover produced by Laraine Welch

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form, or by any means, electronic, mechanical, photocopying, recording or otherwise without prior permission, in writing, from the publisher.

Contents

Pı	reface x	ii			
I	Introduction	1			
1	 Classical computation: Turing, von Neumann, and Chomsky 1.1 Computers, information, and computations				
	 1.3 The Turing-von Neumann paradigm	4 8 9			
II	Preliminaries and notation 1	L 4			
2	Mathematical notions 1 2.1 Basic notions 1 2.1.1 Sets, relations, functions, and operations 1 2.1.2 Binary relations and ordered sets 2 2.2 Discrete structures 3 2.2.1 Algebraic structures and operations 3 2.2.1.1 Algebras and morphisms 3 2.2.1.2 Boolean algebras 3 2.2.2.3 Graphs and trees 3 2.3.1 Mathematical and structural induction 4 2.3.2 Proof by contradiction 4	15 15 24 31 32 33 38 43 44 45			
111 3	Languages, machines, and classical computation 4 Formal grammars and languages 4 3.1 Basic notions 5 3.1.1 Strings and operations on strings 5	18 19 50 50			

Contents

		3.1.2	Formal	languages and operations thereon 52
		3.1.3	Formal	grammars
			3.1.3.1	Central notions $\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots 55$
			3.1.3.2	Rules, symbols, and grammar cleaning . 57
	3.2	Regul	ar langua	$ges \ldots 65$
		3.2.1	Regular	expressions
		3.2.2	Regular	grammars
		3.2.3	Propert	ies of the regular languages
		0.2.0	3231	Pumping lemma for regular languages 76
			3232	Algebra and linear equations for regular
			0.2.0.2	languages 77
			2923	Closure properties of the regular languages 70
	33	Conte	vt_free la	nguages 84
	0.0	331	Context	-free grammars 84
		0.0.1	3 3 1 1	Context free vs_context_sensitive gram-
			0.0.1.1	mars 84
			2219	Normal forms for CECs I: Chomsky nor
			0.0.1.2	mal form
			2212	Normal forms for CECs II: Croibach nor
			0.0.1.0	mal form 00
			2214	Derivation or parso troop 05
			0.0.1.4 0.0.1 E	Ambiguity and inhonent ambiguity 06
		<u></u>	0.0.1.0 Dron ort	Amonguity and innerent amonguity 90
		3.3.2	Propert	Beson the context-free languages 101
			3.3.2.1	Pumping lemma for CFLs and Ogden's
				$\mathbf{E}_{\mathrm{remma}} = \mathbf{E}_{\mathrm{remma}} = \mathbf{E}_{re$
	0.4	D	3.3.2.2	Further properties of CFLs 104
	3.4	Recur	sively enu	$\begin{array}{c} \text{imerable languages} \dots \dots \dots \dots \dots \dots 112 \\ \begin{array}{c} \dots \\ \dots $
	3.5	The C	bomsky l	$\operatorname{hierarchy}(1) \dots \dots \dots \dots \dots \dots \dots \dots \dots $
Л	Мо	dole of	comput	nation 191
-	1 110	Finito	. comput	chines 121
	4.1	1 1 1 1 1 1	1 Finite automata 120	
		4.1.1	11110 a	Basic aspects of finite automata 123
			4.1.1.1	Characteristic equations
			4.1.1.2	The numping lemma for regular languages 121
			4.1.1.3	The pumping lemma for regular languages 151
			4.1.1.4	i ne Myniii-Nerode theorem and FA min-
			4 1 1 E	Deterministic and non deterministic EA = 129
			4.1.1.0	Viene's theorem and the surrentian (
			4.1.1.0	Received theorem and the properties of
		410	D ::	<i>MSZ</i>
		4.1.2	Finite ti	ransaucers
			4.1.2.1	Moore and Mealy machines

Contents

		4.1.2.2	Equivalence of finite transducers	. 154
		4.1.2.3	Minimizing finite transducers	. 156
		4.1.2.4	Conversion of transducers into acceptors	s 161
4.2	Pushd	own auto	omata	. 170
	4.2.1	Basic as	spects of PDAs	. 170
	4.2.2	Two acc	ceptance modes by PDAs: Final state and	l
		empty s	tack	. 174
	4.2.3	Equival	ence between CFLs and PDAs	. 175
	4.2.4	CFLs ac	ccepted by deterministic PDAs	. 180
		4.2.4.1	Deterministic PDAs	. 180
		4.2.4.2	LR(k) grammars	. 182
4.3	Turing	g machine	es	. 196
	4.3.1	Basic as	spects of Turing machines	. 196
	4.3.2	Turing 1	machines computing functions $\ldots \ldots$. 199
	4.3.3	Turing 1	machines accepting languages	. 201
		4.3.3.1	Turing machines and unrestricted gram-	-
			mars	. 201
		4.3.3.2	Linear-bounded automata: Special Tur-	
			ing machines for CSGs	. 205
	4.3.4	The uni	versal Turing machine	. 206
4.4	The C	homsky l	$nierarchy (II) \dots \dots$. 212
5 Cor	nputab	oility and	d complexity	216
5.1	The de	ecision pr	oblem and Turing-decidability	. 216
5.2	Undec	idable pr	oblems and Turing-reducibility	. 219
5.3	The C	homsky l	nierarchy (III)	. 225
5.4	Comp	utational	complexity	. 227
	5.4.1	Comput	ational problems	. 227
	5.4.2	The Blu	im axioms and complexity measures	. 228
	5.4.3	Complex	xity classes \ldots \ldots \ldots \ldots \ldots \ldots	. 232
	5.4.4	The Co	ok-Levin theorem and polynomial-time re-	
		ducibilit	у	. 237
5.5	The C	homsky l	nierarchy (IV)	. 249
Biblio	graphy	V		251
D I I I				
Biblio	graphic	cal refer	ences	252
Index				256

List of Figures

$1.5.1 \\ 1.5.2$	The basic postulate of the Chomsky hierarchy.11Two derivation trees.13
$2.1.1 \\ 2.1.2 \\ 2.2.1$	A partially ordered set.27Hasse diagram of a poset.31A simple graph with five vertices and seven edges.40
$3.2.1 \\ 3.2.2$	A labeled digraph $\vec{\mathfrak{G}}(r)$ for a regular expression $r. \ldots 71$ A labeled digraph $\vec{\mathfrak{G}}(G)$ corresponding to a left-linear
$3.2.3 \\ 3.3.1$	The digraph $\vec{\mathfrak{G}}'(G')$ obtained from $\vec{\mathfrak{G}}(G)$
3.3.2	corresponding partial derivation trees
$3.3.3 \\ 3.3.4$	Parse tree of an unambiguously derived string 100 Parse trees for productions (1) $S \rightarrow a$ and (2) $S \rightarrow AB$ 102
$3.3.5 \\ 3.4.1$	Parse tree for $z = uv^i wx^i y. \dots $
4 1 1	
4.1.1	State diagrams of FAs 126
4.1.2	Δ FA with two accepting states and one rejecting state 126
4.1.0	A FA for the regular language $L = \{c, ha\}^* \{ac, aah^*\}$ 127
415	A finite automaton M for the numping lemma 132
4.1.6	A FA (1) and its minimal equivalent FA (2)
4.1.7	A NDFA for the language $L = \{0,01\}^* \{0,010\}^*$
110	
4.1.0	Equivalent NDFAs with and without ϵ -transitions 142
4.1.8 4.1.9	Equivalent NDFAs with and without ϵ -transitions
4.1.8 4.1.9 4.1.10	Equivalent NDFAs with and without ϵ -transitions
4.1.8 4.1.9 4.1.10	Equivalent NDFAs with and without ϵ -transitions
4.1.8 4.1.9 4.1.10 4.1.11	Equivalent NDFAs with and without ϵ -transitions
$\begin{array}{c} 4.1.8 \\ 4.1.9 \\ 4.1.10 \\ 4.1.11 \\ 4.1.12 \end{array}$	Equivalent NDFAs with and without ϵ -transitions
4.1.8 4.1.9 4.1.10 4.1.11 4.1.12 4.1.13	Equivalent NDFAs with and without ϵ -transitions
$\begin{array}{c} 4.1.8 \\ 4.1.9 \\ 4.1.10 \\ \\ 4.1.11 \\ 4.1.12 \\ 4.1.13 \\ 4.1.14 \end{array}$	Equivalent NDFAs with and without ϵ -transitions
$\begin{array}{c} 4.1.8 \\ 4.1.9 \\ 4.1.10 \\ \\ 4.1.11 \\ 4.1.12 \\ 4.1.13 \\ 4.1.14 \\ 4.1.15 \end{array}$	Equivalent NDFAs with and without ϵ -transitions 142 Equivalent NDFA (1) and FA (2)
$\begin{array}{c} 4.1.8 \\ 4.1.9 \\ 4.1.10 \\ \\ 4.1.11 \\ 4.1.12 \\ 4.1.13 \\ 4.1.13 \\ 4.1.15 \\ 4.1.16 \end{array}$	Equivalent NDFAs with and without ϵ -transitions

List of Figures

4.1.17	Mealy machines
4.1.18	A barcode
4.2.1	Computer model for a PDA
4.2.2	A PDA <i>M</i> accepting the language $L(M) = \{a^m b^m m \ge 0\}.$
	173
4.2.3	Proving the equivalence of $L(M) = N(M)$
4.2.4	NDFA recognizing the viable prefixes for the CFG of Bal-
	anced Parentheses
4.2.5	Pushdown automata
4.2.6	A PDA accepting $L(M) = \{ u \in \Sigma^* u = ww^R \}$
4.3.1	Computer model for a Turing machine
4.3.2	A Turing machine that computes the function $f(m, n) =$
	$m+n$ for $m, n \in \mathbb{Z}^+$
4.3.3	Turing machine M_T that computes the function $f(m, n) =$
	$2m+3n$ for $m, n \in \mathbb{Z}^+$
4.3.4	Program for Turing machine M_T that computes the func-
	tion $f(m,n) = 2m + 3n$ for $m, n \in \mathbb{Z}^+$
4.3.5	The encodings $\langle M_T \rangle$ and $\langle M_T, z \rangle$
4.3.6	A combination of Turing machines
4.3.7	A Turing machine
501	
5.2.1	A combination of Turing machines
5.3.1	The Chomsky hierarchy and beyond: Decidable, Turing-
	recognizable, and not-Turing-recognizable languages 226
5.4.1	The hierarchy of complexity classes with corresponding
	tractability status
5.4.2	A tableau for the Turing machine M
5.4.3	Typical structure of NP -completeness proofs by polynomial-
	time reductions. $\ldots \ldots 245$

List of Tables

3.5.1	The Chomsky hierarchy
4.4.1	The extended Chomsky hierarchy: Grammars, languages, and associated computer models
5.3.1	Decidability ("Yes") and undecidability ("No") of some prop- erties of interest for the Chomsky hierarchy
5.4.1	Rates of growth of some standard functions

List of Algorithms

3.1	Grammar cleaning 59
3.2	Left-/Right-linear grammar to right-/left-linear grammar 75
3.3	Chomsky-normal-form Transformation
3.4	Greibach-normal-form Transformation $\ldots \ldots \ldots \ldots 91$
3.5	Language class by grammar type
4.1	Deterministic FA minimization
4.2	Subset-construction algorithm
4.3	Partition refinement for the states of a Mealy machine $\ . \ . \ 159$
4.4	Mealy machine minimization

Teachers tend to be picky with the material they use in teaching contexts. This may be for personality reasons, but the variety of contexts and students also plays a role in this pickiness. Be it as it may, it often is the case that students end up with teaching material in many formats and from many different sources, creating often a lack of uniformity, both in notation and terminology. Because I am picky for all the reasons above, I typically feel that my teaching task is substantially facilitated and optimized when I have gone to the great lengths of putting all the material for a particular academic subject together in a single manual or textbook. This guarantees not only conceptional and notational uniformity, but also a selection of approaches that I feel work well, or better, for particular topics or problems.

This book is not about discovering the wheel; that is, possibly no novel contents are to be found in it. The objective when writing it was that of "putting together" a textbook on the classical theory of computing. If there is any novel aspect in this textbook, it may well be the fact that I insist on preceding the terms "(theory of) computation" and "(theory of) computing" with the adjective "classical" to collect under the same label the Chomsky hierarchy and the Turing-von Neumann paradigm of computing. The former comprises three closely associated central topics, to wit, formal grammars, formal languages, and models of computation (a.k.a machines, or automata), and the latter gives to these, namely via the Turing machine, measures of the spatial and temporal costs of computation. I say that this collection constitutes (the) classical (theory of) computation, because many, often newer, other forms of computing have emerged or become (more) popular since the Turing "revolution," many of which today may be said to constitute the non-classical (theory of) computation. This is, for the initiated, more immediately the field of quantum computing, but other forms of computation such as artificial neural networks and evolutionary computing may be seen as also nonclassical versions of computing.

It is arguably possible to produce a textbook on formal languages, grammars, and automata with no emphasis on computing, let alone with any specific computational concerns. One such approach might be with linguists in mind, though contemporary linguistics is not averse to

computation. On the extreme pole of this position, formal grammars, languages, and automata are often reduced to *the* theory of computation, namely as it serves the theoretical foundations of the digital computer. Without taking a reductive view, I discuss formal languages and grammars from the viewpoint of computation, and consider the associated automata as models thereof. This said, readers with other foci will find that the computational perspective taken here does not hinder–and may even facilitate–their particular interests and concerns.

The backbone of this book is undoubtedly the Chomsky hierarchy. Although much computing has run in the digital computer since N. Chomsky first conceived it, it still works well for combining the mostly linguistic approach with the computational one. In particular, it keeps reminding us that we are linguistic beings to the point that one of our most interesting creations—the digital computer—is language-based through and through, a feature well-patent in the famous Turing Test, a "test" conceived by the creator of the Turing machine to distinguish a human computer from a non-human one. Indeed, it seems to have been the rationale in Turing (1950) that language is sufficient to distinguish the human from the non-human computer or reasoner. More than anything, it might have been this insistence on the verbal behavior of computers that motivated the can-of-worms idea of AI (artificial intelligence) as ultimately aiming at human-like machines, at least from the viewpoint of intelligence, if not of emotion.

There is no way to go around this and it requires emphasis: (classical) computing is a mathematical subject. Although the presence of automata, of which the most famous is the Turing machine, lends it a flavor of engineering, these are not physical machines nor can they be; they are mathematical objects. To be sure, the digital computer is based on the Turing machine, but this has a feature an infinite tape-that makes of the former a mere approximation of the latter. The mathematical nature of this subject accounts for the clearly mathematical approach in this book: I distinguish statements into definitions and propositions, and provide proofs (or sketches thereof) to further distinguished-if not distinct-statements, to wit, theorems and their companion lemmas and corollaries. The numbering of such statements finds its utility in internal referencing, if it gives a more high-brow quality to the main text. I reserve the status of theoremhood for statements of higher importance than propositions, but the reader is free to consider (most) propositions in this text as *de-facto* theorems; the fact that proofs are provided (or left as exercises) for propositions supports this view.

This mathematical nature of the subject also justifies the large selection of exercises here provided. Indeed, only few students are gifted

with mathematical skills that free them from the arduous and timeconsuming practice of doing exercises. On the other hand, some may find this a pleasant activity. Between these fall most mortals, one should think. But the selection of exercises in this book was also guided by the belief that one should be confronted with novel material and problems, in order to develop research, as well as creative, skills.

Still with regard to the mathematical nature of this text, there are throughout it a few algorithms for the computation of specific functions (e.g., computing the Chomsky normal form of a given grammar). I chose not to stick to a single pseudo-code or to a single algorithm format in the belief that different algorithms can be better grasped in distinct ways. Yet another advantage of this might be the familiarity with diverse pseudo-codes and algorithm formats. Importantly, too, no programming language or software plays any role whatsoever in this book. This is so deliberately to keep the subject matter as general as possible, untied to specific implementations or applications.

As said above, the aim for this book is not (re)inventing the wheel. Although classical computing and its theory are in a current state of development, with many a problem as focus of research-notably so the $\mathbf{P}=?\mathbf{NP}$ problem-, the subject of the theory of classical computing has attained a certain fixed form that is historically justified. In the second half of last century, when this subject emerged, an abundance of textbooks and monographs were published, and a few of these established themselves as standard references in the field. As such, it is only natural that in pedagogical pursuits one should resort to them as sources. This I do with two such classics in particular, to wit, Davis & Weyuker (1983) and Hopcroft & Ullman (1979), the latter of which has evolved into the more undergraduate-friendly Hopcroft, Motwani, & Ullman (2013). A further source is Du & Ko (2001), a thoroughly mathematical approach. Readers can greatly benefit from a direct use of all these referenced works. Texts and manuals on this subject matter directed at undergraduate audiences abound, with many a good one to further assist readers in their academic pursuits. Referencing them all is of course impossible, but interested readers know where to find them. More specific, often more advanced, literature is cited throughout this text in the appropriate places; in particular, I cite the works in which important results (e.g., theorems) were first published.

Lastly, this textbook is a further elaboration on what was originally a chapter in a book of mine first published by College Publications, to wit, Augusto (2018). In this book, a chapter on the theory of computing appeared to be relevant, because issues such as Turing-completeness of logic programming and the complexity of the satisfiability problem

(a.k.a SAT) required a minimal grasp of, among other topics, the Turing machine. Having resorted to this chapter to teach topics in automata, formal languages, and the classical theory of computation, and having obtained satisfactory results, I decided to expand it to what is now the present textbook. The main guideline for this expansion was the inclusion of topics that were left out in the mentioned chapter for spatial and temporal reasons, but which are essential for a fuller treatment of this subject. Some of these new topics–e.g., characteristic equations of finite automata, grammar cleaning algorithm–may appear quite inessential from an Anglo-Saxon perspective, but my individual work with Spanish students preparing themselves to take exams on the above-mentioned topics made me realize the need to be as encompassing and comprehensive as possible, namely with the large diversity of readers of this subject in mind.

I wish to thank Dov M. Gabbay, the scientific director of College Publications, and Ian Mackie, the editor for the Texts in Computing series, for publishing this book. My thanks go also to Jane Spurr, the managing director, for a smooth publication process.

Madrid, February 2019

Luis M. S. Augusto