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
Ian Mackie mackie@lix.polytechnique.fr

Languages, Machines, and Classical Computation

Third Edition

Luis M. Augusto

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

Contents

List of Figures

List of Tables

List of Algorithms

Preface to the 1st edition

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 non-classical 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 time-consuming 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 **P**=?**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).[1] In this book, a chapter on the theory of comput-

[1]Now Augusto (2020a).

ing 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

Preface to the 2nd edition

The present edition corrects identified addenda and errata, and has both improved and new figures. It also includes the odd minor change in the main text of the first edition. The major changes are as follows:

- Figures 4.1.11 and 4.3.7 were replaced by more adequate ones: In the case of Figure 4.1.11, the union of languages was rather opaque, and the original finite automaton was replaced by a clearer illustration; as for Figure 4.3.7, the original Turing machine, which was too simple, was replaced by a more complex one.

- Exercise 4.1.9 (originally 4.1.14) has now five items, an increase aiming at providing more practice of a complex algorithm.

- An additional algorithm for the conversion of a context-free grammar into a pushdown automaton (Algorithm 4.5) is now included and, based on it, Example 4.2.3 was greatly revised and extended. This change entailed a new Figure (4.2.7),[1] to be found in Exercise 4.2.8, which was completely redesigned.

These changes are now made available thanks to the readiness of College Publications to publish a second edition so shortly after the first. Renewed thanks are in order.

Madrid, May 2020

Luis M. S. Augusto

[1]Figure 4.2.4 in this 3rd edition.

Preface to the 3rd edition

The present edition improves on the previous ones in many ways, of which a few can be specified:

- Many exercises were introduced and others were rewritten, in particular in Chapter 2. In this Chapter, the emphasis fell on functions and algorithms, two topics that were not satisfactorily–in any case not sufficiently–discussed in the previous editions. Many of the functions and algorithms involve recursion, a fundamental notion in classical computation that was also not sufficiently discussed before. In the belief that research is an essential task in the understanding of mathematical topics, a belief that had already been present in the writing of the two previous editions, many of the exercises introduced require that the student research integrally in specific topics (e.g., Ackermann functions).
- A wholly new Section on algorithms and programs (Section 2.4) was introduced.
- Further topics in regular languages and finite automata were introduced, namely the Thompson construction, product automata, and the direct sum of automata. This, in turn, motivated the introduction of two new algorithms, to wit, Algorithms 4.3 and 4.4.
- More exercises with Turing machines were added, both for function-computing and language-accepting machines.

Thanks are due to College Publications for their readiness to publish the present 3rd edition.

Madrid, October 2021

Luis M. S. Augusto

References

Anderson, J. A. (2006). *Automata theory with modern applications.* Cambridge, etc.: Cambridge University Press.

Arden, D. N. (1961). Delayed-logic and finite-state machines. *Proceedings of the 2nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, 133-151.

Augusto, L. M. (2018). *Computational logic. Vol. 1: Classical deductive computing with classical logic.* London: College Publications.

Augusto, L. M. (2019). *Formal logic: Classical problems and proofs.* London: College Publications.

Augusto, L. M. (2020a). *Computational logic. Vol. 1: Classical deductive computing with classical logic.* 2nd ed. London: College Publications.

Augusto, L. M. (2020b). *Many-valued logics. A mathematical and computational introduction.* 2nd ed. London: College Publications.

Augusto, L. M. (2020c). *Logical consequences. Theory and applications: An introduction.* 2nd ed. London: College Publications.

Augusto, L. M. (2020d). Toward a general theory of knowledge. *Journal of Knowledge Structures & Systems*, *1*(1), 63-97.

Bar-Hillel, Y., Perles, M., & Shamir, E. (1961). On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, *14*(2), 143-172.

Bloch, E. D. (2011). *Proofs and fundamentals: A first course in abstract mathematics.* 2nd ed. New York, etc.: Springer.

Blum, M. (1967). A machine-independent theory of the complexity of recursive functions. *Journal of the Association for Computing Machinery*, *14*(2), 322-336.

Bridges, D. S. (1994). *Computability. A mathematical sketchbook.* New York, etc.: Springer.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113-124.

Chomsky, N. (1957). *Syntactic structures.* The Hague & Paris: Muton.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, *2*(2), 113-124.

Chomsky, N. (1962). Context-free grammars and pushdown storage. *MIT Quarterly Progress Reports*, *65*, 187-194.

Church, A. (1936a). A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, *1*(1), 40-41.

Church, A. (1936b). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, *58*(2), 345-363.

Cook, S. A. (1971). The complexity of theorem proving procedures. *Proceedings of the 3rd Annual ACM Symposium of Theory of Computing*, 151-158.

Cooper, S. B. (2003). *Computability theory.* Boca Raton, etc.: CRC Press.

Cooper, K. D. & Torczon, L. (2012). *Engineering a compiler.* 2nd ed. Amsterdam, etc.: Morgan Kaufmann.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* 3rd ed. Cambridge, MA & London, UK: MIT Press.

Crama, Y. & Hammer, P. L. (2011). *Boolean functions: Theory, algorithms and applications.* Cambridge, etc.: Cambridge University Press.

Crochemore, M., Hancart, C., & Lecroq, T. (2007). *Algorithms on strings.* Cambridge, etc.: Cambridge University Press.

Davis, M. D. & Weyuker, E. J. (1983). *Computability, complexity, and languages. Fundamentals of theoretical computer science.* Orlando, etc.: Academic Press.

Du, D.-Z. & Ko, K.-I. (2001). *Problem solving in automata, languages, and complexity.* New York, etc.: John Wiley & Sons.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution.* New York, NY: John Wiley & Sons.

Gallier, J. (2011). *Discrete mathematics.* New York, etc.: Springer.

Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness.* New York: W. H. Freeman and Company.

Gödel, K. (1964). Postscriptum to Gödel (1934). In *Collected works I* (pp. 369-371). Oxford: Oxford University Press, 1986.

Gopalakrishnan, G. (2006). *Computation engineering. Applied automata theory and logic.* New York: Springer.

Grune, D. & Jacobs, C. J. H. (2010). *Parsing techniques: A practical guide.* 2nd ed. New York, NY: Springer.

Hausser, R. (2014). *Foundations of computational linguistics. Human-computer communication in natural language.* 3rd ed. Heidelberg, etc.: Springer.

Hilbert, D. & Ackermann, W. (1928). *Grundzüge der theoretischen Logik.* Berlin: Springer.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Hopcroft, J. E & Ullman, J. (1979). *Introduction to automata theory, languages, and computation.* 1st ed. Reading, MA, etc.: Addison-Wesley.

Hopcroft, J. E., Motwani, R., & Ullman, J. (2013). *Introduction to automata theory, languages, and computation.* 3rd ed. Boston, etc.: Pearson.

Khoussainov, B. & Nerode, N. (2001). *Automata theory and its applications.* New York: Springer.

Kleene, S. C. (1938). On notation for ordinal numbers. *Journal of Symbolic Logic*, *3*(4), 150-155.

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In C. E. Shannon & J. McCarthy (eds.), *Automata studies* (pp. 3-42). Princeton: Princeton University Press.

Kohavi, Z. & Jha, N. (2010). *Switching and finite automata theory.* 3rd ed. Cambridge, etc.: Cambridge University Press.

Leary, C. C. & Kristiansen, L. (2015). *A friendly introduction to mathematical logic.* 2nd ed. Geneseo, NY: Milne Library.

Lovelace, A. (1843). Notes on L. Menabrea's "Sketch of the Analytical Engine invented by Charles Babbage, Esq." *Taylor's Scientific Memoirs*, vol. 3. London: J. E. & R. Taylor.

Makinson, D. (2008). *Sets, logic, and maths for computing.* London: Springer.

Matiyasevich, Y. V. (1993). *Hilbert's Tenth Problem*. Cambridge, MA: MIT Press.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115-133.

Mealy, G. H. (1955). A method for synthesizing sequential circuits. *Bell System Technical Journal*, *34*(5), 1045-1079.

Moore, E. F. (1956). Gedanken-experiments on sequential machines. *Automata Studies, Annals of Mathematical Studies*, *34*, 129-153.

Nerode, A. (1958). Linear automaton transformations. *Proceedings of the AMS*, *9*(4), 541-544.

Oettinger, A. G. (1961). Automatic syntactic analysis and the pushdown store. In R. Jakobson (ed.), *Structure of language and its mathematical aspects* (pp. 104-139). *Proceedings of Smposia in Applied Mathematics*, *12*. Providence, RI: American Mathematical Society.

Ogden, W. (1968). A helpful result for proving inherent ambiguity. *Mathematical Systems Theory*, *2*, 191-194.

Rabin, M. O. & Scott, D. (1959). Finite automata and their decision problems. *IBM Journal*, *3*(2), 115-125.

Reghizzi, S. C., Breveglieri, L., & Morzenti, A. (2019). *Formal languages and compilation.* 3rd ed. London: Springer.

Révész, G. E. (1991). *Introduction to formal languages.* Minneola, NY: Dover.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group (1988). *Parallel distributed processing. Explorations in the microstructure of cognition. Vol. 1: Foundations.* Cambridge, MA & London, UK: The MIT Press.

Sakarovitch, J. (2009). *Elements of automata theory.* Cambridge: Cambridge University Press.

Scott, M. L. (2009). *Programming language pragmatics.* 3rd ed. Amsterdam, etc.: Morgan Kaufmann.

Sebesta, R. W. (2012). *Concepts of programming languages.* 10th ed. Boston, etc.: Pearson.

Sippu, S. & Soisalon-Soininen, E. (1990). *Parsing theory. Vol. II: LR(k) and LL(k) parsing.* Berlin, Heidelberg: Springer.

Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, *42*(1), 230-265.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*(236), 433-460.

von Neumann, J. (1945). *First draft of a report on the EVDAC.* Technical report. University of Pennsylvania. (Reprinted in B. Randell (ed.), *The origins of digital computers. Selected papers* (pp. 383-392). 3rd ed. Berlin, etc.: Springer. 1982.)

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, *10*(2), 189-208.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338-353.

Index

Index