

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.

For further volumes:
<http://www.springer.com/series/7592>

Maribel Fernández

Models of Computation

An Introduction
to Computability Theory

Dr. Maribel Fernández
King's College London
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

ISSN 1863-7310

ISBN 978-1-84882-433-1

e-ISBN 978-1-84882-434-8

DOI 10.1007/978-1-84882-434-8

Springer Dordrecht Heidelberg London New York

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

Library of Congress Control Number: Applied for

© Springer-Verlag London Limited 2009

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 licenses 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

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Aim

The aim of this book is to provide an introduction to essential concepts in computability, presenting and comparing alternative models of computation. We define and analyse the most significant models of computation and their associated programming paradigms, from Turing machines to the emergent computation models inspired by systems biology and quantum physics.

About this book

This book provides an introduction to computability using a series of abstract models of computation.

After giving the historical context and the original challenges that motivated the development of computability theory in the 1930s, we start reviewing the traditional models of computation: Turing machines, Church's Lambda calculus (or λ -calculus), and the theory of recursive functions of Gödel and Kleene. These three models of computation are equivalent in the sense that any computation procedure that can be expressed in one of them can also be expressed in the others. Indeed, Church's Thesis states that the set of computable functions is exactly the set of functions that can be defined in these models.

Each of the above-mentioned models of computation gave rise to a programming paradigm: imperative, functional, or algebraic. We also include in the first part of the book a computation model based on deduction in a fragment of first-order logic, which gave rise to the logic programming paradigm,

because the work by Herbrand in this area dates also from the late 1920s and early 1930s.

As programming languages evolved and new programming techniques were developed, other models of computation became available; for instance, based on the concept of object or on a notion of interaction between agents. It is possible, for example, to show that any computable function can be defined by using an abstract device where one can define objects, invoke their methods, and update them. In the second part of the book, we describe a calculus of objects as a foundation for object-oriented programming and compare its computational power with the traditional ones. We also describe a graphical, interaction-based model of computation and a formalism for the specification of concurrent computations.

Recently, there has been a renewed interest in computability theory, with the emergence of several models of computation inspired by biological and physical processes. In the last chapter of the book, we discuss biologically inspired calculi and quantum computing.

This book is addressed to advanced undergraduate students, as a complement to programming languages or computability courses, and to postgraduate students who are interested in the theory of computation. It was developed to accompany lectures in a Master's course on models of computation at King's College London. The book is for the most part self-contained; only some basic knowledge of logic is assumed. Basic programming skills in one language are useful, and knowledge of more programming languages will be helpful but is not necessary.

Each chapter includes exercises that provide an opportunity to apply the concepts and techniques presented. Answers to selected exercises are given at the end of the book. Although some of the questions are just introductory, most exercises are designed with the goal of testing the *understanding* of the subject; for instance, by requiring the student to adapt a given technique to different contexts.

Organisation

The book is organised as follows. Chapter 1 gives an introduction to computability and provides background material for the rest of the book, which is organised into two parts.

In Part I, we present the traditional models of computation. We start with the study of various classes of automata in Chapter 2. These are abstract machines defined by a collection of states and a transition function that con-

trols the way the machine's state changes. Depending on the type of memory and the kind of response that the automaton can give to external signals, we obtain machines with different computation power. After giving an informal description, we provide formal specifications and examples of finite automata, push-down automata, and Turing machines. The chapter ends with a discussion of the applications of these automata to programming language design and implementation.

The next two chapters are dedicated to the study of computation models inspired by the idea of “computation as functional transformation”. In Chapter 3, we give an overview of the λ -calculus, with examples that demonstrate the power of this formalism, highlighting the role of the λ -calculus as a foundation for the functional programming paradigm. In Chapter 4, we define primitive recursion and the general class of partial recursive functions.

The final chapter in Part I describes a model of computation based on deduction in a fragment of first-order logic. We introduce the Principle of Resolution and the notion of unification. We then study the link between these results and the development of logic programming languages based on SLD-resolution.

Part II studies three modern computation paradigms that can be seen as the foundation of three well-known programming styles: object-oriented, interaction-based, and concurrent programming, respectively. In addition, it includes a short discussion on emergent models of computation inspired by biological and physical processes. More precisely, Part II is organised as follows.

In Chapter 6, we analyse the process of computation from an object-oriented perspective: Computation is structured around objects that own a collection of functions (methods in the object-oriented terminology). We describe object-oriented computation models, providing examples and a comparison with traditional models of computation.

In Chapter 7, we study graphical models of computation, where computation is centred on the notion of interaction. Programs are collections of agents that interact to produce a result. We show that some graphical models naturally induce a notion of sequentiality, whereas others can be used to describe parallel functions.

Chapter 8 describes a calculus of communicating processes that can be used to specify concurrent computation systems, and gives a brief account of an alternative view of concurrency inspired by a chemical metaphor.

Chapter 9 gives a short introduction to some of the emergent models of computation: biologically inspired calculi and quantum computing.

The last chapter of the book (Chapter 10) contains answers to a selection of exercises.

At the end of the book there is a bibliographical section with references to articles and books where the interested reader can find more information.

Acknowledgements

The material presented in this book has been prepared using several different sources, including the references mentioned above and notes for my courses in London, Paris, and Montevideo. I would like to thank the reviewers, the editors, and the students in the Department of Computer Science at King's College London for their comments on previous versions of the book, and my family for their continuous support.

Maribel Fernández
London, November 2008

Contents

1. Introduction	1
1.1 Models of computation	3
1.2 Some non-computable functions	4
1.3 Further reading	7
1.4 Exercises	7

Part I. Traditional Models of Computation

2. Automata and Turing Machines	11
2.1 Formal languages and automata	12
2.2 Finite automata	13
2.2.1 Deterministic and non-deterministic automata	16
2.2.2 The power of finite automata	18
2.3 Push-down automata	20
2.4 Turing machines	23
2.4.1 Variants of Turing machines	28
2.4.2 The universal Turing machine	29
2.5 Imperative programming	29
2.6 Further reading	30
2.7 Exercises	31
3. The Lambda Calculus	33
3.1 λ -calculus: Syntax	34
3.2 Computation	38
3.2.1 Substitution	40

3.2.2	Normal forms	42
3.2.3	Properties of reductions	43
3.2.4	Reduction strategies	44
3.3	Arithmetic functions	45
3.4	Booleans	46
3.5	Recursion	48
3.6	Functional programming	49
3.7	Further reading	50
3.8	Exercises	50
4.	Recursive Functions	55
4.1	Primitive recursive functions	56
4.2	Partial recursive functions	61
4.3	Programming with functions	63
4.4	Further reading	67
4.5	Exercises	67
5.	Logic-Based Models of Computation	69
5.1	The Herbrand universe	69
5.2	Logic programs	70
5.2.1	Answers	73
5.3	Computing with logic programs	76
5.3.1	Unification	76
5.3.2	The Principle of Resolution	79
5.4	Prolog and the logic programming paradigm	84
5.5	Further reading	86
5.6	Exercises	86
Part II. Modern Models of Computation		
6.	Computing with Objects	93
6.1	Object calculus: Syntax	94
6.2	Reduction rules	96
6.3	Computation power	98
6.4	Object-oriented programming	99
6.5	Combining objects and functions	101
6.6	Further reading	104
6.7	Exercises	104

7. Interaction-Based Models of Computation	107
7.1 The paradigm of interaction	107
7.2 Numbers and arithmetic operations	111
7.3 Turing completeness	114
7.4 More examples: Lists	115
7.5 Combinators for interaction nets	117
7.6 Textual languages and strategies for interaction nets	118
7.6.1 A textual interaction calculus	120
7.6.2 Properties of the calculus	125
7.6.3 Normal forms and strategies	126
7.7 Extensions to model non-determinism	127
7.8 Further reading	129
7.9 Exercises	130
8. Concurrency	131
8.1 Specifying concurrent systems	134
8.2 Simulation and bisimulation	137
8.3 A language to write processes	139
8.4 A language for communicating processes	142
8.5 Another view of concurrency: The chemical metaphor	147
8.6 Further reading	147
8.7 Exercises	148
9. Emergent Models of Computation	151
9.1 Bio-computing	152
9.1.1 Membrane calculi	152
9.1.2 Protein interaction calculi	153
9.2 Quantum computing	154
9.3 Further reading	157
10. Answers to Selected Exercises	159
Bibliography	179
Index	183