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

Springer

Dr. Maribel Fernández
King's College London
UK

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

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