
Preface

What is the theory of computation all about? The theory of computation embodies the principle by which computers have become the basis of modern digital technology, which makes a computer perform as desired, and which, consequently, has led to the prosperity of our advanced information society (just as physical sciences constitute the principle by which physical phenomena are elucidated, thereby bringing all the prosperity of our material civilization). Throughout this book, the term “computation” points to all tasks that are performed by a computer.

A computer not only performs basic arithmetic operations; it tells us what the weather is going to be, and it can inform us of the most popular product by extracting useful information from enormous amounts of customer purchasing data. The IBM computer *Deep Blue* has even defeated a grand master chess player. In this book, we describe how a computer performs these tasks in principle, and we also clarify the limits on what computers can do. We think of what a computer executes as a problem, which is defined as a correspondence relation between input and output data. When we try to forecast weather or to compute the most popular product based on available data, there might be cases in which the information is not sufficient to accomplish these tasks. In such cases, we need to somehow infer a conclusion based on insufficient data, but this type of discussion will be omitted from this book.

We will take up the game of chess to explain what computation time is. Given an arrangement of pieces on one side of a chessboard, to decide the best move, one must trace all possible sequences of moves until the game is completed. To do so, all possible moves must first be enumerated; then all possible countermoves of an opponent for each move of the player must be envisioned. Those processes must be repeated until the end of the game. Tracing all the possible moves in this way, the first player can choose the best move based on all the moves traced. In principle, a program that plays chess executes those tasks. But it is impossible in practice because it takes too much time, even for a computer, to read ahead all the moves to the end of the game. If even a supercomputer takes, say, 1,000,000 years to compute the next move, it is of no use. So, to beat a grand master of chess, it becomes crucial for a chess program to render a sufficiently appropriate, but perhaps not perfect, judgment about the next move based on some proper criterion of judgment.

This book consists of four parts. In Part IV, taking the computation time into account, we divide computable problems into those that are feasibly computable in a reasonable time and those that are considered to be intractable, requiring an inordinate amount of time. We describe the structure of problems of the latter group based

on their mutual relations. There even exist problems of correspondence between inputs and output that cannot be computed no matter how much time we spend. In Part III, we explain these problems that are incomputable in principle.

To develop the arguments presented in Parts III and IV, we are required to give a model of computing machines by removing many functionalities from computers, retaining only the indispensable fundamental functions. A computer that confronts a chess opponent proceeds through its computations almost identically as a human being might look ahead one move, and study arrangements of pieces one after another using a paper, a pencil, and an eraser. However, to develop the argument strictly, we must express the set of tasks which can be executed by a computer in one step as a clearer and more concise representation than that which can be made using a paper, a pencil, and an eraser. The expression given in this way turns out to be a computation model. In Part III, we introduce a Turing machine as a computational model of a computer; in Part II, we mention other computational models with restrictions imposed on the Turing machine. Part I provides an introduction to and a preparation for the theory of computation.

A computer executes whatever commands we give it, provided that they can be written as a program. So it seems that there exist no underlying laws that govern what takes place in a computer. But in fact such laws do exist. As described above, we can classify computable problems as those that can and cannot be computed feasibly in a reasonable time. The characteristic features distinguishing these two groups lurk in the problems around the boundary separating these groups. Moreover, there are problems that cannot be computed no matter how much time is allocated to compute them. Inherent in such problems are the aspects that make these problems unsolvable. By “exposing computation to extreme constraints,” we can see that the computation reveals its substantial intrinsic nature. This is just like the situation where phenomena in the extremely high-speed world cannot be explained using Newtonian dynamics, thereby beckoning the principle of relativity. In this book, we study substantial mechanisms and underlying principles of computation that can be revealed by examining computational difficulties posed by unsolvable problems and by exploiting those impediments that feasible computation must overcome.

In Parts II, III, and IV of this book, we deal respectively with the topics of automata and language theory, computability theory, and complexity theory. We select the important primary results from these fields together with their proofs, emphasizing the intuition behind them. These results are rich and beautiful. However, those who find these topics fascinating might be limited to a handful of sophisticated researchers of related fields. To improve the situation, in this book we present materials from the theory of computation without omitting any important points of the arguments so that students and engineers just beginning to study the topics for the first time can capture essential aspects intuitively. Researchers in this field often enjoy discussing research topics with each other because they can capture the results along with the genuine intuition behind them. Ideas are often exchanged with intuitive expressions such as “A calls B to make work,” “A tries to check exhaustively,” “A recognizes it as a proof,” and so on. The researchers can communicate through these expressions because their meaning is shared among them.

The textbooks of the field often seem to be produced for students who are supposed to become researchers in the future. The author himself was taught in his school days that he had to read books deeply, i.e., to “read between the lines.” But now, the achievements in the field should be open to not only specialized researchers, but to a larger number of people outside the field. With that in mind, this book is written so that even readers who are learning about this field for the first time can understand material intuitively and thoroughly, saying (we hope) “Ah yes! I’ve got it now!” by virtue of viewing many figures, examining examples, and developing arguments, just as if they were attending a lecture in the classroom. The author, however, is not quite sure whether or not he has written this book as it was supposed to be; it is the readers who ultimately judge whether or not the original goal is attained. I would like to encourage the readers to practice the numerous examples and exercises throughout the book and to examine the figures carefully because the illustrative ideas might not be obtained otherwise. The readers will come to have a deep understanding of modern computers, including their limitations, if they read through this book and capture, intuitively, the principles that govern the computation by studying the examples and solving the exercises.

I hope that the readers take enjoyment in this book and thus equip themselves for life in our information society.

Sendai, Japan

Akira Maruoka



<http://www.springer.com/978-0-85729-534-7>

Concise Guide to Computation Theory

Maruoka, A.

2011, XVII, 281 p., Hardcover

ISBN: 978-0-85729-534-7