Ronald V Book

Ker-I Ko

Stephen R Mahaney

Kenneth McAloon

Studies in Complexity Theory

Ronald V Book (Editor)
University of California at Santa Barbara

Ker-I Ko
University of Houston

Stephen R Mahaney
AT&T Bell Laboratories

Kenneth McAloon
City University of New York

Studies in Complexity Theory

Pitman, London

John Wiley & Sons, Inc., New York, Toronto

PITMAN PUBLISHING LIMITED
128 Long Acre, London WC2E 9AN

A Longman Group Company

First published 1986

Available in the Western Hemisphere from
John Wiley & Sons, Inc.
605 Third Avenue, New York, NY 10158

British Library Cataloguing in Publication Data

Studies in complexity theory.—(Research notes
in theoretical computer science, ISSN 0286-7534)
1. Computational complexity
I. Book, Ronald V. II. Ker-I
III. Mahaney, Stephen R. IV. McAloon, Kenneth
V. Series
519.4 QA297
ISBN 0-273-08755-X (Pitman)
ISBN 0-470-20293 9 (Wiley)

Reproduced and printed by photolithography
in Great Britain by Biddles Ltd, Guildford

Foreword

Research in theoretical computer science has experienced tremendous growth both in the depth to which older theories have been pursued and also in the number of new problem areas that have arisen. While theoretical computer science is mathematical in nature, its goals include the development of an understanding of the nature of computation as well as the solution of specific problems that arise in the practice of computing.

The purpose of this series of monographs is to make available to the professional community expositions of topics that play an important role in theoretical computer science or that provide bridges with other aspects of computer science and with aspects of mathematics. The scope of the series may be considered to be that represented by the leading journals in the field. The editors intend that the scope will expand as the field grows and welcome submissions from all of those interested in theoretical computer science.

Ronald V. Book
Main Editor

Preface

Computational complexity theory has been at the leading edge of research in theoretical computer science for the past twenty years. Since Stephen Cook's discovery of NP-complete problems in 1971 and Richard Karp's application of this notion to combinatorial problems in 1972, there has been a continuing surge of activity in this area. It is quite clear that complexity theory will play an important role in theoretical computer science in the future.

Beyond the design and analysis of algorithms for specific combinatorial problems, the primary goal of researchers in this field is to develop a quantitative theory of complexity. Such a theory must provide general results on different measures of computational difficulty (say, time and space), different models of computation (say, sequential and parallel), and different notions of mode of computation (say, the deterministic, non-deterministic, alternating, and probabilistic modes).

It is clear that the P =? NP problem and the open problems that are related to it are fundamentally important in the development of any such quantitative theory of computational complexity. Furthermore, these problems play an important role in considering applications of complexity theory to other areas of mathematics and computer science.

Three aspects of computational complexity theory are described in this volume. The first paper, by Ker-I Ko, describes how the techniques of discrete complexity theory can be used in the study of numerical computation. Professor Ko shows how concepts developed in the study of the P =? NP and related problems can be used as a basis for developing a complexity theory for numerical computation. The second, by Stephen Mahaney, surveys some recent results in the development of a structure theory for complexity classes. In particular,

Dr. Mahaney examines the possibility of sparse sets being complete sets and considers the consequences for the P =? NP and related problems. The third, by Kenneth McAloon, sketches relationships between problems in logic--more specifically, the study of models of arithmetic--and problems in complexity theory. Professor McAloon describes recent research on initial segments of models of arithmetic and shows how certain "bridging theorems" can be used to relate this work to problems related to the P =? NP problem.

These three papers provide insight into aspects of computational complexity theory. It is expected that they can be read as part of advanced lecture courses and seminars as well as by individual researchers.

These papers are based on talks given by the authors at a conference on computational complexity theory held at the University of California at Santa Barbara in March 1983. The conference was supported by the College of Letters and Science at U.C.S.B. and by the National Science Foundation under Grant MCS82-15544.

The preparation of this volume was supported in part by the National Science Foundation under Grant DCR83-12472.

The Editor gratefully acknowledges the efforts of Ms. Leslie Wilson in preparing the entire manuscript for publication.

Contents

KER-I KO

Applying techniques of discrete complexity theory to numerical computation

1. INTRODUCTION

The study of formal computational theories of real analysis can be traced back to Turing's pioneering work [91], in which he gave, based on the Turing machine (TM) model, a simple definition of computable real numbers. More studies on the definition of computable real numbers have been done by [85,64,73, 72]. In the 1950's a formal theory of computational analysis, called recursive analysis, was developed in the setting of recursive function theory. In this theory, a computable real function is defined as a recursive functional operating on the Cauchy sequence representations of real numbers [22,23].[1] Many interesting results about the constructive versions of theorems in classical real analysis have been obtained (see [69,70] for a summary).

While recursive analysis is useful in studying the computability problems in the theory of real analysis, it has not been applied to the study of computational complexity of numerical algorithms. Instead, the floating-point model of real computation is often used in the analysis of numerical algorithms, mainly because it behaves closer to the computation in actual digital computers. However, as it is well known (see, for example, [34]), this simplistic model of real computation does not reflect correctly the structure of real computation,

and thus cannot be used as a model for a general, mathematical complexity theory. For example, it is difficult to prove, in this model, negative results about the complexity of a numerical problem. In other words, there is a big gap between the theory of recursive analysis and the theory of numerical analysis.

On the other hand, in the discrete theory of computation, such a gap does not exist because of the successful development of an interesting polynomial complexity theory [13,15,31,27,18]. This polynomial complexity theory, especially the NP theory, not only provides powerful tools for classifying the inherent complexity of natural problems, but also reveals interesting mathematical structures of polynomial computation. In addition, this theory has also been widely applied to many subareas of mathematics and computer science [18]. Thus, it is natural to try to extend the polynomial complexity theory to real computation to complement the theories of recursive analysis and numerical analysis. Indeed, such attempts have been made even before the NP theory exists [56,57].

In this paper, we give an overview of a polynomial complexity theory of numerical computation, which was developed in the past five years. Although the computational model and the general setting in this theory follow those of recursive analysis, we use extensively the concepts and techniques of discrete polynomial complexity theory. The main issues to be discussed include the classification of the complexity of basic numerical operations, the relationship among analytical

properties, structural properties and complexity properties of real functions, and the relationship between the structures of continuous and discrete objects.

In Section 2, we discuss how to design a general computational model for numerical problems and how to apply the concepts of discrete complexity theory to this model to define computational complexity of real functions. In Section 3, the computational complexity of numerical operations such as roots, integrals and derivatives is discussed. We apply the concepts of the NP theory to these problems and derive lower bounds for these operations which are otherwise difficult to prove in classical numerical analysis. In Section 4, the structure of real numbers and real functions is examined in the context of discrete complexity theory. It is shown that the Dedekind cut representations of real numbers and sparse sets have similar structural properties. From these structural properties, a partial classification of the complexity of NP left cuts in the low hierarchy in NP is given. In Section 4.3 we discuss the relative complexity of real numbers and compare the structures of recursive real functions with various reducibilities on real numbers. Open questions and future research directions are discussed in the final section.

2. MACHINE MODELS AND COMPLEXITY OF REAL FUNCTIONS

In this section we describe our computational model for numerical problems and give the definition of computational complexity of real functions in this model. Our model is an infinite-precision discrete model. It is different from the

infinite-precision continuous model, such as the one used by Borodin and Munro [10] in the analysis of complexity of algebraic computation, in which the cost of an arithmetic operation is only one unit. We accept the limit of discrete processing in actual digital computers and will use bit-operation measure of complexity. Our model is also different from the finite-precision discrete model such as the floating-point model because we require a good algorithm to be able to produce outputs with arbitrarily small, a priori bounds. A numerical algorithm in our model works as follows. It accepts, as the first input, a user-specified output precision and, as the second input, an approximate value to a real-valued problem instance with high precision, and it outputs an approximate value to the real-valued solution with the predefined precision. The complexity of such an algorithm is measured as a function of both the size of the problem instance and the output precision.

2.1 Computational Complexity of Real Numbers

In order to discuss computational complexity of real numbers, it is necessary to consider first the representations of real numbers. The most general representations of real numbers include the Cauchy sequence representation, the Dedekind cut representation and the binary (or, decimal) expansion representation. The three representations thus provide three natural definitions of computable real numbers. Let N be the set of all non-negative integers, Z the set of all integers, Q the set of all rationals and R the set of all reals.

Definition 2.1(a). A real number x is computable if there is a recursive function $\phi : N \longrightarrow Q$, and a recursive function $\psi : N \longrightarrow N$, such that for all $m, n \in N$, $|\phi(m) - x| \leq 2^{-n}$ whenever $m \geq \psi(n)$.

Definition 2.1(b). A real number x is computable if the set $\{r \in Q : r \leq x\}$ is a recursive set of rationals.

Definition 2.1(c). A real number x is computable if there exists a recursive function $\phi : N \longrightarrow N$ such that (i) $\phi(n) \in \{0,1\}$ for all $n \geq 1$ and (ii) $x = \phi(0) + \sum_{n=1}^{\infty} \phi(n) \cdot 2^{-n}$.

Robinson [73] was the first to point out that the above three definitions are equivalent. On the other hand, Specker [85], Péter [64] and Mostowski [61] noticed that the subrecursive classes of real numbers defined by the three representations are in general not equivalent, and the Cauchy sequence representation appears to be the most comprehensive one. In the following we give more precise definitions of these subrecursive classes of real numbers and discuss their relations.

First, we need some notation. Let D be the set of dyadic rational numbers, i.e., the set of all rational numbers which have finite binary representations. In other words, $D = \{m/2^n : m \in Z, n \in N\}$. For each $n \in N$, D_n is the set of all dyadic rational numbers with $\leq n$ bits in the fraction parts of their binary representations; i.e., $D_n = \{m/2^n : m \in Z\}$. Dyadic rationals will be represented by finite binary strings (with a decimal point). Let $S = (+|-)\ (0|1)^*.(0|1)^*$.

We may define a mapping $\iota : S \longrightarrow D$ by

$$\iota(\pm d_n \ldots d_1 d_0 . e_1 \ldots e_m) = \pm\left(\sum_{i=0}^{n} d_i \cdot 2^i + \sum_{j=1}^{m} e_j \cdot 2^{-j} \right).$$

Note that each dyadic rational d in D has infinitely many string representations, each with a different length. Thus, it appears more convenient to use the string representations of d directly in the following discussion. That is, we will write $s \in S$ to denote both the string s and the dyadic rational number $\iota(s)$. For any string $s \in S$, we write $\ell th(s)$ to denote the length of s.

The set D of dyadic rationals is a dense subset of R with finite representations. Thus, it can replace the set Q of rationals and be used as a basis of a new notation system. We formulate the following representations of real numbers using this notation system.

(a) Cauchy sequence representation. A function $\phi : N \longrightarrow S$ is said to binary converge to a real number x if for each $n \in N$, $\phi(n) \in D_n$ and $|\phi(n) - x| \leq 2^{-n}$. Then, for each real number x there are infinitely many functions binary converging to x. We let $CS(x)$ be the set of all these functions and, for each set C of real numbers, $CS(C) = \cup_{x \in C} CS(x)$.

(a') Set representation of Cauchy sequences. Sometimes it is convenient to have a set representation instead of a sequence or a function representation of a real number. We use the

"projection" of the representation (a). For each $\phi \in CS(R)$, we define $L_\phi = \{s \in S : s \leq \phi(n)$, where n is the length of the fraction part of $s\}$. We call L_ϕ a (general) left cut of x if $\phi \in CS(x)$, and write $LC(x)$ to denote the set $\{L_\phi : \phi \in CS(x)\}$ and $LC(C)$ to denote $\{L_\phi : \phi \in CS(C)\}$ for any set C of real numbers.

(b) Standard left cut representation. We define $L_x = \{s \in S : s \leq x\}$. Let $\phi(n)$ be the string of the integral part of x plus the decimal point plus the first n bits of the fraction part of the binary expansion of x. Then $\phi \in CS(x)$ and $L_\phi = L_x$. That is, L_x is a special left cut representation in $LC(x)$. The main difference between the standard left cut L_x and an arbitrary left cut L_ϕ of x is that L_x satisfies a pleasant property that for any string s and t in S if $\iota(s) = \iota(t)$ then $s \in L_x \Longleftrightarrow t \in L_x$, but this property does not necessarily hold for arbitrary $L_\phi \in LC(x)$. For example, let $x = 1/3 = +.010101 \ldots$. The function ϕ defined by $\phi(2n) = +.01 \ldots 01$ (n 01's) and $\phi(2n+1) = +.01 \ldots 011$ (n 01's and one 1) is in $CS(x)$ and hence $L_\phi \in LC(x)$. We note that $+.011 = \phi(3) \in L_\phi$ but $+.0110 \notin L_\phi$ because $+.0110 > +.0101 = \phi(4)$.

(c) Binary expansion representation. Define $B_x = \{n : n > 0$, the n^{th} bit of the fraction part of the binary expansion of x is 1$\}$. Then $x = I_x \pm \sum_{n \in B_x} 2^{-n}$, where I_x is the integral part of x.

Now we may define various subrecursive classes of real numbers based on these representations. For example, $\{x \in R :$ there is a recursive function $\phi \in CS(x)\}$ is exactly the class of computable real numbers defined by Definition 2.1(a), and is equal to $\{x \in R : L_x$ is recursive$\}$ and $\{x \in R : B_x$ is recursive$\}$. However, for primitive recursive real numbers, these definitions are no longer equivalent. That is, $\{x \in R : L_x$ is primitive recursive$\} = \{x \in R : B_x$ is primitive recursive$\} \neq \{x \in R :$ there is a primitive recursive $\phi \in CS(x)\}$ [85,39]. In fact, this nonequivalence of the three definitions is a general phenomenon in other similarly defined complexity classes of real numbers. For example, $\{x \in R : L_x$ is polynomial-time computable$\} = \{x \in R : B_x$ is polynomial-time computable$\} \neq \{x \in R :$ there is a $\phi \in CS(x)$ which is polynomial-time computable$\}$ where inputs to ϕ and B_x are written in unary notation [39]. It is generally accepted that, among the three, the Cauchy sequence representation is the most general one ([23,61,39]; also see the remark following Definition 2.5, and [94]).

In the following we will define the computational complexity of real numbers using only the Cauchy sequence representation. We use the time and space complexity of Turing machines [25,26] to define the time and space complexity of real numbers.

Definition 2.2 [39]. Let $T : N \rightarrow N$ be an integer function. We say that the time (space) complexity of a real number x is bounded by T if there is a Turing machine M such that the function ϕ computed by M (i.e., $\phi(n) =$ the output of M

on n) binary converges to x, and M halts on input n in $\leq T(n)$ moves (using $\leq T(n)$ cells, respectively).

Intuitively, a real number x has time complexity bounded by T if there is an effective method of finding a dyadic rational d approximating x to within an error 2^{-n} in T(n) steps, or, equivalently, if there is a function $\phi \in CS(x)$ such that its time complexity is bounded by T (where the inputs to ϕ are written in unary notation).

Now we define some interesting complexity classes of real numbers.

Definition 2.3

$P_R = \{x \in R$: the time complexity of x is bounded by a polynomial function$\}$.

$PSPACE_R = \{x \in R$: the space complexity of x is bounded by a polynomial function$\}$.

A real number x is said to be polynomial time (space) computable if $x \in P_R$ ($x \in PSPACE_R$, respectively).

The class P_R contains all rational numbers, algebraic numbers (Theorem 3.2) and some well known transcendental numbers such as e and π. P_R is a real closed field [39].