A Tutorial on Computable Analysis

Vasco Brattka[1], Peter Hertling[2], and Klaus Weihrauch[3]

[1] Laboratory of Foundational Aspects of Computer Science, Department of Mathematics and Applied Mathematics, University of Cape Town, Rondebosch 7701, South Africa
Vasco.Brattka@uct.ac.za
[2] Institut für Theoretische Informatik und Mathematik, Fakultät für Informatik, Universität der Bundeswehr München, 85577 Neubiberg, Germany
Peter.Hertling@unibw.de
[3] Faculty of Mathematics and Computer Science, University of Hagen, 58084 Hagen, Germany
Klaus.Weihrauch@fernuni-hagen.de

Summary. This tutorial gives a brief introduction to computable analysis. The objective of this theory is to study algorithmic aspects of real numbers, real number functions, subsets of real numbers, and higher type operators over the real numbers. In this theory, the classical computability notions and complexity notions based on the Turing machine model and studied in computability theory and computational complexity theory are applied to computational problems involving real numbers.

1 Introduction

This tutorial gives a brief introduction to computable analysis. In computable analysis, computational problems over the real numbers are studied from the point of view of computability theory and computational complexity theory.

A large part of today's computing power all over the world is spent on computational problems that can best be modeled as computational problems involving real numbers. These are mostly numerical problems but also geometric problems and problems studied in other fields, e.g., in the theory of neural networks. Therefore, a theory is needed that tells which computational problems over the real numbers can be solved on a digital computer and how much time or how much computer memory this will take. Computable analysis studies these questions on the basis of the notions studied in computability theory and computational complexity theory that are defined via the Turing machine model.

Computable analysis is based on the one hand on analysis and numerical analysis and on the other hand on computability theory and computational complexity

theory. Other mathematical or computer science disciplines that are closely related to computable analysis are as follows:

- Constructive analysis. There are several schools and varieties of constructive analysis. We give the following references: Bishop and Bridges [8], Bridges and Richman [27], Troelstra and van Dalen [100, 101], Šanin [85], and Kušner [59].
- Domain theory. Domains are used for defining semantics of programming languages; see, e.g., Stoltenberg-Hansen et al. [98]. In our context this is especially interesting for programming languages over the real numbers. One can also define computability notions over the real numbers via domains; see, for instance, Edalat, Escardo, and Blanck [38, 37, 9].
- Mathematical logic. In fact, computability theory, constructive analysis, and domain theory can all be considered as subdisciplines of logic.
- Interval analysis; see, e.g., Moore [69]. Note that intervals of real numbers form a domain.
- Algebraic complexity theory over the real numbers, as studied by Blum et al. [10].
- Information-based complexity theory, a theory that studies mainly the computational complexity of problems involving function spaces over the real numbers in an algebraic computation model; see Traub et al. [99].

Since real numbers and many other objects studied in analysis are "infinite" objects containing an "infinite amount of information," one has to approximate them by "finite" objects containing only a "finite amount of information" and to perform the actual computations on these finite objects. This leads to the following three levels of study of computational problems in analysis:

- Topology: how can one approximate infinite objects?
- Computability theory: how can one compute with infinite objects?
- Computational complexity theory: how can one compute efficiently with infinite objects?

In this tutorial we will introduce the basic notions of computable analysis and present selected results that are supposed to illustrate important aspects and ideas of computable analysis and to give an overview of current areas of research in computable analysis. The main focus of this article is on approximation and computability. Because of limited space, many important or interesting results, for example, from complexity theory, are not mentioned, and the bibliography is far from being complete. We apologize to all whose work is insufficiently, or not, mentioned. A bibliography on computable analysis can be found through the web pages of the CCA (= Computability and Complexity in Analysis) network: `http://cca-net.de`. We will mostly use the approach via representations, which was developed by Hauck

(see, e.g., [42, 43]), and by Kreitz and Weihrauch [58]. A more detailed presentation of computable analysis based on representations is the textbook by the third author [109]. Another approach to computable analysis can be found in the textbook by Pour-El and Richards [80]: via sequential computability and effective uniform continuity of functions and via computability structures. This approach has been generalized from normed spaces to metric spaces by Yasugi et al. [114]. The textbook by Ko [55] covers a large part of the complexity theoretic results in computable analysis and is based on the notion of an oracle Turing machine; see the last two sections. The paper [25] by Braverman and Cook is a short introduction into important basic notions and ideas from computable analysis. This tutorial is based mostly on the slides of the tutorial on computable analysis given by the first author at the conference "Computability in Europe 2005" in June/July 2005 in Amsterdam, the Netherlands. It should be readable by anyone with a basic knowledge in computability theory and analysis and related fields. Here is the table of contents of the paper:

1. Introduction
2. Preliminaries
3. Computable Real Numbers
4. Computable Functions
5. Computability Notions for Subsets of Euclidean Space
6. Representations and Topological Considerations
7. Solvability of Some Problems Involving Sets and Functions
8. Computability of Linear Operators
9. Degrees of Unsolvability
10. Computational Complexity of Real Numbers and Real Number Functions
11. Computational Complexity of Sets and Operators over the Real Numbers

2 Preliminaries

First we introduce some notation. By $\mathbb{N}$ we denote the set of natural numbers, i.e., $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$, by $\mathbb{Q}$ the set of rational numbers, by $\mathbb{R}$ the set of real numbers, and by $\mathbb{C}$ the set of complex numbers. By $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ we denote the Euclidean distance on $\mathbb{R}^n$. Any finite set containing at least two elements is called an *alphabet*. Usually we denote alphabets by uppercase Greek letters, e.g., Σ, Γ. Furthermore, unless stated otherwise, we assume that 0 and 1 are elements of Σ. If Σ is an alphabet, then Σ^* denotes the set of all finite strings over Σ, $\Sigma^\omega = \{p \mid p : \mathbb{N} \to \Sigma\}$ the set of all one-way infinite sequences over Σ, and λ denotes the empty word. The i-th component of a one-way infinite sequence p is written $p(i)$ or p_i. If X and Y are

sets, then $f :\subseteq X \to Y$ denotes a function whose domain $\mathrm{dom}(f)$ is contained in X and whose range $\mathrm{range}(f)$ is contained in Y. If $\mathrm{dom}(f) = X$, then we call f *total* and may write $f : X \to Y$. A *sequence* in X is simply a total function $x : \mathbb{N} \to X$ and is often written as $(x_n)_{n\in\mathbb{N}}$.

We assume that the reader is familiar with the Turing machine model. Via the Turing machine model one defines computability of functions on Σ^*. We also consider computability notions on $\mathbb{N}^n$ and $\mathbb{Q}^m$ for $n, m \in \mathbb{N} \setminus \{0\}$ as basic. For completeness sake and in order to avoid misunderstandings, we formally introduce computability of functions on or between these sets via the Turing machine model. In order to do that we have to represent elements of $\mathbb{N}^n$ and of $\mathbb{Q}^m$ by strings. Let $\nu_{\mathbb{N}^1} := \nu_{\mathbb{N}} :\subseteq \Sigma^* \to \mathbb{N}$ be the usual binary notation of natural numbers. Using the standard bijection $\langle\cdot,\cdot\rangle : \mathbb{N}^2 \to \mathbb{N}$ defined by $\langle x, y\rangle := \frac{(x+y)\cdot(x+y+1)}{2} + y$ and the derived bijections $\langle\ldots\rangle : \mathbb{N}^{k+1} \to \mathbb{N}$ defined by $\langle x_1, \ldots, x_k, x_{k+1}\rangle := \langle\langle x_1, \ldots, x_k\rangle, x_{k+1}\rangle$ for $k \geq 2$, we define notations $\nu_{\mathbb{N}^n} :\subseteq \Sigma^* \to \mathbb{N}^n$ for $n \geq 2$ by

$$\nu_{\mathbb{N}^n}(w) := (x_1, \ldots, x_n) :\iff \nu_{\mathbb{N}}(w) = \langle x_1, \ldots, x_n\rangle.$$

Using the surjection $\nu_Q : \mathbb{N} \to \mathbb{Q}$ defined by $\nu_Q(\langle i, j, k\rangle) := \frac{i-j}{k+1}$, we define notations $\nu_{\mathbb{Q}^n} :\subseteq \Sigma^* \to \mathbb{Q}^n$ for $n \geq 1$ (and use $\nu_{\mathbb{Q}} := \nu_{\mathbb{Q}^1}$) by

$$\nu_{\mathbb{Q}^n}(w) := (q_1, \ldots, q_n) :\iff q_i = \nu_Q(\nu_{\mathbb{N}^n}(w)_i) \text{ for } i = 1, \ldots, n.$$

Finally, for any alphabet Σ^* we define $\nu_{\Sigma^*} := \mathrm{id}_{\Sigma^*}$. Let now X and Y by any of the spaces Σ^*, $\mathbb{N}^n$, and $\mathbb{Q}^m$ for some $n, m \geq 1$. A function $f :\subseteq X \to Y$ is called *computable* if there is a Turing machine that, on input $v \in \Sigma^*$, never stops if $v \notin \mathrm{dom}(f\nu_X)$, and that stops after finitely many steps with some output w satisfying $\nu_Y(w) = f\nu_X(v)$ if $v \in \mathrm{dom}(f)$. We will also use the notions "decidable" and "computably enumerable" for subsets of Σ^*, $\mathbb{N}^n$, and $\mathbb{Q}^m$ in the usual sense; see Section 5 for definitions.

3 Computable Real Numbers

For a long time, mathematics has been concerned with computation problems that would nowadays be considered as computation problems over the real numbers. But only in the twentieth century, a mathematical theory of algorithms, of computability, and of complexity over the real numbers has been developed. Quite early in the twentieth century, constructive mathematics was born. Results in constructive mathematics, e.g., by Brouwer [28, 29] and by Bishop and Bridges [8], certainly have algorithmic content. But these constructive mathematical theories are based on certain logical principles, not on a formal computation model. The so-called "Russian school of constructive analysis," founded by Markov (see Šanin [85] or Kušner [59]) is closer to our approach. We will come back to it in Section 6. Perhaps the real starting point of computable analysis was the landmark paper "On computable numbers,

with an application to the Entscheidungsproblem" by Turing [102]. In this paper, Turing asked which real numbers should be considered as computable and, in order to answer this question, developed the theoretical computer model that subsequently was called "Turing machine model" and has become the standard computation model in computability theory and computational complexity theory. Turing defined the computable real numbers as those real numbers that have a binary expansion that can be computed by a Turing machine. We will formulate several equivalent conditions.

Definition 3.1. A real number x is *computable* if it satisfies one (and then all) of the conditions in the following theorem.

Theorem 3.2. *For $x \in \mathbb{R}$, the following conditions are equivalent:*

1. *There exists a Turing machine that outputs a binary expansion of x (without input and without ever stopping).*
2. *There exists a computable sequence of rational numbers $(q_n)_{n\in\mathbb{N}}$ that converges rapidly to x; i.e., $|x - q_i| < 2^{-i}$ for all i.*
3. *There exists a computable sequence of rational shrinking intervals enclosing only x; i.e., there exist two computable sequences $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ of rational numbers with $a_0 < \cdots < a_n < a_{n+1} < \cdots x \cdots < b_{n+1} < b_n < \cdots < b_0$ and with $\lim_{n\to\infty} a_n = x = \lim_{n\to\infty} b_n$.*
4. *$\{q \in \mathbb{Q} \mid q < x\}$ is a decidable set of rational numbers.*
5. *There exist an integer $z \in \mathbb{Z}$ and a decidable set $A \subseteq \mathbb{N}$ such that $x = z + x_A$, where $x_A := \sum_{i\in A} 2^{-i-1}$.*
6. *x is rational (then it has a finite continued fraction expansion) or x has a computable infinite continued fraction expansion, i.e., there exist an integer $z \in \mathbb{Z}$ and a computable function $f : \mathbb{N} \to \mathbb{N} \setminus \{0\}$ such that*

$$x = z + \cfrac{1}{f(0) + \cfrac{1}{f(1) + \cfrac{1}{f(2) + \cdots}}}.$$

Proof. We prove only that the first two notions are equivalent.

"1 $\Rightarrow$ 2": Let us assume that x has a binary expansion that can be computed by a Turing machine. Then let q_n be the rational number that one obtains when in this binary expansion one replaces all digits from the $(n+2)$-th digit after the binary point on by zeros. The sequence $(q_n)_{n\in\mathbb{N}}$ is a computable sequence of rational numbers and converges rapidly to x.

"2 $\Rightarrow$ 1": If x happens to be of the form: integer divided by a power of two, then it has a binary expansion in which only finitely many digits are different from zero. Obviously, such a binary expansion can be computed by a Turing machine. Now,

let us assume that x is not of this form. Then x has a uniquely determined binary expansion. Let $(q_n)_{n\in\mathbb{N}}$ be a computable sequence of rational numbers converging rapidly to x. We wish to show that one can compute the binary expansion of x. First, let us determine the part of the binary expansion of x in front of the binary point. Since x is not an integer, there is an i such that the interval $[q_i - 2^{-i}, q_i + 2^{-i}]$ does not contain an integer. By computing sufficiently many q_i for $i = 0, 1, 2, \ldots$ we will find such an i. Then the part of the binary expansion of q_i in front of the binary point is equal to the part of the binary expansion of x in front of the binary point. Now, let us assume that we have computed the part of the binary expansion of x in front of the binary point and, for some $n \in \mathbb{N}$, also the first n digits after the binary point in the binary expansion of x. We can determine whether the next digit is a 0 or a 1 in a similar way as we determined the part of the binary expansion of x in front of the binary point: since x is not of the form: integer divided by 2^{n+1}, there is an i such that the interval $[q_i - 2^{-i}, q_i + 2^{-i}]$ does not contain an integer divided by 2^{n+1}. Then the binary expansions of x and of q_i are identical up to the $(n+1)$-th digit after the binary point. □

The argument just given works as well for the decimal representation or the representation with base $b \geq 2$ instead of the binary representation. Thus we have for any base $b \geq 2$: a real number is computable if, and only if, its has a computable base b expansion.

Remark 3.3. Note that it is easy to go from a computable binary expansion of a real number x to a computable sequence of rational numbers converging rapidly to x. But in the proof of the other direction we made a case distinction that is not a result of a computation. Furthermore, the algorithm in the second case of this case distinction might ask for a very good approximation (with precision much higher than 2^{-i}) of x in order to compute the i-th digit of the binary expansion of x. This shows that the two underlying representations of real numbers used in the two conditions above are quite different. Indeed, Turing himself noticed in a correction [103] to the paper cited above [102] that the binary representation is fine for defining the class of computable real numbers but unsuitable for performing computations on real numbers. We will come back to this. In contrast, the representation of real numbers by rapidly converging sequences of rational numbers is also suitable for performing computations on real numbers, as we will see soon.

Theorem 3.4 (Rice [83]). *The set of computable real numbers $\mathbb{R}_c$ forms a real algebraically closed field.*

That means it is a subfield of the field $\mathbb{R}$ of real numbers, and it contains the real zeros of any polynomial $a_n x^n + \cdots + a_1 x + a_0$ whose coefficients $a_n, \ldots, a_1, a_0$ are computable real numbers.

Examples 3.5.
1. All rational numbers are computable real numbers. Even more, all real algebraic numbers are computable.
2. Also π and e are computable real numbers.

Remark 3.6. In the Russian school of constructive analysis, computability is defined for functions mapping computable real numbers to computable real numbers; see Šanin [85] or Kušner [59]. We shall mainly consider a different notion of computability for functions mapping real numbers to real numbers. In Section 4 we will explain the connection between these two notions and other notions of computability for real number functions.

We list a few more properties of the set of computable real numbers.

Proposition 3.7. *The set of computable real numbers is countably infinite.*

Proof. It is clear that it is infinite since it contains all rational numbers. And it is countable because for every computable real number, there exists a Turing machine computing its binary expansion, and there are only countably many Turing machines. □

Now it is natural to ask whether one can effectively list all computable real numbers. This turns out to be impossible. In order to make this statement precise we need the notion of a computable sequence of real numbers.

Definition 3.8. A sequence $(x_n)_{n\in\mathbb{N}}$ of real numbers is called *computable* if there exists a computable sequence $(q_k)_{k\in\mathbb{N}}$ of rational numbers such that $|x_n - q_{\langle n,i\rangle}| < 2^{-i}$ for all $n, i \in \mathbb{N}$.

Obviously, any member of a computable sequence of real numbers is itself a computable real number.

Proposition 3.9. *No computable sequence of real numbers contains all computable real numbers.*

The proof goes by diagonalization; see, e.g., Weihrauch [106, p. 486].

Since there are uncountably many real numbers, by Proposition 3.7, there are uncountably many noncomputable real numbers.

Proposition 3.10. *The set of computable real numbers is not complete.*

Proof. Any real number, also a noncomputable one, is the limit of a sequence of rational numbers, hence, the limit of a sequence of computable real numbers. □

Many noncomputable real numbers are an even limit of a computable sequence of reals. Nevertheless, the set of computable real numbers is "computably complete" in a certain sense, as we will see now.

Definition 3.11. 1. If $(r_n)_{n\in\mathbb{N}}$ is a convergent sequence of real numbers with limit x, and $m : \mathbb{N} \to \mathbb{N}$ is a function such that, for all $i, n \in \mathbb{N}$, if $i \geq m(n)$ then $|x - r_i| < 2^{-n}$, then we call m a *modulus of convergence* of the sequence $(r_n)_{n\in\mathbb{N}}$.

2. A sequence $(r_n)_{n\in\mathbb{N}}$ of real numbers *converges computably* if it has a computable modulus of convergence.

Proposition 3.12. *The limit of any computable sequence of real numbers that converges computably is again a computable real number.*

Proof. Let $(r_n)_{n\in\mathbb{N}}$ be a computable sequence of real numbers that converges computably. Let m be a computable modulus of convergence, and let $(q_k)_{k\in\mathbb{N}}$ be a computable sequence of rational numbers proving the computability of $(r_n)_{n\in\mathbb{N}}$ in the sense of Definition 3.8. Then the sequence $(p_n)_{n\in\mathbb{N}}$ defined by $p_n := q_{\langle m(n+1),n+1\rangle}$ is a computable sequence of rational numbers, and it converges rapidly to the limit of $(r_n)_{n\in\mathbb{N}}$. □

We end this section by an excursion into the class of noncomputable real numbers. Quite a lot of research has been going and is still going into various types of effectivity properties for real numbers that are more general than computability. We describe some of them.

Definition 3.13. A real number x is *left-computable* (often called *c.e.*) if it satisfies one and then all of the conditions in the following proposition. A real number x is *right-computable* if $-x$ is left-computable.

Proposition 3.14. *For a real number x, the following conditions are equivalent.*

1. *There exists a computable, strictly increasing sequence of rational numbers with limit x.*
2. *There exists a computable, nondecreasing sequence of rational numbers with limit x.*
3. *The set $\{q \in \mathbb{Q} \mid q < x\}$ is a c.e. subset of $\mathbb{Q}$.*

Further characterizations of left-computable real numbers have been given by Calude et al. [30, Theorem 4.1]. Right-computable real numbers can be characterized in a similar way. The following lemma is obvious.

Lemma 3.15. *A real number is computable if, and only if, it is left-computable and right-computable.*

Examples of left-computable but not computable real numbers can be constructed easily as follows.

Examples 3.16. 1. For $A \subseteq \mathbb{N}$, let

$$x_A := \sum_{i \in A} 2^{-i-1}.$$

This is a number in the interval $[0, 1]$. The number x_A is computable if, and only if, A is decidable. If A is c.e., then x_A is left-computable. Hence, if A is c.e. but not decidable, then x_A is left-computable but not computable (Specker [96]).

2. It is interesting that there are left-computable real numbers in $[0, 1]$ that are not of the form x_A with c.e. $A \subseteq \mathbb{N}$. Indeed, if $B \subseteq \mathbb{N}$ is c.e. but not decidable, then $B \oplus B^c := \{2n \mid n \in B\} \cup \{2n + 1 \mid n \notin B\}$ is not c.e., but $x_{B \oplus B^c}$ is nevertheless left-computable (Jockusch 1969, unpublished; see Soare [95]).

Besides the left-computable real numbers and among many others, the following classes of real numbers, defined by effectivity properties, have been studied:

- The *strongly c.e.* reals. These are the reals x_A with c.e. $A \subseteq \mathbb{N}$.
- The *weakly computable* reals; see Ambos-Spies et al. [2] and Zheng and Rettinger [116]. These are the differences of left-computable reals. These real numbers form a field, which is the arithmetical closure of the left-computable reals.
- The *computably approximable* reals; see, e.g., Ho [52] and Barmpalias [5]. These are the limits of computable and converging (but not necessarily computably converging) sequences of rational numbers. These real numbers form a field as well. This field is closed under application of computable real number functions (computable real number functions will be defined soon).
- The Ω *numbers* defined by Chaitin [33] as the halting probabilities of universal self-delimiting Turing machines. These real numbers can also be characterized as those real numbers in $[0, 1]$ that are left-computable and random, and also as those left-computable real numbers in $[0, 1]$ such that any computable, strictly increasing sequence of rational numbers converging to such a number converges as slowly as it is possible for a computable, strictly increasing sequence of rational numbers; see Calude et al. [30] and Kučera and Slaman [60].
- The *trivial* reals; see Downey et al. [36]. A real in $[0, 1]$ is *trivial* if the Kolmogorov complexity of the prefix of length l of its binary expansion differs at most by a constant from the Kolmogorov complexity of a string of l zeros (or, which is equivalent, from the Kolmogorov complexity of the prefix of length l of the binary expansion of some computable real number).

One can also define a hierarchy of real numbers corresponding to the arithmetical hierarchy of sets of natural numbers; see Zheng and Weihrauch [117]. For an overview of many classes of real numbers, defined by effectivity properties, the reader is referred to Zheng [115]. Effective randomness notions for real numbers or infinite binary sequences constitute an area of research of its own; the paper [66] by Miller and Nies contains a list of open problems in that area and many references.

Finally, for $n \geq 1$, a point $x \in \mathbb{R}^n$ is *computable*, if each of its components is a computable real number. Obviously, a point $x \in \mathbb{R}^n$ is computable if, and only if, there is a computable sequence $(q_i)_{i\in\mathbb{N}}$ of rational points $q_i \in \mathbb{Q}^n$ rapidly converging to x. Also Definitions 3.8 and 3.11 and Propositions 3.7, 3.9, 3.10, and 3.12 can be generalized directly to points in $\mathbb{R}^n$.

4 Computable Functions

Computable real numbers and the various kinds of weaker effectivity properties for real numbers are certainly interesting and worth studying. But in most computational problems over the real numbers, the task is not to compute some specific real number. Instead, there is some input, often one or several real numbers, and one wishes to compute some real number depending on this input. That is, given a vector x of real numbers, one wishes to compute the value $f(x)$ of some real number function f (of course, there are also more complicated types of computational problems; we will discuss them later on).

How can one compute a real number function $f :\subseteq \mathbb{R}^n \to \mathbb{R}$? We want to define a precise computability notion that describes exactly the real number functions that can be computed by a digital computer, except that at this moment we do not want to worry about time or space constraints (later we shall consider also the question of computational complexity). We use the Turing machine model as a model for digital computers.

What does it mean to compute a real number function $f :\subseteq \mathbb{R}^n \to \mathbb{R}$ in practice? Since a real number contains an infinite amount of information, it is unrealistic to expect that, given x, one can compute $f(x)$ exactly in finitely many steps. Rather, in numerical computations one aims at computing a "good" rational approximation of $f(x)$. We replace "good" by "arbitrarily good." Furthermore, the computer should not need x with infinite precision, but a "good" rational approximation should be sufficient. Again, we will allow the computer to ask for "arbitrarily good" rational approximations to x, i.e., approximations as good as it likes. Here is our first definition of computable real number functions.

Definition 4.1. A function $f :\subseteq \mathbb{R}^n \to \mathbb{R}$ is *computable* if there is an oracle Turing machine that, given any $k \in \mathbb{N}$, may ask for arbitrarily good rational approximations of the input $x \in \mathrm{dom}(f)$; i.e., it may ask finitely many questions of the kind "Give me a vector $p \in \mathbb{Q}^n$ of rational numbers with $d(x, p) < 1/2^i$," where the exponent i may depend on the answers to the previous questions, and after finitely many steps, it writes a rational number q on the output tape with $|f(x) - q| < 2^{-k}$.

Here, the precision i of a request as above has to be written in binary form on a special tape, the *oracle tape*, and the machine has to enter a special state, the *query state*. Then the answer will be provided in one step on the oracle tape, with the tape head on the field to the left of the answer.

That means the input $x \in \mathbb{R}^n$ is given to the machine only through rational approximations that the machine has to request. One can obviously modify this idea slightly without changing the computability notion by assuming that via some input tape(s) the machine has access to a rapidly converging rational sequence with limit x.

Once one has done that, one might as well change the model even more by not asking the machine to produce a rational approximation of precision 2^{-k} to $f(x)$, for any given $k \in \mathbb{N}$, but rather asking the machine to produce such approximations directly for all $k \in \mathbb{N}$, i.e., to ask the machine to produce a rapidly converging rational sequence with limit $f(x)$. This is slightly more elegant because in this notion the additional input k does not appear anymore, and input and output are of the same type (both are rapidly converging rational sequences).

Now we want to make this idea precise. We will directly formulate the basic definitions more generally since they will be useful for defining computability for many other kinds of objects than just real number functions.

Definition 4.2. A *representation* of a set X is a surjective function $\delta :\subseteq \Sigma^\omega \to X$, where Σ is some alphabet. Then for any $x \in X$ and any $p \in \Sigma^\omega$ with $\delta(p) = x$, the sequence p is called a δ*-name* of x.

Example 4.3. The usual decimal representation can be defined precisely so that it is a representation $\rho_{10} :\subseteq \Sigma^\omega \to \mathbb{R}$ in this sense, i.e., such that $\rho_{10}(1.414\ldots) = \sqrt{2}$, $\rho_{10}(3.141\ldots) = \pi$, $\rho_{10}(-0.999\ldots) = -1$. Similarly, one can define representations to any other base $b \in \mathbb{N}$ with $b \geq 2$.

Definition 4.4. The *Cauchy representation* $\rho :\subseteq \Sigma^\omega \to \mathbb{R}$ of the real numbers is a representation where a real number x is represented by a one-way infinite stream of symbols if this one-way infinite stream encodes a sequence of rational numbers converging rapidly to x:

$$\rho(w_0\#w_1\#w_2\#\ldots) = x :\iff |x - \nu_{\mathbb{Q}}(w_i)| < 2^{-i}, \text{ for all } i \in \mathbb{N}.$$

Here, the alphabet Σ contains at least the symbols $0, 1, \#$.

Often, it is useful to consider names of objects that enumerate basic information about the object.

Definition 4.5. We say that a $p \in \Sigma^\omega$ *enumerates* a set $A \subseteq \mathbb{N}$ if it satisfies the following two conditions:

1. if $\#w\#$ is a substring of p, then w is either empty or the binary name of an element of A,
2. for every element $n \in A$, the string $\#\nu_{\mathbb{N}}^{-1}(n)\#$ is a substring of p.

By using notations as in Section 2, one can generalize this notion straightforwardly to subsets A of $\mathbb{N}^n \times \mathbb{Q}^m$.

We use this idea in order to define two more representations of real numbers that we will need later.

Definition 4.6. The representation $\rho_< :\subseteq \Sigma^\omega \to \mathbb{R}$ is defined by

$$\rho_<(p) = x :\iff \quad p \text{ enumerates all } q \in \mathbb{Q} \text{ with } q < x.$$

Analogously, the representation $\rho_>$ is defined.

How can one perform computations with one-way infinite streams of symbols? One can feed the input stream(s) symbol by symbol into a Turing machine and demand that the Turing machine produces the output stream symbol by symbol.

Definition 4.7. Let Σ be an alphabet. A function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is *computable* if there exists a Turing machine that, given a $p \in \mathrm{dom}(F)$ as a stream on an input tape, writes the output stream $F(p)$ symbol by symbol, without ever stopping, on a one-way output tape, and that, given a $p \in \Sigma^\omega \setminus \mathrm{dom}(F)$ does not write infinitely many symbols on the output tape.

Remark 4.8. Note that we demand that the output tape is one way. This ensures that output symbols that have already been written will never be erased or changed again. With a two-way output tape it would be possible to erase or change already written output symbols later again. But then any output symbol written after finitely many steps might be false and would therefore be useless.

Definition 4.9. Let $f :\subseteq X \to Y$ be a function and $\delta_X :\subseteq \Sigma^\omega \to X$ and $\delta_Y :\subseteq \Sigma^\omega \to Y$ be representations. A function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is called a (δ_X, δ_Y)-*realizer* of f if

$$\delta_Y F(p) = f\delta_X(p) \quad \text{for all } p \in \mathrm{dom}(f\delta_X),$$

i.e., such that for any δ_X-name of some $x \in \mathrm{dom}(f)$, the value $F(p)$ is a δ_Y-name of $f(x)$. The function f is (δ_X, δ_Y)-*computable* if a computable (δ_X, δ_Y)-realizer of f exists.

Remark 4.10. Note that we do not impose any condition on how the realizer F should behave for input $p \notin \mathrm{dom}(f\delta_X)$. The motivation for this is that for any f that one wishes to compute, one is usually interested more in computing $f(x)$ for all valid input values x than in characterizing the domain of definition of f. That is a different problem, and it seems convenient to treat it separately. We have taken care of the domain only in our definition of computable F on Σ^ω, in Definition 4.7. Note that according to our definition, any restriction of a (δ_X, δ_Y)-computable function is (δ_X, δ_Y)-computable as well.

Since many real number functions that one wishes to compute expect as input not one real number but two or more, we need a representation of $\mathbb{R}^k$ for $k \geq 2$.

Definition 4.11. 1. Let $\delta :\subseteq \Sigma^\omega \to X$, $\delta' :\subseteq \Sigma^\omega \to X'$ be representations of sets X and X'. Then the representation $[\delta, \delta'] :\subseteq \Sigma^\omega \to X \times Y$ is defined by

$$[\delta, \delta']\langle p, q\rangle = (x, y) :\iff \delta(p) = x \text{ and } \delta'(q) = y,$$

where for $p, q \in \Sigma^\omega$ we define $\langle p, q\rangle := p(0)q(0)p(1)q(1)p(2)q(2)\ldots \in \Sigma^\omega$.

2. For $k \geq 1$, the representation δ^k of X^k is defined recursively by $\delta^1 := \delta$, $\delta^{k+1} := [\delta^k, \delta]$.

3. Sometimes we will also combine a notation $\nu :\subseteq \Sigma^* \to X$ and a representation $\delta :\subseteq \Sigma^\omega \to X'$ to a representation $[\nu, \delta] :\subseteq \Sigma^\omega \to X \times X'$, defined by

$$[\nu, \delta](0a_1 0a_2 0 \ldots a_{n-1}0a_n 1p) = (x, x') :\iff \nu(a_1a_2\ldots a_{n-1}a_n) = x \text{ and } \delta(p) = x'.$$

The discussion after Definition 4.1 shows that one can characterize the computable real number functions via the Cauchy representation ρ.

Lemma 4.12. *A real number function $f :\subseteq \mathbb{R}^k \to \mathbb{R}$ is computable in the sense of Definition 4.1 if, and only if, it is (ρ^k, ρ)-computable.*

Given the prominent role played by the decimal representation in daily life, it is natural to look at real number functions that are computable with respect to the decimal representation. But something goes wrong with the decimal representation, as the following observation shows.

Proposition 4.13. *The function $f : \mathbb{R} \to \mathbb{R}$, $f(x) := 3 \cdot x$ is not (ρ_{10}, ρ_{10})-computable.*

Proof. For the sake of a contradiction, assume that there is a Turing machine that, given a ρ_{10}-name of an arbitrary real number x, produces a ρ_{10}-name of $3 \cdot x$. Let us feed $0.3333\ldots$ into the machine. Then the machine has to produce either $0.99999\ldots$ or $1.00000\ldots$. Let us first assume that it produces $0.99999\ldots$. The machine will write the first symbol (a zero) of this output after finitely many steps, thus, after reading only finitely many symbols of the input, say, after reading at most the prefix 0.3^n, for some $n \in \mathbb{N}$. But then the first output symbol of the machine on input $0.3^n4444\ldots$ must be a zero as well although the numerical value of $3 \cdot \rho_{10}(0.3^n4444\ldots)$ is greater than 1, which means that it does not have ρ_{10}-name starting with a zero. Contradiction! In the other case, when on input $0.3333\ldots$ the machine produces $1.00000\ldots$, one arrives at a similar contradiction. We can summarize the argument as follows: it is impossible to determine even the first symbol of a decimal name for the real number $1 = 3 \cdot \rho_{10}(0.33333\ldots)$ correctly after reading only finitely many symbols of $0.33333\ldots$. □

Thus, not even multiplication with 3 is computable with respect to the decimal representation. Similarly one shows that also addition is not computable with respect to the decimal representation. This shows that the decimal representation is not suitable for computing real number functions. In fact, these negative statements are all due to topological reasons. This will be discussed in Section 6. There we will also discuss other representations of real numbers than the decimal representation and the Cauchy representation. Right now we stick to the computability notion for functions introduced above that can be described as computability with respect to the Cauchy representation (Lemma 4.12). This notion captures our intuition well that computations over the reals are approximative in nature and actually performed on rational numbers.

Our computability notion has properties that one expects from a computability notion for real number functions.

Theorem 4.14. *The following functions are computable:*

1. *The arithmetical operations* $+, -, \cdot, / :\subseteq \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.
2. *The absolute value function* $\mathrm{abs} : \mathbb{R} \to \mathbb{R}, x \mapsto |x|$.
3. *The functions* $\min, \max : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.
4. *The constant functions* $\mathbb{R} \to \mathbb{R}, x \mapsto c$ *with computable value* $c \in \mathbb{R}$.
5. *The projections* $\mathrm{pr}_i : \mathbb{R}^n \to \mathbb{R}, (x_1, \ldots, x_n) \mapsto x_i$.
6. *All polynomials* $p : \mathbb{R}^n \to \mathbb{R}$ *with computable coefficients.*
7. *The exponential function and the trigonometric functions* $\exp, \sin, \cos : \mathbb{R} \to \mathbb{R}$.
8. *The square root function* $\sqrt{}\{x \in \mathbb{R} \mid x \geq 0\} \subseteq \mathbb{R} \to \mathbb{R}$ *and the logarithm function* $\log : \{x \in \mathbb{R} \mid x > 0\} \subseteq \mathbb{R} \to \mathbb{R}$.

Proof. We sketch the proof that addition $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is computable. Given two sequences $(q_n)_{n\in\mathbb{N}}$ and $(r_n)_{n\in\mathbb{N}}$ of rational numbers that rapidly converge to x and y, respectively, we can compute the sequence $(p_n)_{n\in\mathbb{N}}$ of rational numbers defined by $p_n := q_{n+1} + r_{n+1}$. This sequence converges rapidly to $x + y$:

$$|x + y - p_n| \leq |x - q_{n+1}| + |y - r_{n+1}| < 2^{-n-1} + 2^{-n-1} = 2^{-n}.$$

Since addition on rational number can be computed by Turing machines, it follows that f is computable as well. □

Remark 4.15. Addition requires only a uniform lookahead of one step that does not depend on the input. For functions that are not uniformly continuous, such as multiplication, the "modulus of continuity" and thus the lookahead depend on the input.

Also more complicated functions such as the Gamma function or Riemann's zeta function are computable.

Now we list some basic properties of computable real number functions:

- they map computable real numbers to computable real numbers,
- they also map computable sequences of real numbers to computable sequences of real numbers,
- and they are closed under composition.

We can easily state these results more generally with respect to arbitrary representations of arbitrary representable sets.

Definition 4.16. 1. A sequence $p \in \Sigma^\omega$ is *computable* if there is a Turing machine that, given $k \in \mathbb{N}$ in binary form, produces the k-th symbol of p. It is equivalent to demand that there is a Turing machine with a one-way output tape that, without input and without ever stopping, writes p symbol by symbol on the output tape.

2. Let $\delta_X :\subseteq \Sigma^\omega \to X$ be a representation of a set X. An element $x \in X$ is δ_X*-computable* if it possesses a computable δ_X-name.

Example 4.17. By Theorem 3.2, the following conditions are equivalent for a real number x:

1. x is a computable real number,
2. x is a ρ_{10}-computable real number,
3. x is a ρ-computable real number.

Proposition 4.18. *If $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is a computable function and $p \in \Sigma^\omega$ is a computable element in the domain of F, then $F(p)$ is computable as well.*

Proof. One simply has to combine a Turing machine computing p with a Turing machine computing F in order to obtain a Turing machine computing $F(p)$. □

Corollary 4.19. *Let X and Y be sets with representations δ_X and δ_Y. If $f :\subseteq X \to Y$ is (δ_X, δ_Y)-computable and $x \in X$ is a δ_X-computable element in the domain of f, then $f(x)$ is δ_Y-computable.*

In particular, a computable real number function maps any computable real number in its domain to a computable real number.

Everything that we just said about computable elements is also true for computable sequences of elements. Computable functions preserve computability of sequences as well.

Definition 4.20. 1. A sequence $(s^{(n)})_{n\in\mathbb{N}}$ of elements $s^{(n)} \in \Sigma^\omega$ is *computable* if a Turing machine, given $n, k \in \mathbb{N}$ in binary form, produces the k-th symbol of

$s^{(n)}$. It is equivalent to demand that there is a Turing machine with a one-way output tape that, given $n \in \mathbb{N}$ in binary form, writes $s^{(n)}$ symbol by symbol on the output tape.

2. Let $\delta_X :\subseteq \Sigma^\omega \to X$ be a representation of a set X. A sequence $(x_n)_{n\in\mathbb{N}}$ of elements of X is *δ_X-computable* if there is a computable sequence of δ_X-names for the x_n.

Example 4.21. A sequence of real numbers is computable if, and only if, it is ρ-computable.

Proposition 4.22. *If $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is a computable function and $(s^{(n)})_{n\in\mathbb{N}} \in (\Sigma^\omega)^\omega$ is a computable sequence of elements in the domain of F, then the sequence $(F(s^{(n)}))_{n\in\mathbb{N}}$ is computable as well.*

Proof. Similar to the proof of Proposition 4.18. □

Corollary 4.23. *Let X and Y be sets with representations δ_X and δ_Y. If $f :\subseteq X \to Y$ is (δ_X, δ_Y)-computable and $(x_n)_{n\in\mathbb{N}} \in X$ is a δ_X-computable sequence of elements in the domain of f, then $(f(x_n))_{n\in\mathbb{N}}$ is a δ_Y-computable sequence.*

In particular, a computable real number function maps any computable sequence of real numbers in the domain of the function to a computable sequence of real numbers.

Finally, the composition of computable functions is computable again.

Proposition 4.24. *If $F, G :\subseteq \Sigma^\omega \to \Sigma^\omega$ are computable, then their composition $G \circ F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is computable as well.*

Proof. We combine Turing machines T_G for G and T_F for F in the following way. T_G uses the output tape of T_F as input tape and T_F receives p on its input tape. We start T_G. Whenever T_G needs to read a symbol of $F(p)$, we check whether T_F has already written this symbol. If not, we let T_G pause and start T_F and let it run until it has written this symbol. Then we let it pause again and take up the computation of T_G again. And so on. Does this composition of machines compute the composition $G \circ F$ of the functions F and G? Almost. In fact, this composition of the two machines might compute an extension of the composition $G \circ F$. On input $p \in \mathrm{dom}(G \circ F)$, it correctly computes the value $G \circ F(p)$. But, when there is some input $p \in \Sigma^\omega \setminus \mathrm{dom}(F)$, where T_F produces only finitely many output symbols, and T_G happens to need no more than these for producing some infinite output, then this composition of machines produces some infinite output on input p although $G \circ F$ is not defined at p. Therefore, we introduce the following modification of the composition of the two machines described above: whenever G is about to write the n-th output symbol, we check whether F has already produced at least n symbols. If not, we let F run until it has (if it never does, G will never write an n-th output symbol). This modified machine indeed computes the function $G \circ F$. □

Corollary 4.25. *Let X, Y, Z be sets with representations $\delta_X, \delta_Y, \delta_Z$, respectively. If $f :\subseteq X \to Y$ is (δ_X, δ_Y)-computable and $g :\subseteq Y \to Z$ is (δ_Y, δ_Z)-computable, then $g \circ f$ is (δ_X, δ_Z)-computable.*

In particular, the composition of computable real number functions is computable as well.

Now we come to a very important property of computable real number functions: they are continuous. In fact, they are exactly the real number functions that are effectively continuous in the following sense. Here $\mathrm{B}(x, \varepsilon) := \{y \in \mathbb{R}^n \mid d(x, y) < \varepsilon\}$ is the open ball in $\mathbb{R}^n$ with midpoint x and radius ε, for $x \in \mathbb{R}^n$ and $\varepsilon > 0$. We define a total numbering B^n of all open rational balls in $\mathbb{R}^n$ by

$$\mathrm{B}^n(\langle i_1, \ldots, i_n, j, k\rangle) := \mathrm{B}\left((\nu_Q(i_1), \ldots, \nu_Q(i_n)), \frac{j+1}{k+1}\right).$$

Definition 4.26. A function $f :\subseteq \mathbb{R}^n \to \mathbb{R}$ is *effectively continuous* if there is a c.e. subset $S \subseteq \mathbb{N}$ with the following two properties:

1. For any $\langle i, j\rangle \in S$, $f(\mathrm{B}^n(i)) \subseteq \mathrm{B}^1(j)$.
2. For any $x \in \mathrm{dom}(f)$ and any $\varepsilon > 0$, there is some $\langle i, j\rangle \in S$ such that $x \in \mathrm{B}^n(i)$ and such that the radius of $\mathrm{B}^1(j)$ is at most as large as ε.

Theorem 4.27. *A function $f :\subseteq \mathbb{R}^n \to \mathbb{R}$ is computable if, and only if, it is effectively continuous.*

Proof. For simplicity, we consider only the case $n = 1$. First, assume that f is effectively continuous. Then, given a ρ-name of some point $x \in \mathrm{dom}(f)$, one can, using an enumeration of S, compute $f(x)$ with arbitrary precision. Thus, f is computable.

Now, assume that f is computable. Then a Turing machine, given any k and given any name p of any point $x \in \mathrm{dom}(f)$, computes a rational 2^{-k}-approximation q of $f(x)$, i.e., a rational number q with $f(x) \in \mathrm{B}(q, 2^{-k})$. The machine does so after reading at most a finite prefix of p having the form $w_0\#w_1\#w_2\#\ldots w_l\#$. Then $U := \bigcap_{i=0}^{l} \mathrm{B}(\nu_{\mathbb{Q}}(w_i), 2^{-i})$ is an open neighborhood of x, and any point in this open neighborhood has a ρ-name starting with this prefix. Thus, given a name starting with $w_0\#w_1\#w_2\#\ldots w_l\#$ of any point in this neighborhood, the Turing machine will produce the same output q. This implies $f(U) \subseteq \mathrm{B}(q, 2^{-k})$, and that means that f is continuous. By systematically enumerating all prefixes of ρ-names of real numbers and testing the Turing machine on them, one can construct a c.e. set S that shows that f is even effectively continuous. □

The fact that even a function as simple as the sign function

$$\mathrm{sign} : \mathbb{R} \to \mathbb{R}, \quad \mathrm{sign}(x) := \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0 \end{cases}$$

is not computable may seem counter-intuitive at first. But, if the input real number x is given only through approximations (even arbitrarily good ones) and if x happens to be equal to a point of discontinuity of the function, then the value of the function can simply not be computed from the input. Of course, for example, the sign function is computable in various weaker senses (e.g., lower semi-computable, a notion we will define soon), which are also useful, and it is also a computable point in a certain function space (this will be defined soon as well). Thus, it is simple in certain senses. But in the numerical sense considered by us, it is definitely not computable.

It is interesting that continuity is a feature not only of this computability notion but also of two other natural computability notions for real number functions that can be defined via the Turing machine model. Both of them apply to functions that are defined only on computable real numbers and that map computable real numbers to computable real numbers. In the following we list four computability notions on real numbers or on computable real numbers and explain the relations between them.

1. Computability for functions $f :\subseteq \mathbb{R} \to \mathbb{R}$, i.e., the computability notion for real number functions that we have considered so far. It goes back to Grzegorczyk [41] and Lacombe [62].
2. Markov computability for functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$. If one calls a standard description of a Turing machine a *program*, and if the Turing machine computes a ρ-name of a real number, then this description can be called a *program* for this computable real number. A function $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ is *Markov computable* if there exists a computable function mapping finite strings to finite strings that maps any program for any $x \in \mathrm{dom}(f)$ to a program for $f(x)$. This computability notion has been considered in the Russian school of constructive analysis, e.g., by Ceĭtin [31], Šanin [85], and Kušner [59]. For a purely computability theoretic presentation of many results in this direction, the reader is referred to Aberth [1].
3. Sequential computability for functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$. A function $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ is *sequentially computable* if it maps any computable sequence of real numbers in $\mathrm{dom}(f)$ to a computable sequence of real numbers. This notion has been studied by Mazur [63].
4. Computable invariance for functions $f :\subseteq \mathbb{R} \to \mathbb{R}$. A function $f :\subseteq \mathbb{R} \to \mathbb{R}$ is *computably invariant* if it maps any computable real number in its domain to a computable real number.

Now we explain the connections between these notions.

- First we observe that one can apply the computability notion for functions $f :\subseteq \mathbb{R} \to \mathbb{R}$, of course, also to functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ and that this gives rise to a fifth class of functions. We have already seen that any computable function $f :\subseteq \mathbb{R} \to \mathbb{R}$ is computably invariant. And it is clear that any restriction of a computable real number function is a computable real number function as well.

Thus, if $f :\subseteq \mathbb{R} \to \mathbb{R}$ is computable, then its restriction $f|_{\mathbb{R}_c} :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ is computable again. But, there exists a computable function $g : \mathbb{R}_c \to \mathbb{R}_c$ defined on all computable real numbers which cannot be extended to a continuous function defined on all real numbers; see Aberth [1].

- Now we compare computable functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ and Markov computable functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$. It is easy to show that any computable function $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ is Markov computable. The converse is not true in general; see Slisenko [93] or Weihrauch [109, Example 9.6.5]. But it is an important fact due to Ceĭtin [31] that the converse is true for Markov computable functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ with *computably separable* domain, i.e., a domain $\mathrm{dom}(f)$ such that there exists a computable sequence $(x_n)_{n\in\mathbb{N}}$ of real numbers such that $\{x_n \mid n \in \mathbb{N}\}$ is a dense subset of $\mathrm{dom}(f)$. Related results were obtained by Kreisel et al. [57] and by Moschovakis [70]. See Kušner [59] for a careful presentation and discussion of Ceĭtin's result and related results. In Theorem 4.27, we observed that computability is equivalent to effective continuity. So, Ceĭtin's result says that any Markov computable function $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ with a computably separable domain is effectively continuous.

- Next, we compare Markov computable functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ and sequentially computable functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$. It is clear that any Markov computable function $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ is sequentially computable. The converse is not true: there exists even a sequentially computable function defined on all computable real numbers that is not Markov computable; see Hertling [49]. Nevertheless, surprisingly, any sequentially computable function with computably separable domain is continuous. This has already been observed by Mazur [63] (for functions defined on an interval).

- Finally, we compare sequentially computable functions $f :\subseteq \mathbb{R}_c \to \mathbb{R}_c$ and computably invariant functions $f :\subseteq \mathbb{R} \to \mathbb{R}$. Trivially, any real number function that maps any computable sequence of real numbers in its domain to a computable sequence of real numbers is computably invariant. The converse is not true. For example, the restriction of the sign function to the computable real numbers is a total function from $\mathbb{R}_c$ to $\mathbb{R}_c$ and computably invariant but not continuous, hence, not sequentially computably.

Functions considered in analysis that are not computable are often not computable simply because they are discontinuous. But they may still be computably invariant, like the sign function. It is often, not only for real number functions, but for many other kinds of functions, an interesting task to show that some noncomputable function is not even computably invariant, i.e., that there exists a computable input element such that the output element is not computable. We will see some examples of this later on.

5 Computability Notions for Subsets of Euclidean Space

Over the natural numbers, not only computable functions are of fundamental importance, but also effectivity notions for sets are important. The two most important classes of subsets of $\mathbb{N}^n$ defined by computability conditions are certainly

1. the computable (or decidable or recursive) subsets,
2. and the computably enumerable (or recursively enumerable) subsets.

We have already learned about computable functions over the real numbers. In this section we will see that there are natural computability notions for sets of real numbers that correspond to computable or to computably enumerable sets of natural numbers. Furthermore, these notions have a natural computational meaning, and they generalize the notions for subsets of $\mathbb{N}$.

Let us first consider the computable subsets. A subset $A \subseteq \mathbb{N}$ is called *computable* or *decidable* or *recursive* if its characteristic function $\chi_A : \mathbb{N} \to \mathbb{N}$, defined by

$$\chi_A(n) := \begin{cases} 0 & \text{if } n \in A, \\ 1 & \text{if } n \notin A, \end{cases}$$

is computable. That means, a set A is decidable if a Turing machine, given a natural number n, will after finitely many steps give the answer yes or no, saying whether n is an element of A. Let us first try to translate this idea to subsets of $\mathbb{R}^n$.

Definition 5.1. Let X be a set with a representation δ_X. Let us call a subset $A \subseteq X$ *δ_X-decidable* if the characteristic function $\chi_A : X \to \mathbb{R}$, defined by

$$\chi_A(x) := \begin{cases} 0 & \text{if } x \in A, \\ 1 & \text{if } x \notin A, \end{cases}$$

is (δ_X, ρ)-computable. The (ρ^n, ρ)-decidable subsets of $\mathbb{R}^n$ are called *decidable*.

Instead of considering the output values 0 and 1 as elements of $\mathbb{R}$, one might as well consider them as elements of $\mathbb{N}$. That means, a set $A \subseteq \mathbb{R}^n$ is decidable if, and only if, a Turing machine, given a ρ^n-name of a point $x \in \mathbb{R}^n$, will after finitely many steps give the answer yes or no, saying whether x is an element of A. Unfortunately, only two trivial subsets of $\mathbb{R}^n$ are decidable in this sense.

Lemma 5.2. *The only decidable subsets of $\mathbb{R}^n$ are $\emptyset$ and $\mathbb{R}^n$.*

Proof. The function $\chi_\emptyset$ and $\chi_{\mathbb{R}^n}$ are constant functions with value 1 and value 0, respectively, and obviously computable. The characteristic function of any other subset of $\mathbb{R}^n$ is discontinuous and therefore not computable, according to Theorem 4.27. □

Thus, this notion of decidability is not very interesting. What is wrong here? Is in the end our definition of computability of real number functions unsuitable? We saw that it can be characterized as computability with respect to the representation ρ. Is there perhaps a representation of the real numbers that is better suited for real number computations than ρ, perhaps a representation that would make basic nontrivial subsets of $\mathbb{R}^n$ decidable, e.g., which would allow comparison of real numbers? This is not the case.

Proposition 5.3. *There is no representation $\delta :\subseteq \Sigma^\omega \to \mathbb{R}$ of the real numbers such that any of the tests $=, <, \leq$ is decidable with respect to δ, i.e., such that any of the three following subsets of $\mathbb{R}^2$ is δ^2-decidable:*

$$\{(x,y) \in \mathbb{R}^2 \mid x = y\}, \quad \{(x,y) \in \mathbb{R}^2 \mid x < y\}, \quad \{(x,y) \in \mathbb{R}^2 \mid x \leq y\}.$$

Proof. We give the proof for the equality test. Let us assume that there is a representation δ such that the equality of two real numbers is decidable with respect to δ. Now consider an arbitrary real number x and an arbitrary δ-name p for x. On input $\langle p, p\rangle$, after finitely many steps, the Turing machine will have written a prefix of a ρ-name of 0 that cannot be a prefix of a ρ-name of 1. That means, the machine has come to the conclusion "equal." During these finitely many steps it can have read only a finite prefix of its input. Let n be a number such that it has read at most the first $2n$ symbols of its input $\langle p, p\rangle$. Then, for any input $\langle p, q\rangle$ such that q starts with $u := p(0)p(1)\ldots p(n-1)$, the machine would also come to the conclusion "equal." This implies that $\delta(q) = \delta(p) = x$ for every δ-name q starting with the string u. Since there are only countably many strings in Σ^* and $\delta :\subseteq \Sigma^\omega \to \mathbb{R}$ is surjective, this implies that $\mathbb{R}$ is countable. Contradiction. □

Thus, trying to change to another representation does not help. Therefore, we will stick to our original notion of computability of real number functions and accept that one cannot effectively make a discontinuous decision. This observation leads to the question of whether, maybe, there is a useful "smooth" replacement for discontinuous decisions? Let us consider a nonempty closed subset $A \subseteq \mathbb{R}^n$. The distance function $d_A : \mathbb{R}^n \to \mathbb{R}$, defined by

$$d_A(x) := \inf_{y \in A} d(x,y),$$

can be considered as a "smooth" version of the characteristic function of A.

Definition 5.4. We call a closed subset $A \subseteq \mathbb{R}^n$ *computable* or *recursive* if either it is empty or d_A is computable. We call an open subset $U \subseteq \mathbb{R}^n$ *computable* or *recursive* if its complement (which is a closed set) is computable.

First, we notice that this notion is a generalization of the decidability notion for subsets of $\mathbb{N}^n$.

Proposition 5.5. *A subset $A \subseteq \mathbb{N}^n$ is decidable in the classical sense (as a subset of $\mathbb{N}^n$) if, and only if, it is computable when considered as a closed subset of $\mathbb{R}^n$ (under the natural embedding $\mathbb{N}^n \to \mathbb{R}^n$).*

The proof is straightforward.

Examples 5.6.

1. The empty set $\emptyset$ and the full set $\mathbb{R}^n$ are computable closed sets and computable open sets.
2. For any $x \in \mathbb{R}^n$, the set $\{x\}$ is computable if, and only if, x is a computable point.
3. The equality relation and the $\leq$-Relation are computable; i.e., the closed subsets $\{(x,y) \in \mathbb{R}^2 \mid x = y\}$ and $\{(x,y) \in \mathbb{R}^2 \mid x \leq y\}$ of $\mathbb{R}^2$ are computable.
4. The open ball $\mathrm{B}(x,\varepsilon) = \{y \in \mathbb{R}^n \mid d(x,y) < \varepsilon\}$ and the closed ball $\overline{\mathrm{B}}(x,\varepsilon) = \{y \in \mathbb{R}^n \mid d(x,y) \leq \varepsilon\}$ are computable if $x \in \mathbb{R}^n$ is a computable point and $\varepsilon > 0$ is a computable real number.
5. For $a, b \in \mathbb{R}$ with $a \leq b$, the closed interval $[a,b]$ is computable if, and only if, a and b are computable real numbers. The same is true for the open interval (a,b).
6. For any total function $f : \mathbb{R}^n \to \mathbb{R}$, the graph $\mathrm{graph}(f) := \{(x_1, \ldots, x_n, y) \mid x_1, \ldots, x_n, y \in \mathbb{R},\ y = f(x_1, \ldots, x_n)\}$ is a computable closed set if, and only if, f is a computable function.

The computability notion for closed sets has a very intuitive meaning: the following proposition can be interpreted as saying that a closed subset $A \subseteq \mathbb{R}^n$ is computable if, and only if, one can plot pixel images of it with any desired precision. This condition, i.e., the second condition in the following proposition, will be used later for defining the computational complexity of a closed set. Here, computability on $\mathbb{Z}$ can be reduced to computability on $\mathbb{N}$ via the bijection $\nu : \mathbb{N} \to \mathbb{Z}$, $\nu(2n) := n$, $\nu(2n+1) := -n-1$.

Proposition 5.7. *For a closed set $A \subseteq \mathbb{R}^k$, the following two conditions are equivalent.*

1. *A is computable.*
2. *There exists a computable function $f : \mathbb{N} \times \mathbb{Z}^k \to \mathbb{N}$ with $\mathrm{range}(f) \subseteq \{0,1\}$ and such that for all $n \in \mathbb{N}$ and $z \in \mathbb{Z}^k$*

$$f(n,z) = \begin{cases} 0 & \text{if } d_A(\frac{z}{2^n}) < 2^{-n}, \\ 1 & \text{if } d_A(\frac{z}{2^n}) > 2 \cdot 2^{-n}, \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases}$$

Now let us see whether we can find also a natural generalization of the notion of a computably enumerable subset of $\mathbb{N}^n$ to subsets of $\mathbb{R}^n$. We will see that there are

even two natural generalizations, one for open subsets and one for closed subsets. As is well known, a set $A \subseteq \mathbb{N}$ is called *computably enumerable* (short: *c.e.*) or *recursively enumerable* if it satisfies one (and then both) of the following two equivalent conditions:

1. there is a computable function $f :\subseteq \mathbb{N} \to \mathbb{N}$ with $\mathrm{dom}(f) = A$; i.e., a Turing machine, given (a binary name of) some $n \in \mathbb{N}$, stops after finitely many steps if, and only if, $n \in A$,
2. A is empty or there is a total computable function $f : \mathbb{N} \to \mathbb{N}$ whose range is equal to A.

A set $A \subseteq \mathbb{N}$ is called *co-c.e.* if its complement is c.e. Using the pairing function $\langle \cdot, \cdot \rangle$, one can generalize these notions to subsets of $\mathbb{N}^k$ for $k \geq 2$. And using ν_Q from Section 2, one can translate it to subsets of $\mathbb{Q}^m$: we call a subset $S \subseteq \mathbb{Q}^m$ *c.e.* if the set $\{(n_1, \ldots, n_m) \in \mathbb{N}^m \mid (\nu_Q(n_1), \ldots, \nu_Q(n_m)) \in S\}$ is a c.e. subset of $\mathbb{N}^m$. The idea in the first characterization can be transferred directly to a represented set.

Definition 5.8. Let X be a set with a representation $\delta :\subseteq \Sigma^\omega \to X$. A set $U \subseteq X$ is called *δ-c.e.* if a Turing machine, given an arbitrary δ-name p of an arbitrary element $x \in X$, halts after finitely many steps if, and only if, $x \in U$.

Applying the idea in the first characterization to real numbers leads to the first condition in the following proposition, whereas the second condition above corresponds to the second and third conditions in the proposition.

Proposition 5.9. *Let $U \subseteq \mathbb{R}^n$ be open and $A := \mathbb{R}^n \setminus U$ be its (closed) complement. The following conditions are equivalent.*

1. *U is ρ^n-c.e.*
2. *$U = \bigcup_{(q,\varepsilon)\in S} \mathrm{B}(q, \varepsilon)$ for some c.e. set $S \subseteq \mathbb{Q}^n \times \mathbb{Q}_+$ (here $\mathbb{Q}_+ := \{q \in \mathbb{Q} \mid q > 0\}$).*
3. *The set $\{(q, \varepsilon) \in \mathbb{Q}^n \times \mathbb{Q}_+ \mid \overline{\mathrm{B}}(q, \varepsilon) \subseteq U\}$ is computably enumerable.*
4. *$A = f^{-1}(\{0\})$ for some total computable function $f : \mathbb{R}^n \to \mathbb{R}$.*
5. *The function $\chi_A : \mathbb{R}^n \to \mathbb{R}$ is* lower semicomputable*; i.e., it is $(\rho^n, \rho_<)$-computable.*
6. *Either $A = \emptyset$ or the function $d_A : \mathbb{R}^n \to \mathbb{R}$ is* lower semicomputable*.*

Definition 5.10. Let $U \subseteq \mathbb{R}^n$ be open and $A := \mathbb{R}^n \setminus U$ be its (closed) complement. If one (and then all) of the conditions in the previous proposition are satisfied, then U is called *c.e. open* and A is called *co-c.e. closed*.

The third condition above can be understood as saying that we can enumerate "negative" information about the closed set A, namely all closed rational balls contained in the complement of A. This corresponds to the following well-known topology on the space of all closed subsets (see Beer [6] for hyperspace topologies).

Definition 5.11. The *upper Fell topology* on the space of all closed subsets of $\mathbb{R}^n$ is the topology generated by the subbase consisting of all sets

$$\{A \subseteq \mathbb{R}^n \mid A \text{ is closed and } A \cap K = \emptyset\},$$

for compact sets $K \subseteq \mathbb{R}^n$.

There is another natural generalization of computable enumerability to subsets of $\mathbb{R}^n$.

Proposition 5.12. *Let $A \subseteq \mathbb{R}^n$ be closed. The following are equivalent.*

1. *The set $\{(q, \varepsilon) \in \mathbb{Q}^n \times \mathbb{Q}_+ \mid \mathrm{B}(q, \varepsilon) \cap A \neq \emptyset\}$ is computably enumerable.*
2. *Either A is empty or there is a computable sequence $(x_i)_{i \in \mathbb{N}}$ of points $x_i \in \mathbb{R}^n$ such that A is the closure of the set $\{x_i \mid i \in \mathbb{N}\}$.*
3. *Either A is empty or the function $d_A : \mathbb{R}^n \to \mathbb{R}$ is* upper semicomputable*; i.e., it is $(\rho^n, \rho_>)$-computable.*

Definition 5.13. Let $A \subseteq \mathbb{R}$ be closed. If one (and then all) of the conditions in the previous proposition are satisfied, then A is called *c.e. closed*, and its (open) complement is called *co-c.e. open*.

The first condition in the proposition can be understood as saying that we can enumerate "positive" information about the closed set A, namely all rational balls having nonempty intersection with A. As above, this corresponds also to a well-known topology on the space of all closed subsets.

Definition 5.14. The *lower Fell topology* on the space of all closed subsets of $\mathbb{R}^n$ is the topology generated by the subbase consisting of all sets

$$\{A \subseteq \mathbb{R}^n \mid A \text{ is closed and } A \cap U \neq \emptyset\},$$

for open sets $U \subseteq \mathbb{R}^n$.

Both of these two notions of computable enumerability for open or closed subsets of $\mathbb{R}^n$ are generalizations of the corresponding notions for subsets of $\mathbb{N}^n$.

Proposition 5.15. *1. A subset $A \subseteq \mathbb{N}^n$ is c.e. in the classical sense (as a subset of $\mathbb{N}^n$) if, and only if, it is c.e. closed when considered as a closed subset of $\mathbb{R}^n$ (under the natural embedding $\mathbb{N}^n \to \mathbb{R}^n$).*

2. *A subset $A \subseteq \mathbb{N}^n$ is co-c.e. in the classical sense (as a subset of $\mathbb{N}^n$) if, and only if, it is co-c.e. closed when considered as a closed subset of $\mathbb{R}^n$ (under the natural embedding $\mathbb{N}^n \to \mathbb{R}^n$).*

A subset of $\mathbb{N}^n$ is decidable if, and only if, it is c.e. and its complement is c.e. as well. This generalizes to subsets of $\mathbb{R}^n$ as well.

Proposition 5.16. *An open or closed subset of $\mathbb{R}^n$ is computable if, and only if, it is c.e. and its complement is c.e. as well.*

Proof. It suffices to prove this for closed sets. It follows from the fact that a real number function is computable if, and only if, it is upper semicomputable and lower semicomputable. Apply this to the function d_A for some nonempty closed set $A \subseteq \mathbb{R}^n$. □

For any class of subsets of $\mathbb{R}^n$, it is natural to ask whether it is closed under basic set theoretic operations.

Proposition 5.17. *Let A and B be closed subsets of $\mathbb{R}^n$.*

1. *If A and B are co-c.e. closed, so are $A \cup B$ and $A \cap B$.*
2. *If A and B are c.e. closed, so is $A \cup B$.*
3. *If A and B are computable, so is $A \cup B$.*

Proof. 1. Let $f, g : \mathbb{R}^n \to \mathbb{R}$ be computable functions with $A = f^{-1}(\{0\})$ and $B = g^{-1}(\{0\})$. Then $A \cup B = (f \cdot g)^{-1}(\{0\})$ and $A \cap B = (|f| + |g|)^{-1}(\{0\})$, and $f \cdot g$ and $|f| + |g|$ are also computable functions.

2. Remember that a closed set is c.e. closed if, and only if, the set of all open rational balls intersecting the set can be enumerated effectively. Now note that any open (rational) ball intersects $A \cup B$ if, and only if, it intersects at least one of the two sets A and B.

3. This follows from the first two statements.

□

We illustrate a typical technique that is used in computable analysis. We encode a c.e. but nondecidable set $K \subseteq \mathbb{N}$ into a subset of the reals in order to obtain a counterexample.

Example 5.18. There are two computable closed sets $A, B \subseteq \mathbb{R}$ such that $A \cap B$ is not even c.e. closed. Indeed, let $h : \mathbb{N} \to \mathbb{N}$ be a total computable function with nondecidable range $K := \mathrm{range}(h)$. We define $A := \bigcup_{n=0}^{\infty} A_n$ with

$$A_n := \begin{cases} [n - \frac{1}{2}, n] & \text{if } n \notin K, \\ [n - \frac{1}{2}, n - 2^{-k-2}] & \text{if } n \in K \text{ and } k = \min\{i \mid h(i) = n\}. \end{cases}$$

It is not hard to check that A is computable. Furthermore, we define $B := \mathbb{N} \subseteq \mathbb{R}$. Then $A \cap B = \mathbb{N} \setminus K$. This is a closed, but not a c.e. closed, subset of $\mathbb{R}$.

We conclude this section with an open problem. The famous Mandelbrot set M is a subset of the complex plane $\mathbb{C}$ that can be defined as follows:

$$M = \{c \in \mathbb{C} \mid \text{ all numbers in the sequence } (z_i)_{i\in\mathbb{N}} \text{ of complex numbers defined by } z_0 := 0 \text{ and } z_{i+1} := z_i^2 + c \text{ satisfy } |z_i| \leq 2\}.$$

The Mandelbrot set is a closed subset of the complex plane contained in the circle with radius 2 around the origin. In the following we identify $\mathbb{C}$ with $\mathbb{R}^2$. It is known that the complement of the Mandelbrot set is c.e. open (Weihrauch [109, Exercise 5.1.32]) and that the boundary of the Mandelbrot set is c.e. closed (Hertling [50]).

Problem 5.19. 1. Is the Mandelbrot set computable?

2. Is the interior of the Mandelbrot set c.e. open?

The first question is equivalent to the question of whether the Mandelbrot set is c.e. closed: compare Proposition 5.16. The second question is stronger than the first because of the following simple lemma.

Lemma 5.20. *The closure of a c.e. open subset $U \subseteq \mathbb{R}^n$ is c.e. closed.*

The famous *hyperbolicity conjecture* says that a certain subset of the interior of the Mandelbrot set, the union of its so-called "hyperbolic components," is actually equal to the interior of the Mandelbrot set; compare, e.g., the introductory text [13] by Branner. Since the union of the hyperbolic components is c.e. open (Hertling [50]), one arrives at the following conclusion.

Proposition 5.21 (Hertling [50]). *If the hyperbolicity conjecture is true, then the answer to both questions in Problem 5.19 is yes.*

6 Representations and Topological Considerations

We have introduced computability notions for real numbers, for real number functions, and for sets of real numbers. We have seen that all these notions can be characterized using Turing machines and the Cauchy representation. We have also seen that the decimal representation is unsuitable for real number computations. But one

can define many other representations of real numbers. Maybe some other representation is even more suitable for real number computations than the Cauchy representation? In this section we will analyze representations more systematically. We will compare various real number representations, and we will make some general observations about representations that will be useful when in later sections we consider computability on spaces that are "more complicated" than the space of real numbers.

We will see soon that a good deal of the difficulties arising in the context of representations is of a topological nature. Indeed, already our first definition of computable real number functions contained the idea of approximating the real value $f(x)$ with arbitrary precision. Approximation is a notion from topology. So, here topology enters the stage, and we will see that it plays a very important role for computability notions over the real numbers or other nondiscrete sets.

First, the set Σ^ω, on which our Turing machines carry out the actual computations, carries a natural topology, namely the usual product topology of the discrete topology on Σ. Endowed with this topology Σ^ω is called *Cantor space*. A base of the topology is the set of all sets $w\Sigma^\omega := \{p \in \Sigma^\omega \mid p \text{ starts with } w\}$ with $w \in \Sigma^*$. This topology is also induced by the metric d on the Cantor space defined by

$$d(p,q) := \begin{cases} 0 & \text{if } p = q, \\ 2^{-\min\{n \mid p(n) \neq q(n)\}} & \text{otherwise.} \end{cases}$$

Thus, Σ^ω is a separable metric space. In addition, it is complete and (if Σ is finite, which we always assume) compact. A function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is continuous if for any finite prefix w of $F(p)$, there exists a finite prefix v of p such that $F(v\Sigma^\omega \cap \mathrm{dom}(F)) \subseteq w\Sigma^\omega$, i.e., such that w is a prefix of $F(q)$ for any $q \in v\Sigma^\omega \cap \mathrm{dom}(F)$. That means, any symbol of $F(p)$ depends only on finitely many symbols in p. An important observation is that any computable function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is continuous.

Proposition 6.1. *Any computable function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is continuous.*

Proof. A Turing machine that computes F has to compute any finite prefix w of $F(p)$ for any $p \in \mathrm{dom}(F)$ within finitely many steps. But until then it can have read only a finite prefix v of p. Then for any $q \in \mathrm{dom}(F)$ that starts with v as well, the Turing machine will also produce w within the same number of steps. Since the Turing machine has a one-way output tape, this implies that $F(q)$ starts with w as well. Hence, $F(v\Sigma^\omega \cap \mathrm{dom}(F)) \subseteq w\Sigma^\omega$. □

Similarly as in Theorem 4.27 for computable real number functions, the computable functions $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ can be characterized as exactly the effectively continuous functions $F :\subseteq \Sigma^\omega \to \Sigma^\omega$.

This proposition allows us to define the following purely topological weakening of our relative computability notion defined in Definition 4.9.

Definition 6.2. Let X and Y be sets with representations δ_X and δ_Y. A function $f :\subseteq X \to Y$ is called (δ_X, δ_Y)-*continuous* if a continuous (δ_X, δ_Y)-realizer F of f exists.

Often, a function that is not computable with respect to some representations is so for purely topological reasons; that is, it is not continuous with respect to the representations. This is for example the case for some real number functions that one certainly wants to be able to compute but that are not computable with respect to the decimal representation.

Example 6.3. We have seen that multiplication of real numbers by 3 is not (ρ_{10}, ρ_{10})-computable. A short look at the proof shows that this operation is not even (ρ_{10}, ρ_{10})-continuous.

By a similar argument one shows that also addition is not even continuous with respect to the decimal representation. As mentioned, already Turing noticed in a correction [103] to his paper [102] that the decimal (or binary) representation is unsuitable for defining computability of real number functions. Turing suggested to use a different representation instead. We have done that as well; namely, we have used and will be using in the future the Cauchy representation, which is "equivalent" to the representation suggested by Turing. Later we will consider other useful "equivalent" representations of the set of real numbers. What does "equivalent" mean here? As with relative computability and relative continuity, we define a computability-theoretic and a topological version.

Definition 6.4. Let $\delta, \delta' :\subseteq \Sigma^\omega \to X$ be representations of some set X.

1. δ is *reducible* to δ', written $\delta \leq \delta'$, if there is a computable function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ with $\delta(p) = \delta' F(p)$ for all $p \in \mathrm{dom}(\delta)$.
2. δ is *equivalent* to δ', written $\delta \equiv \delta'$, if $\delta \leq \delta'$ and $\delta' \leq \delta$.

If one asks F only to be continuous instead of computable, one obtains topological variants of these notions: *continuous reducibility*, written $\leq_{\mathrm{t}}$, and *continuous equivalence*, written $\equiv_{\mathrm{t}}$.

Example 6.5. We can define a variant ρ' of the Cauchy representation ρ of the real numbers as follows:

$$\rho'(w_0\#w_1\#w_2\#\ldots) = x :\iff \lim_{i\to\infty} \nu_{\mathbb{Q}}(w_i) = x \text{ and } (\forall i < j)\ |\nu_{\mathbb{Q}}(w_i) - \nu_{\mathbb{Q}}(w_j)| \leq 2^{-i}.$$

If $w_0\#w_1\#w_2\#\ldots$ is a ρ'-name of a real number x, then $w_1\#w_2\#\ldots$ is a ρ-name of the same number, and vice versa. Thus, ρ and ρ' are equivalent.

The following lemma is easy to prove.

Lemma 6.6. *1. δ is reducible (continuously reducible) to δ' if, and only if, the identity* $\mathrm{id} \to X$ *is (δ, δ')-computable ((δ, δ')-continuous).*

2. (Continuous) reducibility is a reflexive and transitive relation on the representations of a fixed set X.

We have already met several representations of the real numbers: the decimal representation ρ_{10} as well as the analogously defined representations ρ_b to any base $b \geq 2$, the Cauchy representation ρ, and the two representations $\rho_<$ and $\rho_>$. Here is another one.

Example 6.7. The *naive Cauchy representation* ρ_{nC} is defined like the Cauchy representation with the difference that the sequence of rational numbers listed by a ρ_{nC}-name of some real number x has to converge to x, but it does not need to converge rapidly, not even computably.

We have also learned how to combine representations of spaces into a representation of the product space. One can also combine different representations of a space into one representation as follows.

Definition 6.8. Let $\delta, \delta' :\subseteq \Sigma^\omega \to X$ be representations of a set X. Then the representation $\delta \wedge \delta' :\subseteq \Sigma^\omega \to X$ is defined by

$$(\delta \wedge \delta')\langle p, q\rangle = x :\iff \delta(p) = x \text{ and } \delta'(q) = x.$$

That means, a $\delta\wedge\delta'$-name of an element x contains the information about x contained in a δ-name for x and the information about x contained in a δ'-name for x.

In Figure 1 we describe the relationships between some real number representations. We observe that the differences between the representations in this diagram are really of a topological nature: whenever a representation in the diagram is not reducible to another one, it is not even continuously reducible to it.

Of course, equivalent representations induce the same computability notions.

Lemma 6.9. *Let $\gamma, \gamma' :\subseteq \Sigma^\omega \to X$ be representations with $\gamma \leq \gamma'$.*

1. Any γ-computable element of X is also γ'-computable.

2. Any γ-computable sequence in X is also γ'-computable.

3. Any γ'-c.e. subset $U \subseteq X$ is also γ-c.e.

4. If, additionally, $\delta, \delta' :\subseteq \Sigma^\omega \to Y$ are representations with $\delta \leq \delta'$, then any (γ', δ)-computable function $f :\subseteq X \to Y$ is also (γ, δ')-computable. The same holds true with "continuous" instead of "computable."

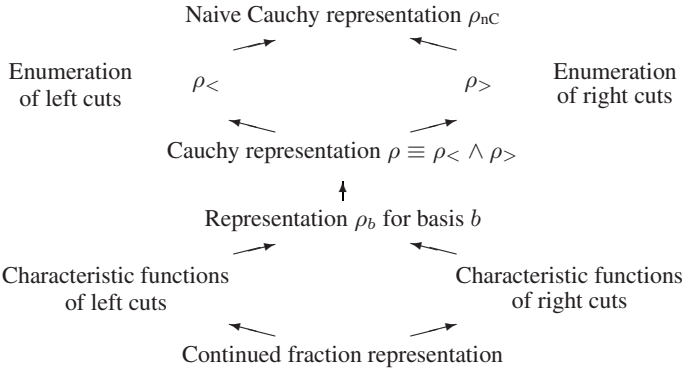

Fig. 1. Each arrow means $\leq$ and $\not\geq_{\mathrm{t}}$.

Proof. By Lemma 6.6 and by Corollary 4.19 for the first statement, by Corollary 4.23 for the second statement, and by Corollary 4.25 for the fourth statement. For the third statement one combines a Turing machine computing a reduction from γ to γ' with a Turing machine proving that U is γ'-c.e. For the topological version of the fourth statement, use that the composition of continuous functions is continuous as well. □

Corollary 6.10. *If $\gamma, \gamma' :\subseteq \Sigma^\omega \to X$ are equivalent representations, then*

1. *an element $x \in X$ is γ-computable if, and only if, it is γ'-computable,*
2. *a sequence of elements of X is γ-computable if, and only if, it is γ'-computable,*
3. *a subset $U \subseteq X$ is γ-c.e. if, and only if, it is γ'-c.e.*

If additionally $\delta, \delta' :\subseteq \Sigma^\omega \to Y$ are equivalent representations, then

4. *a function $f :\subseteq X \to Y$ is (γ, δ)-computable if, and only if, it is (γ', δ')-computable.*

Also the topological version of the last statement holds true.

It is interesting that representations that are not equivalent may nevertheless define the same class of computable elements: although the four lower representations in Figure 1 are not equivalent to the Cauchy representation ρ, the real numbers that are computable with respect to these representation are exactly the computable real numbers, i.e., the ρ-computable real numbers. The $\rho_<$-computable real numbers are exactly the left-computable real numbers, the $\rho_>$-computable real numbers are exactly the right-computable real numbers, and the ρ_{nC}-computable real numbers are exactly the computably approximable real numbers.

In contrast, already for sequences the notion of computability with respect to a representation is very sensitive with respect to the representation (Mostowski [72]), which is similar to functions (Turing [103]).

That the equivalence class of the Cauchy representation is really the right choice for defining computability over the real numbers can also be justified by the observation that it is the only equivalence class of representations with respect to which certain elementary operations are computable that correspond to the structure of the real numbers; see Hertling [48]. Very soon we will also characterize the continuous equivalence class of the Cauchy representation in a natural way.

One may now ask whether this equivalence class contains some representations with special properties, e.g., an injective or a total representation. But the different nature of the spaces Σ^ω and $\mathbb{R}$ implies that this is not the case.

Proposition 6.11. *1. No injective representation $\delta :\subseteq \Sigma^\omega \to \mathbb{R}$ is continuously equivalent to ρ.*

2. No total representation $\delta : \Sigma^\omega \to \mathbb{R}$ is continuously equivalent to ρ (given that Σ is finite).

In the following sections, we will consider computability on spaces "more complicated" than the space of real numbers, e.g., on function spaces or spaces of subsets of Euclidean space. We have already developed a machinery that allows us to introduce approximative notions of computability on such spaces in a natural and simple way: the machinery of representations and computability with respect to representations. But for any given space and any computational task involving that space the question arises of which representation to use. A natural requirement is that the representation should fit with the topology on the space with respect to which we wish to perform approximate computations. A very useful definition in this context is the following one.

Definition 6.12. A representation δ of a topological space X is called *admissible* if δ is continuous and $\delta' \leq_t \delta$ holds true for any continuous representation of X.

Examples 6.13. 1. The Cauchy representation ρ of $\mathbb{R}$ is admissible with respect to the usual Euclidean topology on $\mathbb{R}$.

2. The representations $\rho_<$ and $\rho_>$ are admissible with respect to the lower and upper Euclidean topology, respectively (these topologies are generated by the intervals (x, ∞) and $(-\infty, x)$, for $x \in \mathbb{R}$, respectively).

We have already observed that any computable real number function is continuous. In fact, one can show that a function $f :\subseteq \mathbb{R} \to \mathbb{R}$ is continuous if, and only if, it is (ρ, ρ)-continuous. This statement carries over to arbitrary admissible representations.

Theorem 6.14 (Kreitz and Weihrauch [58] and Schröder [91]). *Let (X, δ_X) and (Y, δ_Y) be admissibly represented topological spaces. Then a function $f :\subseteq X \to Y$ is (δ_X, δ_Y)-continuous if, and only if, it is sequentially continuous.*

A function $f :\subseteq X \to Y$ between topological spaces is *sequentially continuous* if it preserves convergence of sequences (in the domain of f). For T_0-spaces with countable base, e.g., $\mathbb{R}^n$ with the Euclidean topology, sequential continuity is equivalent to ordinary continuity. But in general it is a slightly weaker notion. The theorem has been generalized by Schröder to so-called weak limit spaces [89]. Schröder's results allow the definition of useful representations even for spaces that are not countably based. This has turned out to be very useful, e.g., for defining computability on spaces of generalized functions; see Zhong and Weihrauch [119]. Another interesting generalization by Schröder are multi-representations; see [90].

We conclude this section by mentioning that there are very useful standard ways for constructing new representations from given ones. We have seen this already for the product of represented sets in Definition 4.11. If (X, δ_X) and (Y, δ_Y) are spaces with representations, then obviously the projection $X \times Y \to X$ is $([\delta_X, \delta_Y], \delta_X)$-computable, and the projection $X \times Y \to Y$ is $([\delta_X, \delta_Y], \delta_Y)$-computable. But one can construct also a representation with natural properties for, e.g., the space of all relatively continuous functions.

Proposition 6.15. *Let (X, δ_X) and (Y, δ_Y) be spaces with representations. There is a representation $[\delta_X \to \delta_Y]$ of the set of all total (δ_X, δ_Y)-continuous functions $f : X \to Y$ such that the following statements are true.*

1. *The function*
$$(f, x) \mapsto f(x)$$
*is $([[\delta_X \to \delta_Y], \delta_X], \delta_Y)$-computable (*evaluation*).*
2. *For any represented space (Z, δ_Z), any function $f : Z \times X \to Y$ is $([\delta_Z, \delta_X], \delta_Y)$-computable if, and only if, the function*
$$z \mapsto (x \mapsto f(z, x))$$
*is $(\delta_Z, [\delta_X \to \delta_Y])$-computable (*type conversion*).*

A suitable representation $[\delta_X \to \delta_Y]$ can be defined via a standard encoding of the continuous functions $h :\subseteq \Sigma^\omega \to \Sigma^\omega$ whose domains are effective G_δ-sets.

Lemma 6.16. *The representation $[\rho^n \to \rho]$ is admissible with respect to the compact-open topology.*

The following result indicates that the class of so-called sequential T_0-spaces with admissible representations form a natural class for computability considerations.

Theorem 6.17 (Schröder [91]). *The category consisting of admissibly represented sequential T_0-spaces as objects and total continuous functions between them as morphisms is cartesian closed.*

7 Solvability of Some Problems Involving Sets and Functions

For many numerical problems the input is not a vector of real numbers, but rather a real number function, e.g., for integration and zero finding. In order to compute with functions as input, we have to feed some descriptions of them into the computer. Therefore, we need a representation. Many function spaces and other spaces on which one wishes to perform computations turn out to be computable metric spaces.

Definition 7.1. A triple (X, d, α) is called a *computable metric space*, if

1. $d : X \times X \to \mathbb{R}$ is a metric on X,
2. $\alpha : \mathbb{N} \to X$ is a sequence such that the set $\{\alpha(n) \mid n \in \mathbb{N}\}$ is dense in X,
3. $d \circ (\alpha \times \alpha) : \mathbb{N}^2 \to \mathbb{R}$ is a computable (double) sequence in $\mathbb{R}$.

Definition 7.2. Let (X, d, α) be a computable metric space. Then we define the *Cauchy representation* $\delta_X :\subseteq \Sigma^\omega \to X$ by

$$\delta_X(01^{n_0+1}01^{n_1+1}01^{n_2+1}...) := \lim_{i\to\infty} \alpha(n_i)$$

for any sequence $(n_i)_{i\in\mathbb{N}}$ such that $(\alpha(n_i))_{i\in\mathbb{N}}$ converges rapidly (and $\delta_X(p)$ is undefined for all other input sequences $p \in \Sigma^\omega$). We say that a sequence $(x_i)_{i\in\mathbb{N}}$ *converges rapidly*, if $d(x_i, \lim_{n\to\infty} x_n) < 2^{-i}$ for all i.

Examples 7.3. 1. $(\mathbb{R}^n, d, \alpha_{\mathbb{R}^n})$ with the *Euclidean metric*

$$d(x, y) := \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

and the standard numbering $\alpha_{\mathbb{R}^n}$ of $\mathbb{Q}^n$ defined by

$$\alpha_{\mathbb{R}^n}(\langle i_1, \ldots, i_n\rangle) := (\nu_Q(i_1), \ldots, \nu_Q(i_n))$$

is a computable metric space.

2. $(\mathcal{C}[0, 1], d_\mathcal{C}, \alpha_\mathcal{C})$ with the *supremum metric*

$$d_\mathcal{C}(f, g) := ||f - g|| := \sup_{x\in[0,1]} |f(x) - g(x)|$$

and some standard numbering $\alpha_{\mathcal{C}}$ of $\mathbb{Q}[x]$, for instance, $\alpha_{\mathcal{C}}(\langle k, \langle i_0, \ldots, i_k\rangle\rangle) := \sum_{j=0}^{k} \nu_Q(i_j) \cdot x^j$, is a computable metric space. Note that the Cauchy representation $\delta_{\mathcal{C}[0,1]}$ of $\mathcal{C}$ allows effective evaluation: the function $(f, x) \mapsto f(x)$ is $([\delta_{\mathcal{C}[0,1]}, \rho^{[0,1]}], \rho)$-computable (where $\rho^{[0,1]}$ is the restriction of ρ to names of real numbers in $[0, 1]$). Even more, $\delta_{\mathcal{C}[0,1]}$ is equivalent to the representation $[\rho^{[0,1]} \to \rho]$. This implies that the computable points in this space are exactly the computable functions $f : [0, 1] \to \mathbb{R}$. This is in fact an effective version of the Weierstrass Approximation Theorem; compare Pour-El and Richards [80].

3. $(\mathcal{K}(X), d_{\mathcal{K}}, \alpha_{\mathcal{K}})$ with the set $\mathcal{K}(X)$ of nonempty compact subsets of a computable metric space (X, d, α), with the *Hausdorff metric*

$$d_{\mathcal{K}}(A, B) := \max\left\{\sup_{a\in A}\inf_{b\in B} d(a, b), \sup_{b\in B}\inf_{a\in A} d(a, b)\right\},$$

and with some standard numbering $\alpha_{\mathcal{K}}$ of the nonempty finite subsets of $\mathrm{range}(\alpha)$; e.g., $\alpha_{\mathcal{K}}(-1 + \sum_{i\in E} 2^i) = \{\alpha(i) \mid i \in E\}$, for any finite nonempty $E \subseteq \mathbb{N}$, is a computable metric space. The computable points are exactly the nonempty computable compact subsets $A \subseteq X$.

Proposition 7.4. *Let X and Y be computable metric spaces with Cauchy representations δ_X and δ_Y, respectively. Then a function $f : X \to Y$ is (δ_X, δ_Y)-continuous if, and only if, it is continuous in the ordinary sense.*

Proof. It is clear that any computable metric space is a countably based T_0-space. We also note that the Cauchy representation of a computable metric space is admissible with respect to the topology induced by the metric. The assertion follows now from Theorem 6.14. □

To give an example, we consider several versions of the problem to find a zero of a continuous function. We remind the reader of the classic Intermediate Value Theorem.

Proposition 7.5. *For every continuous function $f : [0, 1] \to \mathbb{R}$ with $f(0) \cdot f(1) < 0$, there exists some $x \in [0, 1]$ with $f(x) = 0$.*

A number x with $f(x) = 0$ is called *a zero* of f. Is there an effective version of this statement? Under what circumstances can one find a zero of a continuous function $f : [0, 1] \to \mathbb{R}$ with $f(0) \cdot f(1) < 0$? First we are modest and restrict ourselves to functions that have exactly one zero. Then we can compute it.

Theorem 7.6. *The partial function on $\mathcal{C}[0, 1]$ that maps any $f \in \mathcal{C}[0, 1]$ with $f(0) \cdot f(1) < 0$ and exactly one zero to its zero is $(\delta_{\mathcal{C}[0,1]}, \rho)$-computable.*

Proof. Note that in the classical bisection method functions, values are compared with 0. We have seen that such a comparison cannot be performed reliably in finite time. Therefore, we use the *trisection* method, which is a stable variant of the classic bisection method. Assume that a $\delta_{\mathcal{C}[0,1]}$-name of a function as above is given. Remember that using this $\delta_{\mathcal{C}[0,1]}$-name and given any ρ-name of a real number in $x \in [0,1]$, we can compute $f(x)$.

We start with $a_0 := 0$ and $d_0 := 1$. Now assume that we have computed two numbers $a_i, d_i \in [0,1]$ with $d_i - a_i = (2/3)^i$ such that $f(a_i) \cdot f(d_i) < 0$ (this is true for $i = 0$). In parallel we compute the two values

$$f(a_i) \cdot f\left(a_i + \frac{2}{3}(d_i - a_i)\right) \quad \text{and} \quad f\left(a_i + \frac{1}{3}(d_i - a_i)\right) \cdot f(d_i)$$

with higher and higher precision until one of them turns out to be smaller than zero. Note that at least one of them must be smaller than zero! If the first one for which we verify this is the left one, we set $a_{i+1} := a_i$, $d_{i+1} := a_i + \frac{2}{3}(d_i - a_i)$, and if it is the right one, we set $a_{i+1} := a_i + \frac{1}{3}(d_i - a_i)$, $d_{i+1} := d_i$. In this way we obtain a sequence of intervals $[a_i, d_i]$ of length $(2/3)^i$ converging to the unique zero of f. This can easily be turned into a ρ-name of the zero. □

For functions with more than one zero, things are not that easy.

Theorem 7.7. *There is no continuous (and, hence, no $(\delta_{\mathcal{C}[0,1]}, \rho)$-computable) function $Z :\subseteq \mathcal{C}[0,1] \to \mathbb{R}$ with $f(Z(f)) = 0$ for all $f \in \{f \in \mathcal{C}[0,1] \mid f(0) \cdot f(1) < 0,$ and f has at most three zeros$\}$.*

The proof is quite simple and quite similar to the even slightly easier proof of Theorem 7.10, given below. We omit it.

But for functions with not too many zeros, we can still compute some zero.

Theorem 7.8. *There is a Turing machine which, given a $\delta_{\mathcal{C}[0,1]}$-name of some function $f \in \mathcal{C}$ with $f(0) \cdot f(1) < 0$ and such that $f^{-1}(\{0\})$ does not contain an interval, computes a (ρ-name of a) zero of f.*

Proof. First, we note that if f is such a function and $a < d$ are numbers in $[0,1]$ with $f(a) \cdot f(d) < 0$, then for any $n \in \mathbb{N}$ there are rational numbers b and c with $a \leq b < c \leq d$, with $c - b \leq 2^{-n}$, and with $f(b) \cdot f(c) < 0$.

Let us assume that a $\delta_{\mathcal{C}[0,1]}$-name of such a function f is given. The algorithm works similarly to the trisection algorithm: it starts with $a_0 := 0$, $d_0 := 1$, and once two rational numbers a_i and d_i with $a_i < d_i$, with $d_i - a_i \leq 2^{-i}$, and with $f(a_i) \cdot f(d_i) < 0$ are found, it searches for two rational numbers a_{i+1}, d_{i+1} with $a_i \leq a_{i+1} < d_{i+1} \leq d_i$, with $d_{i+1} - a_{i+1} \leq 2^{-i-1}$, and with $f(a_{i+1}) \cdot f(d_{i+1}) < 0$. □

Note that, due to Theorem 7.7, the zero found by a Turing machine as in Theorem 7.8 cannot depend only on the input function f. But it depends continuously on the actual $\delta_{\mathcal{C}[0,1]}$-name given to the Turing machine. Hence, for some function f, the algorithm will compute different zeros of f depending on the $\delta_{\mathcal{C}[0,1]}$-name of f given to the algorithm. This can be expressed in the terminology of representation-based computability using multi-valued functions. In the following, we call a relation $f \subseteq X \times Y$ a *multi-valued function* and write it as $f :\subseteq X \rightrightarrows Y$.

Definition 7.9. A multi-valued function is called (δ_X, δ_Y)-*computable*, if there exists a computable function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that

$$\delta_Y F(p) \in f\delta_X(p)$$

for all $p \in \mathrm{dom}(f\delta)$. Analogously, f is called (δ_X, δ_Y)-*continuous*, if there exists a continuous F with this property.

For example, a multi-valued function $Z :\subseteq \mathcal{C}[0,1] \rightrightarrows \mathbb{R}$ can be defined by

$$\mathrm{dom}(Z) := \{f \in \mathcal{C}[0,1] \mid f(0) \cdot f(1) < 0\},$$
$$Z(f) := f^{-1}(\{0\}).$$

In this terminology, Theorem 7.8 can be expressed as follows: the restriction of Z to the set

$$\{f \in \mathcal{C}[0,1] \mid f(0) \cdot f(1) < 0 \text{ and } f^{-1}(\{0\}) \text{ does not contain an interval}\}$$

is $(\delta_{\mathcal{C}[0,1]}, \rho)$-computable. In the general unrestricted case, one cannot even determine a zero in this weaker multi-valued sense.

Theorem 7.10. *The multi-valued function Z defined above is not $(\delta_{\mathcal{C}[0,1]}, \rho)$-continuous.*

Proof. For the sake of a contradiction, let us assume that Z is $(\delta_{\mathcal{C}[0,1]}, \rho)$-continuous. Consider the continuous, piecewise linear function $h_y : [0,1] \to \mathbb{R}$ with breakpoints $(0,-1), (1/3, y), (2/3, y), (1,1)$, for arbitrary $y \in \mathbb{R}$. Given a ρ-name of y, we can compute a $\delta_{\mathcal{C}[0,1]}$-name of h_y. Thus, if Z were $(\delta_{\mathcal{C}[0,1]}, \rho)$-continuous, then there would also be a continuous function mapping any ρ-name of any y to a ρ-name of a zero of h_y. But in any neighborhood of a ρ-name of $y = 0$, there are names of numbers $y < 0$ and of numbers $y > 0$. The unique zero of h_y for $y < 0$ is greater than $2/3$, whereas the unique zero of h_y for $y > 0$ is smaller than $1/3$. Hence, given a name of $y = 0$, a continuous function producing a name of a zero of h_y would have to produce a name of a number $\geq 2/3$ and at the same time a name of a number $\leq 1/3$. Impossible! □

All the results so far are *uniform* results in the sense that we asked what one can compute if (a $\delta_{\mathcal{C}[0,1]}$-name of) a continuous function is given. The following statement is a *nonuniform* result in the sense that we only ask whether a computable zero exists, not whether we can compute it from some input data.

Theorem 7.11. *Every computable function $f : [0,1] \to \mathbb{R}$ with $f(0) \cdot f(1) < 0$ has a computable zero.*

Proof. We distinguish two cases:

1. The set $f^{-1}(\{0\})$ does not contain an interval. Then, given a computable name of f, an algorithm as in Theorem 7.8 will produce a computable name of a zero of f. Hence, this zero is computable.
2. The set $f^{-1}(\{0\})$ contains an interval. Then it contains also a rational, hence, computable number.

□

In general, by Corollary 4.19, if one has a positive uniform result, also the non-uniform positive version holds true. But the previous two results show that the nonuniform version may be true even if the corresponding uniform version is false.

So far we looked at zero-finding for arbitrary continuous functions with a sign change on the unit interval. Now let us look at the more specialized problem to compute the zeros of a polynomial. By the Fundamental Theorem of Algebra, any polynomial

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0 \in \mathbb{C}[z]$$

of degree n, i.e., with $a_n \neq 0$, has exactly n complex zeros, when counted with multiplicity. Can one compute them? In the following, for computation purposes, we will always identify $\mathbb{C}$ with $\mathbb{R}^2$ and $\mathbb{C}^n$ with $\mathbb{R}^{2n}$ and we define a representation of $\mathbb{C}$ by $\rho_{\mathbb{C}} := \rho^2$. On first sight, the following well-known statement seems to say no. For $n \geq 1$, set

$$P_n := \{(a_n, \ldots, a_0) \in \mathbb{R}^{n+1} \mid a_n \neq 0 \text{ and } p(z) := a_n z^n + \cdots + a_0 \text{ has exactly } n \text{ pairwise different zeros}\}.$$

Lemma 7.12. *Fix some $n \geq 2$. There is no continuous function $Z : P_n \to \mathbb{C}^n$ such that $Z(p)$ contains all zeros of p, for any $p \in P_n$.*

What does it mean? Let us fix some $n \geq 2$. It means that even when restricted to polynomials that have exactly n pairwise different zeros, the problem to find a vector of all zeros contains a discontinuity. By Proposition 7.4 this implies that "there is no Turing machine which, given a $p \in P_n$, would produce a vector depending only on p and containing the zeros of p." But the statement in quotation marks should be read very carefully: in the statement and in Lemma 7.12, we are saying that for each polynomial $p \in P_n$, we wish to obtain a vector of zeros where the vector is determined by the polynomial. But a vector(!) of zeros contains actually more information than what we are interested in: in addition to the information that complex numbers are the zeros, in the vector the zeros are ordered! It turns out that this ordering is

the only problem and the reason for the discontinuity and, hence, unsolvability of this version of the problem. If we forget about the ordering, then the problem can be solved, even for arbitrary polynomials.

Theorem 7.13 (Weyl [112]). *There is a Turing machine which, given some $n \geq 1$ and a $\rho_{\mathbb{C}}^{n+1}$-name of the coefficients of a polynomial*

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0 \in \mathbb{C}[z] \text{ with } a_n \neq 0,$$

computes a $\rho_{\mathbb{C}}^n$-name of a vector in $\mathbb{C}^n$ containing all zeros of p with correct multiplicity.

For any fixed $n \geq 1$, this can be expressed also as follows: the multi-valued function that associates with any polynomial of degree n all permutations of its n zeros (counted with multiplicity) is computable. One could formulate this also without using the notion of a multi-valued function if instead of $\mathbb{C}^n$ as the output space one uses the computable metric space $\mathbb{C}^n/\mathcal{S}_n$, where $\mathcal{S}_n$ is the group of permutations of n elements; see Specker [97]. Note the difference to the negative statement above: the algorithm described in Theorem 7.13 still computes a vector of zeros when given a (standard description of a) polynomial, but the order in which the zeros appear in the vector may depend on the description of the polynomial given to the algorithm. It does not need to be the same for all descriptions of the polynomial (as it would have to be for an algorithm computing a Z as in Lemma 7.12).

By Proposition 4.18, Theorem 7.13 implies that the zeros of any polynomial with computable coefficients are computable again; compare Theorem 3.4. We conclude this discussion with the remark that it is still an open problem to determine the precise "degree of discontinuity" or "topological complexity" of functions Z as in Lemma 7.12; compare Smale [94], and Vassiliev [104]. More on "degrees of discontinuity" can be found in Hertling [44, 45, 46].

Now we wish to consider computational problems that receive not only numbers and functions as input but also subsets of $\mathbb{R}^n$. Therefore, we need representations of subsets.

Definition 7.14. Let (X, d, α) be a computable metric space.

1. We define a representation ϑ of all open subsets of X by

$$\vartheta(p) = U :\iff p \text{ enumerates a set } A \subseteq \mathbb{N} \times \mathbb{Q}_+ \text{ such that}$$
$$U = \bigcup_{(n,\varepsilon) \in A} \mathrm{B}(\alpha(n), \varepsilon).$$

2. We define three representations of the set $\mathcal{A}(X) := \{A \subseteq X \mid A \text{ is closed}\}$ by

   a) $\psi_>(p) = A :\iff \vartheta(p) = X \setminus A$,

b) $\psi_<(p) = A :\iff$ p enumerates all pairs $(n, \varepsilon) \in \mathbb{N} \times \mathbb{Q}_+$ with $A \cap \mathrm{B}(\alpha(n), \varepsilon) \neq \emptyset$,

c) $\psi_= := \psi_< \wedge \psi_>$.

Note that the ϑ-computable sets are exactly the c.e. open sets, that the $\psi_>$-computable elements of $\mathcal{A}(\mathbb{R}^n)$ are exactly the co-c.e. closed subsets of $\mathbb{R}^n$, that the $\psi_<$-computable elements of $\mathcal{A}(\mathbb{R}^n)$ are exactly the c.e. closed subsets of $\mathbb{R}^n$, and that the $\psi_=$-computable elements of $\mathcal{A}(\mathbb{R}^n)$ are exactly the computable closed subsets of $\mathbb{R}^n$. The characterizations of Propositions 5.9 and 5.12 can be generalized to the case of (certain) computable metric spaces, and they even hold uniformly (Brattka and Presser [21]). Proposition 5.17 can be formulated uniformly, which gives the first two of the statements in the following lemma. The counterexample in Example 5.18 implies the noncomputability part of the third statement.

Lemma 7.15. *1. The union operator $\cup : \mathcal{A}(X) \times \mathcal{A}(X) \to \mathcal{A}(X)$ is computable with respect to $\psi_<$, $\psi_>$, and $\psi_=$; i.e., for any $\psi \in \{\psi_<, \psi_>, \psi_=\}$, it is $([\psi, \psi], \psi)$-computable.*

2. The intersection operator $\cap : \mathcal{A}(X) \times \mathcal{A}(X) \to \mathcal{A}(X)$ is computable with respect to $\psi_>$.

3. But the intersection operator $\cap$ is not computable with respect to $\psi_<$ or $\psi_=$. It is not even $([\psi_=, \psi_=], \psi_<)$-continuous.

The representations $\psi_>, \psi_<, \psi_=$ are admissible with respect to natural topologies on $\mathcal{A}(X)$.

Lemma 7.16. *The representation $\psi_<$ is admissible with respect to the lower Fell topology, the representation $\psi_>$ is admissible with respect to the upper Fell topology, and the representation $\psi_=$ is admissible with respect to the Fell topology (which is the topology generated by the lower and the upper Fell topology).*

We use these representations in order to formulate an effective version of the Riemann Mapping Theorem. For completeness sake, we remind the reader of its statement. It says that for a subset U of the complex plane $\mathbb{C}$, the following two conditions are equivalent:

1. U is a nonempty, proper, open, connected, and simply connected subset of $\mathbb{C}$.
2. There exists a conformal function $f : D \to \mathbb{C}$ with $f(D) = U$.

Here $D := \{z \in \mathbb{C} : ||z|| < 1\}$ is the open unit disk in the complex plane. We identify $\mathbb{C}$ with $\mathbb{R}^2$ again, and we use the representation $\rho_\mathbb{C} = \rho^2$ for $\mathbb{C}$.

Theorem 7.17 (Hertling [47]).

1. Given a $[\rho_\mathbb{C} \to \rho_\mathbb{C}]$-name of a conformal mapping defined on D, one can compute a ϑ-name of $U := f(D)$ and a $\psi_<$-name of the boundary of U.

2. *Given a ϑ-name of some nonempty, proper, open, connected, and simply connected subset U of $\mathbb{C}$, a $\psi_<$-name of its boundary and a $\rho_{\mathbb{C}}$-name of a point z_0 in U, one can compute a $[\rho_{\mathbb{C}} \to \rho_{\mathbb{C}}]$-name of the uniquely determined conformal mapping $f : D \to \mathbb{C}$ with $f(D) = U$, with $f(0) = z_0$, and with $f'(0) > 0$.*

By looking at computable objects (see Corollary 4.19), one arrives at the following nonuniform version.

Corollary 7.18 (Hertling [47]). *For $U \subseteq \mathbb{C}$ the following are equivalent:*

1. *There exists a computable holomorphic bijection $f : D \to U$.*
2. *U is a nonempty, proper, open, connected, simply connected subset of $\mathbb{C}$ that is c.e. open and has a c.e. closed boundary ∂U.*

Next, we formulate an effective version of the Baire Category Theorem. The classical result can be stated as follows: if X is a complete metric space and $(A_n)_{n\in\mathbb{N}}$ a sequence of closed nowhere dense subsets of X, then $X \setminus \bigcup_{n\in\mathbb{N}} A_n$ is nonempty; i.e., it contains at least one point. In order to formulate the uniform version, we need a representation of a sequence of closed sets.

Definition 7.19. Let (X, δ) be a represented set. Then by

$$\delta^\omega(p) = (x_n)_{n\in\mathbb{N}} :\iff \text{for each } n \in \mathbb{N}, \text{ the sequence } (i \mapsto p(\langle n, i\rangle)) \text{ is a } \delta\text{-name of } x_n,$$

one defines a natural representation of the space of all sequences $(x_n)_{n\in\mathbb{N}}$ of elements in X.

Theorem 7.20 (Brattka [15]). *Let X be a complete computable metric space. The multi-valued function that associates with any sequence $(A_n)_{n\in\mathbb{N}}$ of closed nowhere dense subsets $A_n \subseteq X$ all sequences $(x_i)_{i\in\mathbb{N}}$ that are dense in $X \setminus \bigcup_{n=0}^{\infty} A_n$ is $(\psi_>^\omega, \delta_X^\omega)$-computable.*

Corollary 7.21 (Yasugi et al. [114]). *Let X be a complete computable metric space. For any computable sequence $(A_n)_{n\in\mathbb{N}}$ of co-c.e. closed nowhere dense subsets A_n of X, there exists a computable sequence $(x_i)_{i\in\mathbb{N}}$ that is dense in $X \setminus \bigcup_{n=0}^{\infty} A_n$.*

If X has no isolated points, then all singletons $\{x\}$ are nowhere dense and we obtain the following corollary, which is a generalization of Proposition 3.9.

Corollary 7.22 (Brattka [15]). *If X is a complete computable metric space without isolated points, then there is no computable sequence $(x_n)_{n\in\mathbb{N}}$ such that the set $\{x_n \mid n \in \mathbb{N}\}$ contains all computable points of X.*

The following statement can be proved directly, but it can also be deduced from Corollary 7.21.

Corollary 7.23 (Brattka [15]). *There exists a computable but nowhere differentiable function* $f : [0,1] \to \mathbb{R}$.

With the Intermediate Value Theorem we have seen a result that has a nonuniform computable version but no general uniform computable version. The Riemann Mapping Theorem and the Baire Category Theorem both admit fully uniform computable versions. The Brouwer Fixed Point Theorem is an example of a theorem that does not even admit a nonuniform computable version. It states that any continuous function $f : [0,1]^2 \to [0,1]^2$ possesses a fixed point. This statement does not hold true computationally: there is the following surprising negative result.

Theorem 7.24 (Orevkov [78], Baigger [4]). *There exists a computable function* $f : [0,1]^2 \to [0,1]^2$ *without a computable fixed point.*

We conclude this section with another interesting result in the context of the Fixed Point Theorem. Let us call a subset $A \subseteq [0,1]^n$ *fixable* if there exists a computable function $f : [0,1]^n \to [0,1]^n$ such that $f(A) = A$.

Theorem 7.25 (Miller [67]). *Let* $A \subseteq [0,1]^n$ *be a co-c.e. closed set. Then the following are equivalent:*

1. A *is fixable,*
2. A *contains a nonempty co-c.e. closed connected component,*
3. A *contains a nonempty co-c.e. closed connected subset,*
4. $f(A)$ *contains a computable real for all computable* $f : [0,1]^n \to \mathbb{R}$.

8 Computability of Linear Operators

Many problems in numerical analysis can be expressed by linear operators over normed spaces or even Banach spaces.

We start our selection of observations concerning computability of linear operators with several simple facts about linear (or affine) mappings on Euclidean spaces. The first lemma is pretty obvious.

Lemma 8.1. *Consider a matrix* $A \in \mathbb{R}^{m \times n}$. *The linear function mapping any vector* $x \in \mathbb{R}^n$ *to* $A \cdot x \in \mathbb{R}^m$ *is computable if, and only if,* A *is computable, i.e., if, and only if, all coefficients of* A *are computable real numbers.*

If A is invertible, then the range of this linear function is equal to $\mathbb{R}^m$. Otherwise, the problem to decide whether a given vector $b \in \mathbb{R}^m$ is in the range of this function is discontinuous and therefore unsolvable. But even if one knows in advance that b is in the range, computing a preimage is impossible in general, due to a well-known discontinuity.

Proposition 8.2. *The multi-valued function that associates with a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathrm{range}(x \mapsto Ax) \subseteq \mathbb{R}^m$ all solutions $x \in \mathbb{R}^n$ of the equation $A \cdot x = b$ is not $(\rho^{m(n+1)}, \rho^n)$-continuous. This implies that no Turing machine, given (a $\rho^{m\cdot n}$-name of) A and (a ρ^m-name of) b, would compute (a ρ^n-name of) a solution x to $A \cdot x = b$.*

This changes if one knows the dimension of the solution space.

Theorem 8.3 (Ziegler and Brattka [120]). *Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $b \in \mathbb{R}^m$ a vector, and let $L := \{x \in \mathbb{R}^n \mid A \cdot x = b\}$ be the affine solution space of the corresponding linear equation. If A and b are computable, then L is a computable closed set. Given names with respect to ρ of some A and b such that L is not empty and given the dimension d of L, we can even compute a ψ-name of L. This implies that we can computably find a solution $x \in L$ and in case $d > 0$ also a basis of the homogeneous solution space $\{x \in \mathbb{R}^n \mid A \cdot x = 0\}$.*

In fact, instead of a $\nu_{\mathbb{N}}$-name of the dimension, a $\rho_<$-name of the dimension is sufficient.

Now let us turn to linear operators over normed spaces. Many normed spaces used in the practice of numerical computation are computable according to the following definitions. In the following, $\mathbb{F}$ is either $\mathbb{R}$ or $\mathbb{C}$, and for computation purposes, we identify $\mathbb{C}$ with $\mathbb{R}^2$, and use $\alpha_{\mathbb{C}} := \alpha_{\mathbb{R}^2}$.

Definition 8.4. $(X, ||\ ||, e)$ is called a *computable normed space*, if

1. X is a linear space,
2. $||\ || \to \mathbb{R}$ is a norm on X,
3. the linear span of the sequence $e : \mathbb{N} \to X$ is dense in X,
4. (X, d, α_e) with $d(x, y) := ||x - y||$ and $\alpha_e\langle k, \langle n_0, \ldots, n_k\rangle\rangle := \sum_{i=0}^{k} \alpha_{\mathbb{F}}(n_i)e_i$ is a computable metric space.

Note that in any computable normed space, the point 0 is a computable point and that vector space addition and scalar multiplication are automatically computable operations.

In the following, whenever we say "computable" in connection with a computable normed space X, we always mean computable with respect to the Cauchy representation δ_X of X.

Theorem 8.5. *Let X, Y be computable normed spaces, let $T : X \to Y$ be a linear operator, and let $(s_n)_{n\in\mathbb{N}}$ be a computable sequence in X whose linear span is dense in X. Then the following are equivalent:*

1. $T : X \to Y$ is computable,

2. $(T(s_n))_{n\in\mathbb{N}}$ *is a computable sequence and* T *is bounded,*
3. T *maps computable sequences to computable sequences and is bounded,*
4. $\mathrm{graph}(T)$ *is a computable closed subset of* $X \times Y$ *and* T *is bounded,*
5. $\mathrm{graph}(T)$ *is an c.e. closed subset of* $X \times Y$ *and* T *is bounded.*

In case X *and* Y *are even Banach spaces, one can omit boundedness in the last two conditions, because it follows from the Closed Graph Theorem.*

Examples 8.6. We use the norm $||f|| := \sup_{x\in[0,1]} |f(x)|$ on $\mathcal{C}[0,1]$ and the norm $||f||_1 := ||f|| + ||f'||$ on $\mathcal{C}^1[0,1]$.

1. The integration operator
$$I : \mathcal{C}[0,1] \to \mathcal{C}[0,1], f \mapsto \left(x \mapsto \int_0^x f(t)\,\mathrm{d}t\right)$$
is a linear computable operator.
2. The differentiation operator
$$d :\subseteq \mathcal{C}[0,1] \to \mathcal{C}[0,1], f \mapsto f',$$
defined for continuously differentiable functions on $[0,1]$, is a linear unbounded operator with a closed graph.
3. The differentiation operator
$$D : \mathcal{C}^1[0,1] \to \mathcal{C}[0,1], f \mapsto f'$$
is a computable linear operator. This is not surprising because a Cauchy name of a function $f \in \mathcal{C}^1[0,1]$ contains already the information needed for computing f'.
4. The differentiation operator
$$D :\subseteq \mathcal{C}^2[0,1] \to \mathcal{C}[0,1], f \mapsto f'$$
is computable if for the elements of $\mathcal{C}^2[0,1]$ one uses the representation δ defined by
$$\delta\langle p,q\rangle = f :\iff \delta_{\mathcal{C}[0,1]}(p) = f \text{ and } \rho(q) > ||f''||.$$

Corollary 8.7. *If a computable function* $f \in \mathcal{C}[0,1]$ *has a continuous second derivative, then* f' *is computable as well.*

We had remarked at the end of Section 4 that a noncomputable function may nevertheless be computably invariant; i.e., map computable elements to computable elements, and if it is not computably invariant, it can be a challenge to construct a computable element of the domain that is mapped to a noncomputable element. Pour-El and Richards [80] have shown a general result that shows that for linear operators the situation is simpler.

Theorem 8.8 (Pour-El and Richards [80]). *Let X, Y be computable normed spaces, let $T : X \to Y$ be a linear operator with closed graph, and let $(s_n)_{n\in\mathbb{N}}$ be a computable sequence in X whose linear span is dense in X and such that $(T(s_n))_{n\in\mathbb{N}}$ is a computable sequence in Y. Then T is unbounded if, and only if, there exists a computable $x \in X$ such that $T(x)$ is noncomputable.*

We have seen that there exists a computable nowhere differentiable function. The result by Pour-El and Richards, applied to the differentiation operator in Example 8.6.2, gives the following observation.

Corollary 8.9 (Myhill [76]). *There exists a computable and continuously differentiable function $f : [0, 1] \to \mathbb{R}$ whose derivative $f' : [0, 1] \to \mathbb{R}$ is not computable.*

A striking application of the theorem by Pour-El and Richards is the following statement concerning the three-dimensional wave equation.

Theorem 8.10 (Pour-El and Richards [79, 80]). *There exists a computable function $f : \mathbb{R}^3 \to \mathbb{R}$ such that the unique solution u of the three-dimensional wave equation*

$$\begin{cases} u_{tt} = \Delta u \\ u(0, x) = f,\ u_t(0, x) = 0,\ t \in \mathbb{R}, x \in \mathbb{R}^3 \end{cases}$$

has the following properties: the wave $x \mapsto u(0, x) = f(x)$ at time 0 is computable, but the wave $x \mapsto u(1, x)$ at time 1 is not computable.

This result has led to a number of misinterpretations because it appears as if the wave equation produces some noncomputability that emerges out of nothing. However, a careful analysis of the result shows that the noncomputability in the counterexample is already hidden in the derivative of the initial condition, and hence, the example can be considered as yet another instance of Corollary 8.9. The solvability of the wave equation was analyzed further by Washihara [105] and by Weihrauch and Zhong [111].

Theorem 8.11 (Weihrauch and Zhong [111]). *The solution operator of the wave equation $S : \mathcal{C}^k(\mathbb{R}^3) \to \mathcal{C}^{k-1}(\mathbb{R}^4), f \mapsto u$ is computable.*

Remark 8.12. Weihrauch and Zhong [111] proved also that the operator of wave propagation is computable without any loss of the degree of differentiability if physically appropriate Sobolev spaces are used.

The next result deals with the Hahn–Banach Theorem.

Theorem 8.13 (Metakides and Nerode [64]). *Let X be a finite-dimensional computable Banach space with some c.e. closed linear subspace $Y \subseteq X$. For any computable linear functional $f : Y \to \mathbb{R}$ with computable norm $||f||$, there exists a computable linear extension $g : X \to \mathbb{R}$ with $||g|| = ||f||$.*

It follows from results of Metakides et al. [65] that this result cannot be extended to infinite-dimensional spaces. However, it is worth mentioning that for normed spaces X with strictly convex dual X', the norm-preserving extension of functionals is always unique, and in the unique case, the extension can always be computed (even uniformly, that is given f one can compute a suitable g); see Brattka [18]. This class of spaces with strictly convex dual includes, for instance, all Hilbert spaces.

Many results from functional analysis can be analyzed with respect to their computational content; see, e.g., Pour-El and Richards [80], Brattka and Dillhage [19], and Brattka and Yoshikawa [22].

Often, a computational problem can be formulated as the problem to compute the inverse of a given linear operator. This poses the question under which circumstances the inverse of a linear operator can be computed. Let X, Y be Banach spaces, and let $T : X \to Y$ be a linear operator. By Banach's Inverse Mapping Theorem, any linear, bijective, and bounded T has a bounded inverse T^{-1}. If we can compute T, can we compute T^{-1} as well? Interestingly, here we have a positive nonuniform answer but a negative uniform answer.

Theorem 8.14 (Brattka [16]).

1. *Let X, Y be Banach spaces, and let $T : X \to Y$ be a linear operator. If T is bijective and computable, then T^{-1} is computable as well.*
2. *There exist Banach spaces X and Y such that the function that maps any bijective linear operator $T : X \to Y$ to its inverse $T^{-1} : Y \to X$ is not $([\delta_X \to \delta_Y], [\delta_Y \to \delta_X])$-continuous.*

We conclude this section with an interesting application of the first, positive, statement of the previous theorem to the theory of differential equations. Although the first statement of the previous theorem is nonuniform, the following application is a uniform statement. Here we represent $\mathcal{C}^n[0,1]$ by the Cauchy representation for the norm $||f||_n := \sum_{i=0}^n ||f^{(i)}||$ that includes information on all the derivatives.

Theorem 8.15 (Brattka [16]). *Let $n \geq 1$, and let $f_0, \ldots, f_n : [0,1] \to \mathbb{R}$ be computable with $f_n \neq 0$. The solution operator $L : \mathcal{C}[0,1] \times \mathbb{R}^n \to \mathcal{C}^n[0,1]$ that maps each $(y, a_0, \ldots, a_{n-1}) \in \mathcal{C}[0,1] \times \mathbb{R}^n$ to the unique $x \in \mathcal{C}^n[0,1]$ such that*

$$\sum_{i=0}^{n} f_i(t)x^{(i)}(t) = y(t) \text{ with } x^{(j)}(0) = a_j \text{ for } j = 0, \ldots, n-1,$$

is computable.

Proof. The following operator is linear and computable:

$$L^{-1} : \mathcal{C}^n[0,1] \to \mathcal{C}[0,1] \times \mathbb{R}^n, x \mapsto \left(\textstyle\sum_{i=0}^n f_i x^{(i)}, x^{(0)}(0), \ldots, x^{(n-1)}(0)\right).$$

By the computable Inverse Mapping Theorem it follows that L is computable too.

□

9 Degrees of Unsolvability

In this section we describe several approaches for classifying unsolvable computational problems over the real numbers. First, remember that computability of real number functions can be expressed as effective continuity. We generalize this notion using the Borel hierarchy. In the second approach one looks only at computable input and uses the arithmetical hierarchy for the classification. Finally, we ask about Turing degrees of real number functions and look at a reducibility relation on functions on computable metric spaces, which is introduced in a natural way using Cauchy representations.

In Example 8.6 we have seen that the differentiation operator

$$d :\subseteq \mathcal{C}[0,1] \to \mathcal{C}[0,1], f \mapsto f',$$

defined for continuously differentiable functions on $[0,1]$, is a linear unbounded operator and therefore not computable. Can one characterize the degree of noncomputability of differentiation more precisely? In order to do this we remind the reader of the (finite beginning of the) Borel hierarchy. In the following, X and Y are computable metric spaces unless stated otherwise. Let $\mathbf{\Sigma}^0_1(X)$ be the set of all open subsets of X, and for $k \geq 1$,

$$\mathbf{\Pi}^0_k(X) \text{ be the set of all complements of sets in } \mathbf{\Sigma}^0_k(X),$$
$$\mathbf{\Sigma}^0_{k+1}(X) \text{ be the set of all countable unions of sets in } \mathbf{\Pi}^0_k(X).$$

This hierarchy can be extended to transfinite levels. Here, we will consider only finite levels. One can define representations of the finite levels of the Borel hierarchy in a straightforward manner.

Definition 9.1. Let (X, d, α) be a computable metric space. For $k \geq 1$, we define representations $\delta_{\mathbf{\Sigma}^0_k(X)}$ of $\mathbf{\Sigma}^0_k(X)$ and $\delta_{\mathbf{\Pi}^0_k(X)}$ of $\mathbf{\Pi}^0_k(X)$ as follows:

$$\delta_{\mathbf{\Sigma}^0_1(X)}(p) := \bigcup_{(i,\varepsilon)\in\mathbb{N}\times\mathbb{Q}_+ \text{ enumerated by } p} \mathrm{B}(\alpha(i),\varepsilon),$$

$$\delta_{\mathbf{\Sigma}^0_{k+1}(X)}(p) := \bigcup_{i=0}^{\infty} \delta^{\omega}_{\mathbf{\Pi}^0_k(X)}(p)(i),$$

$$\delta_{\mathbf{\Pi}^0_k(X)}(p) := X \setminus \delta_{\mathbf{\Sigma}^0_k(X)}(p).$$

The representation $\delta_{\mathbf{\Sigma}^0_1(X)}$ is identical to the representation ϑ of open subsets as defined earlier.

Definition 9.2. Let X, Y be computable metric spaces. A function $f :\subseteq X \to Y$ is $\mathbf{\Sigma}^0_k$*-measurable* if for any $U \in \mathbf{\Sigma}^0_1(Y)$ there is a set $V \in \mathbf{\Sigma}^0_k(X)$ with $f^{-1}(U) = V \cap \mathrm{dom}(f)$. It is called *effectively* $\mathbf{\Sigma}^0_k$*-measurable* or $\mathbf{\Sigma}^0_k$*-computable*, if a Turing machine, given a $\delta_{\mathbf{\Sigma}^0_1(X)}$-name of such a U, computes a $\delta_{\mathbf{\Sigma}^0_k(X)}$-name of such a V.

Remark 9.3. 1. A function $f :\subseteq X \to Y$ is $\mathbf{\Sigma}^0_1$-measurable if, and only if, it is continuous.

2. In Theorem 4.27 we have seen that the computable real number functions are exactly the real number functions that are effectively continuous in a certain sense. Using this, one can easily show that they are exactly the $\mathbf{\Sigma}^0_1$-computable real number functions. And this can easily be generalized to functions between computable metric spaces: a function $f :\subseteq X \to Y$ is computable (with respect to the Cauchy representations) if, and only if, it is $\mathbf{\Sigma}^0_1$-computable.
3. The total $\mathbf{\Sigma}^0_k$-computable functions $f : X \to Y$ are in fact the $\mathbf{\Sigma}^0_k$-recursive functions in the sense of Moschovakis [71]; see Brattka [17, page 24].

The following result is an extension of Theorem 6.14.

Theorem 9.4 (Brattka [17]). *Let X, Y be computable metric spaces and k be a positive integer. Then a total function $f : X \to Y$ is $\mathbf{\Sigma}^0_k$-measurable ($\mathbf{\Sigma}^0_k$-computable) if, and only if, there is a $\mathbf{\Sigma}^0_k$-measurable ($\mathbf{\Sigma}^0_k$-computable) function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ with $f\delta_X(p) = \delta_Y F(p)$ for all $p \in \mathrm{dom}(f\delta_X)$.*

Because of this result, it is reasonable to extend the definition of $\mathbf{\Sigma}^0_k$-computability to arbitrary represented spaces as follows.

Definition 9.5. Let X, Y be sets with representations δ_X, δ_Y, respectively. Then a function $f :\subseteq X \to Y$ is called $\mathbf{\Sigma}^0_k$-*computable* (with respect to (δ_X, δ_Y)) if, and only if, it has a $\mathbf{\Sigma}^0_k$-computable realizer.

Due to the previous theorem, this is a conservative extension of the concept of $\mathbf{\Sigma}^0_k$-computability. With respect to a suitable reducibility relation, for each $k \geq 1$, there are complete problems in the class of $\mathbf{\Sigma}^0_k$-computable functions.

Definition 9.6. Consider two functions $f :\subseteq \Sigma^\omega \to \Sigma^\omega$ and $g :\subseteq \Sigma^\omega \to \Sigma^\omega$. We say that f is *computably reducible* to g, written $f \leq_{\mathrm{c}} g$, if there are computable functions $A :\subseteq \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$ and $B :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that

$$f(p) = A(p, g(B(p)))$$

for all $p \in \mathrm{dom}(f)$.

A topological version of this reducibility relation, also on more general topological spaces, has been studied by Hirsch [51], Weihrauch [108], and Hertling [44, 46].

Proposition 9.7 (Brattka [17]). *For $k \geq 0$, there exist functions $C_k :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that any $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ is $\mathbf{\Sigma}^0_{k+1}$-computable if, and only if, F is computably reducible to C_k.*

We wish to compare not only functions on Σ^ω but on more general spaces. Transferring Definition 9.6 in a straightforward manner to functions $f :\subseteq X \to Y$ and $g :\subseteq U \to V$ does not look very promising because then it becomes important whether the spaces are connected. For example, in case $X = \mathbb{R}$, and $U = \Sigma^\omega$, there is no nontrivial continuous function $B : X \to U$. Instead, we remember that computations are performed on names.

Definition 9.8. Let X, Y, U, V be sets with representations $\delta_X, \delta_Y, \delta_U, \delta_V$, respectively, and let $f :\subseteq X \to Y$ and $g :\subseteq U \to V$ be functions. We say that f is *computably realizer reducible* to g, written $f \leq_{\mathrm{c}} g$, if there are computable functions $A :\subseteq \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$ and $B :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that for any (δ_U, δ_V)-realizer G of g, the function

$$p \mapsto A(p, G(B(p)))$$

is a (δ_X, δ_Y)-realizer of f.

Definition 9.9. Let X, Y be sets with representations δ_X, δ_Y, respectively. We say that a function $f :\subseteq X \to Y$ is $\mathbf{\Sigma}^0_k$*-complete* (with respect to (δ_X, δ_Y)), if it is $\mathbf{\Sigma}^0_k$-computable with respect to (δ_X, δ_Y) and any other $\mathbf{\Sigma}^0_k$-computable function is computably realizer reducible to it.

Note that the functions C_k whose existence is stated in Proposition 9.7 are $\mathbf{\Sigma}^0_{k+1}$-complete. A linear operator satisfying the assumptions in Theorem 8.8 by Pour-El and Richards is at least as difficult to compute as C_1.

Theorem 9.10 (Brattka [14]). *Let X, Y be computable Banach spaces, and let $f :\subseteq X \to Y$ be a linear and unbounded operator with closed graph. Let $(s_n)_{n\in\mathbb{N}}$ be a computable sequence in $\mathrm{dom}(f)$ whose linear span is dense in X (note that this makes $\mathrm{dom}(f)$ a computable metric space), and let $(f(s_n))_{n\in\mathbb{N}}$ be a computable sequence in Y. Then $C_1 \leq_{\mathrm{c}} f$.*

This can be used to prove that certain operators such as differentiation are $\mathbf{\Sigma}^0_2$-complete. For nonlinear maps, other techniques are required (see for instance Brattka [14] and Gherardi [40]).

Example 9.11. The following functions are examples of complete functions:

1. The differentiation $d :\subseteq \mathcal{C}[0,1] \to \mathcal{C}[0,1], f \mapsto f'$ is $\mathbf{\Sigma}^0_2$-complete.
2. The limit map $\lim :\subseteq \mathbb{R}^{\mathbb{N}} \to \mathbb{R}, (x_n)_{n\in\mathbb{N}} \mapsto \lim_{n\to\infty} x_n$ is $\mathbf{\Sigma}^0_2$-complete.
3. The boundary operation $\partial : \mathcal{A}(\mathbb{R}^n) \to \mathcal{A}(\mathbb{R}^n), A \mapsto \partial A$ is $\mathbf{\Sigma}^0_2$-complete.
4. The derived set operator $D : \mathcal{A}(\mathbb{R}^n) \to \mathcal{A}(\mathbb{R}^n), A \mapsto A'$ is $\mathbf{\Sigma}^0_3$-complete.

Here we assume that $\mathcal{A}(\mathbb{R}^n)$ is represented by ψ.

It is worth mentioning that any $\mathbf{\Sigma}^0_{k+1}$-complete map has the property that it maps some computable input to some Δ^0_{k+1}-computable output that is not Δ^0_k-computable (Brattka [17]). Here the light face classes Δ^0_{k+1} refer to the arithmetical hierarchy. Applied to the differentiation operator d, it gives us yet another proof of Corollary 8.9. Applied to the limit map, it yields a computably approximable real number that is not computable and applied to the boundary operator it yields a computable closed set whose boundary is not a computable closed set.

This brings us to the second approach for classifying unsolvable problems that we wish to describe. It uses the arithmetical hierarchy. Cenzer and Remmel [32] consider operators whose input are real number functions. They use a Gödel numbering of computable real number functions and characterize the set of Gödel numbers of computable input functions that are mapped by the operator to computable output. If the operator is computable, this is of course a trivial question because we know that computable operators are computably invariant; i.e., they map every computable input to computable output. The question becomes interesting for noncomputable operators. We define a Gödel numbering of the total computable functions $f : \mathbb{R}^n \to \mathbb{R}$, for each $n \geq 1$. In the following definition, φ denotes a total standard numbering of all computable functions $F :\subseteq \mathbb{N} \to \mathbb{N}$; i.e., $(\varphi_i)_{i\in\mathbb{N}}$ is a list containing exactly all computable functions $F :\subseteq \mathbb{N} \to \mathbb{N}$ such that φ satisfies the utm property and the smn property; compare any book on computability theory, e.g., Rogers [84] and Weihrauch [106].

Definition 9.12. Fix some $n \geq 1$. We write $F^n_e = f$ if the c.e. set $S := \mathrm{dom}(\varphi_e)$ describes a total effectively continuous function $f : \mathbb{R}^n \to \mathbb{R}$ as in Definition 4.26. Furthermore, $I^n := \mathrm{dom}(F^n)$.

By Theorem 4.27, F^n is a numbering of all total computable functions $f : \mathbb{R}^n \to \mathbb{R}$. Actually, Cenzer and Remmel [32] used a slightly different numbering. But the two numberings are equivalent, with the consequence that the following theorem holds also for our numbering F^n. The following theorem lists several results by Cenzer and Remmel [32] related to computational problems that we considered earlier.

Theorem 9.13 (Cenzer and Remmel [32]). *The following sets are Σ^0_3-complete.*

1. *The set of all $e \in I^1$ such that F^1_e is differentiable and the derivative is computable.*
2. *The set of all $e \in I^2$ such that the differential equation $y'(x) = F^2_e(x, y(x))$ with initial condition $y(0) = 0$ has a computable solution on the interval $[-\delta, \delta]$, for some $\delta > 0$.*
3. *The set of all $e \in I^3$ such that the wave equation*

$$u_{xx} + u_{yy} + u_{zz} - u_{tt} = 0$$

with initial conditions $u_t(x, y, z, 0) = 0$ and $u(x, y, z, 0) = F^3_i(x, y, z)$ has a computable solution.

We note that Cenzer and Remmel also found natural Π^0_3-complete problems and Σ^0_4-complete problems over the real numbers [32].

Finally, we describe work by Miller [68] concerning degrees of uncomputability of real number functions and functions over computable metric spaces. A large part of classic computability theory deals with Turing reducibility and Turing degrees. Thus, in the context of computable analysis, it is natural to try to characterize the degree of uncomputability of a real number function by asking for its Turing degree. Let X be a computable metric space with Cauchy representation δ_X. It is natural to define the *Turing degree* $\deg_T(x)$ of an element $x \in X$ by

$$\deg_T(x) := \min\{\deg_T(p) \mid \delta_X(p) = x\}.$$

Indeed, for $X := \mathbb{R}$ and any real number $x \in \mathbb{R}$, the Turing degree $\deg_T(x)$ is well defined and coincides with the Turing degree of the fractional part of the binary expansion of x. But this approach does not work well anymore for real number functions.

Theorem 9.14 (Miller [68]). *Consider $X := \mathcal{C}[0,1]$. There exists some $f \in X$ such that $\deg_T(f)$ is not well defined; i.e., there is no Cauchy name for f with minimal Turing degree.*

So, another approach is needed. Turing reducibility is defined for subsets of $\mathbb{N}$ or for their characteristic sequences: a sequence $p \in \{0,1\}^\omega$ is Turing reducible to a sequence $q \in \{0,1\}^\omega$ if there is a computable function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ with $F(q) = p$. By using Cauchy representations, one can translate this reducibility to elements of computable metric spaces.

Definition 9.15. Let X and Y be computable metric spaces. An element $x \in X$ is *representation reducible* to an element $y \in Y$ if there is a computable (i.e., (δ_X, δ_Y)-computable) function $f :\subseteq Y \to X$ with $f(y) = x$.

This relation is clearly reflexive and transitive. Let us call the equivalence classes of elements of computable metric spaces under this reducibility relation *continuous degrees*.

Proposition 9.16 (Miller [68]). *Every continuous degree contains a real-analytic function.*

Theorem 9.17 (Miller [68]). *There is a natural, nontrivial embedding from the Turing degrees into the continuous degrees and a natural, nontrivial embedding from the continuous degrees into the enumeration degrees.*

10 Computational Complexity of Real Numbers and Real Number Functions

In the previous section we discussed unsolvable computational problems over the real numbers. In this and the following section, we turn in the other direction: we analyze the computational complexity of solvable problems. The two most important complexity measures in classical complexity theory are time complexity and space complexity. We will restrict ourselves to time complexity. We wish to classify computational problems over the real numbers according to the time one needs in order to solve them. We measure the time by counting the steps of a Turing machine. In classic complexity theory, discrete problems are considered, and the time complexity is usually analyzed in dependence of the input size of the concrete instance of the computational problem. Often, this input size corresponds quite well to the input dimension of the concrete instance of the problem. But for continuous problems whose output is real-valued, another parameter turns out to be at least as important in the practice of numerical analysis: the desired precision of the output value. In fact, for a single real number, this is the only parameter on which the time can depend. And for any fixed real number function, the dimension is fixed as well. This has the consequence that the time complexity of computable real numbers and computable real number functions is usually analyzed as a function of the desired output precision. In the following definition, we use a standard binary notation $\nu_{\mathbb{D}}$ of the set

$$\mathbb{D} := \{z/2^n \mid z \in \mathbb{Z},\ n \in \mathbb{N}\}$$

of dyadic rational numbers, defined by

$$\nu_{\mathbb{D}}(a_{-k} \ldots a_0 \cdot a_1 a_2 \ldots a_n) := \sum_{i=-k}^{n} a_i \cdot 2^{-i} \quad \text{and}$$

$$\nu_{\mathbb{D}}(-a_{-k} \ldots a_0 \cdot a_1 a_2 \ldots a_n) := -\sum_{i=-k}^{n} a_i \cdot 2^{-i},$$

where $k, n \in \mathbb{N}$, $a_i \in \{0,1\}$, and $a_{-k} = 0 \Rightarrow k = 0$. Such a string $a_{-k} \ldots a_0 \cdot a_1 a_2 \ldots a_n$ or $-a_{-k} \ldots a_0 \cdot a_1 a_2 \ldots a_n$ with n digits after the binary point will be called a *binary name of precision* n of the corresponding dyadic rational number. Our first definition of the time complexity of computable real numbers and computable real number functions is in analogy to our first definition of computable real number functions (Definition 4.1).

Definition 10.1. Let $t : \mathbb{N} \to \mathbb{N}$ be a total function.

1. A real number x is *precision-computable in time* $\mathcal{O}(t)$, if there are a constant c and a Turing machine that, given any $n \in \mathbb{N}$ in binary notation, computes within $c \cdot t(n) + c$ steps a binary name of precision n of a dyadic rational number q with $|x - q| < 2^{-n}$.

2. Consider a compact set $K \subseteq \mathbb{R}^k$. A function $f :\subseteq \mathbb{R}^k \to \mathbb{R}$ with $K \subseteq \mathrm{dom}(f)$ is *precision-computable in time $\mathcal{O}(t)$ on K* if there are a constant c and an oracle Turing machine that, given any $n \in \mathbb{N}$ in binary notation, may ask for arbitrarily good dyadic approximations of the input $x \in K$; i.e., it may ask finitely many questions of the kind "Give me a k-vector of binary names with precision i of the components of a dyadic vector $r \in \mathbb{D}^k$ with $\max_{l=1,\ldots,k} |x_l - r_l| < 1/2^i$," where the exponent i may depend on the answers to the previous questions, and after at most $c \cdot t(n) + c$ steps, it stops, having written onto the output tape a binary name with precision n of a dyadic rational number q with $|f(x) - q| < 2^{-k}$.

   Here, the precision i of a request as above has to be written in binary form on a special tape, the *oracle tape*, and the machine has to enter a special state, the *query state*. Then the answer will be provided in one step on the oracle tape, with the tape head on the field to the left of the answer. For reading the answer, the Turing machine will need time as for any usual reading operation on any other tape.

We consider here only functions on compact sets because for arbitrarily large input numbers, one may need arbitrarily much time just to read the digits in front of the binary point. Then in general, one will be unable to give any bound on the time needed even for computing the result with precision 1.

We had characterized computability of real number functions not only by an oracle approach as in Definition 4.1 but also via the Cauchy representation and equivalent representations. And we had seen that both approaches were equivalent. Can one say the same also for the time complexity of real number functions and of real numbers? One would say that a real number x is computable in time $t : \mathbb{N} \to \mathbb{N}$ with respect to a representation δ of $\mathbb{R}$ if a Turing machine computes a δ-name p for x in time $t(n)$; i.e., without ever stopping it writes p onto the output tape and it needs at most $t(n)$ steps for writing the first n symbols of p. Using the Cauchy representation is not a good idea because for any rational number one can easily construct arbitrarily long $\nu_{\mathbb{Q}}$-names. This has the consequence that for any computable real number x, there is a Cauchy name for x computable in linear time. Thus, a definition of time complexity of real numbers via the Cauchy representation does not make any sense. One could "repair" the Cauchy representation ρ by using names of the form $w_0\#w_1\#w_2\#\ldots$, where each w_i is not a $\nu_{\mathbb{Q}}$-name of a rational number but a binary name of precision i of a dyadic rational number. But such a name would be quite wasteful because information that is contained already in w_i for some i will be repeated in all w_j with $j > i$. The following representation is a much more elegant way of representing real numbers. It is defined just like the usual binary representation except that besides the digits 0 and 1 also a digit $\overline{1}$, standing for -1, can be used. Avizienis [3] studied it in the context of computer arithmetic. This representation as well as similar "redundant number representations" have turned out to be very important in computer arithmetic; see, e.g., Muller [73].

Definition 10.2. The *signed digit representation* $\rho_{\mathrm{sd}} :\subseteq \Sigma^\omega \to \mathbb{R}$ is defined by

$$\rho_{\mathrm{sd}}(a_n a_{n-1} \ldots a_0 \centerdot a_{-1} a_{-2} \ldots) := \sum_{i=n}^{-\infty} a_i \cdot 2^i$$

for all sequences with $a_i \in \{\overline{1}, 0, 1\}$ and $n \geq -1$, and with the additional properties that $a_n \neq 0$, if $n \geq 0$ and $a_n a_{n-1} \notin \{1\overline{1}, \overline{1}1\}$, if $n \geq 1$, where we interpret $\overline{1}$ as -1.

As in the binary representation, the number of digits after the binary point of any prefix of a name of a real number corresponds directly to the precision by which this prefix already describes the real number. But, in contrast to the binary representation, the signed digit representation is redundant in a symmetric way. If w is a prefix of a ρ_{sd}-name, then $\rho_{\mathrm{sd}}(w\Sigma^\omega)$ is a closed interval, $\rho_{\mathrm{sd}}(w\overline{1}\Sigma^\omega)$ is the left half of $\rho_{\mathrm{sd}}(w\Sigma^\omega)$, $\rho_{\mathrm{sd}}(w0\Sigma^\omega)$ covers the two middle quarters of $\rho_{\mathrm{sd}}(w\Sigma^\omega)$, and $\rho_{\mathrm{sd}}(w1\Sigma^\omega)$ is the right half of $\rho_{\mathrm{sd}}(w\Sigma^\omega)$. If additionally w contains the binary point and n digits after the binary point, then the interval $\rho_{\mathrm{sd}}(w\Sigma^\omega)$ has length $2 \cdot 2^{-n}$.

Proposition 10.3. *The signed digit representation ρ_{sd} is equivalent to the Cauchy representation of the real numbers.*

Definition 10.4. Let $t : \mathbb{N} \to \mathbb{N}$ be a total function. A real number x is *computable in time $\mathcal{O}(t)$* if there are a constant c and a Turing machine that, without ever stopping, produces a ρ_{sd}-name of x and, after $c \cdot t(n) + c$ steps, has written at least the prefix containing n digits after the binary point.

Before we compare this definition with the previous definition of "precision-computability in time $\mathcal{O}(t)$," we explain how one can define the time complexity of real number functions via the signed digit representation. As we will see soon, it is useful not only to count the steps needed by a Turing machine but also to count how many digits of the input the machine needs in order to compute the first n output symbols.

Definition 10.5. Let $t : \mathbb{N} \to \mathbb{N}$ and $l : \mathbb{N} \to \mathbb{N}$ be total functions. Consider a compact set $K \subseteq \mathbb{R}^k$. A function $f :\subseteq \mathbb{R}^k \to \mathbb{R}$ with $K \subseteq \mathrm{dom}(f)$ is *computable in time $\mathcal{O}(t)$ with input lookahead l on K* if a Turing machine with k input tapes, given ρ_{sd}-names $p_1, \ldots, p_k$ of the k components of any $x \in K$, does the following:

1. without ever stopping it computes a ρ_{sd}-name q of $F(x)$,
2. after $c \cdot t(n) + c$ steps it has written at least the prefix of q containing n symbols after the binary point,
3. until it has written this prefix of q, for any $i \in \{1, \ldots, k\}$, it has read at most the first $l(n)$ symbols after the binary point of p_i.

Lemma 10.6. *For any co-c.e. closed, compact $K \subseteq \mathbb{R}^k$ and any computable $f : K \to \mathbb{R}$ there exist computable $t, l : \mathbb{N} \to \mathbb{N}$ such that f is computable in time $\mathcal{O}(t)$ and with input lookahead l.*

Remark 10.7. In order to be able to speak about a uniform time bound for the function, we consider the function only on a compact set. For a function with, e.g., domain $\mathbb{R} = \bigcup_{i=1}^{\infty}[-i, i]$, one can consider the time complexity of f on $[-i, i]$ for each positive i and then use i as an additional parameter that corresponds to the "size" of the input.

Now let us compare the two definitions of time complexity of real numbers and real number functions that we have formulated. A real number x or a real number function $f :\subseteq \mathbb{R}^k \to \mathbb{R}$ on a compact set $K \subseteq \mathrm{dom}(f)$ is precision-computable in time $\mathcal{O}(t)$ (see Definition 10.1) if it is computable in time $\mathcal{O}(t)$ (see Definition 10.4 and Definition 10.5). In general, the converse does not need to be true. But it is true for so-called regular time bounds.

Definition 10.8. A total function $t : \mathbb{N} \to \mathbb{N}$ is *regular* if it is nondecreasing and there exist constants $N, c > 0$ such that $t(N) > 0$ and

$$2 \cdot t(n) \leq t(2n) \leq c \cdot t(n)$$

for all $n > N$.

A regular function grows polynomially and at least linearly. Any function obtained by multiplying n^k for some $k \geq 1$ by a power of logarithms or multiple logarithms is regular. For the practice of computing, the regular functions are presumably the most important time bounds.

Proposition 10.9. *Let $t : \mathbb{N} \to \mathbb{N}$ be regular. A real number x or a real number function $f : K \to \mathbb{R}$ for some compact set $K \subseteq \mathbb{R}^k$ is precision-computable in time $\mathcal{O}(t)$ if, and only if, it is computable in time $\mathcal{O}(t)$.*

Proof. We sketch the proof for the case of real numbers.

First, let us assume that x is a real number computable in time $\mathcal{O}(t)$. Then a Turing machine M_{sd} produces a signed-digit name of x in time $\mathcal{O}(t)$. Given some n in binary notation, one can simulate M_{sd} until it has computed a signed digit name of x up to $n + 2$ digits after the binary point. This prefix describes a dyadic number q_1 with denominator 2^{-n-2}. Then one computes a dyadic number q_2 with denominator 2^{-n} as close as possible to q_1 and writes its binary name with precision n onto the output tape. All this takes time $\mathcal{O}(t(n + 2) + n) \subseteq \mathcal{O}(t(n))$. Thus, x is precision computable in time $\mathcal{O}(t)$.

Now, let us assume that x is precision computable in time $\mathcal{O}(t)$. Let M_{prec} be a Turing machine which, given an n in binary form, computes a dyadic rational q with

precision n and with $|x - q| < 2^{-n}$. In order to produce a signed digit name of x, one can proceed as follows. One can simulate M_{prec} repeatedly with input 2^i for $i = 0, 1, 2, 3, \ldots$ and, after each simulation, use the new dyadic approximation of x in order to improve the previously computed prefix of a signed digit name of x. In fact, the number of computed digits after the decimal point doubles after each simulation. In order to compute the first n digits after the binary point, the new machine will need about $\mathcal{O}(\sum_{i=0}^{\lceil \log_2(n) \rceil} t(2^i))$ time. It is easy to verify that regularity of t implies that $\mathcal{O}(t)$ is an upper bound for this term. □

The following two statements are in analogy to Theorem 3.4 for computable real numbers. Let $\mathcal{M}(n) := n \cdot \log_2(n) \cdot \log_2(\log_2(n))$ be the time bound for the Schönhage–Strassen algorithm for multiplying two natural numbers given by binary words of length n.

Theorem 10.10 (Müller [74]). *Let t be a regular function with $\mathcal{M} \in \mathcal{O}(t)$. The set of real numbers computable in time $\mathcal{O}(t)$ forms a real algebraically closed field.*

Corollary 10.11 (Ko and Friedman [56, 55]). *The set of polynomial time computable real numbers forms a real algebraically closed field.*

In particular, the zeros of a polynomial whose coefficients can be computed fast, e.g., in polynomial time, can be computed fast as well, e.g., in polynomial time. The problem to find fast algorithms for computing the zeros of polynomials has received a lot of attention; see Neff and Reif [77]. Also numbers like π and e are polynomial time computable. The time complexity of π has been analyzed in great detail. Finding even faster algorithms for computing π as well as actually computing π with precision as high as possible is still a challenge; see, e.g., Borwein and Borwein [12].

Remark 10.12. We have seen in Section 3 that for defining the set of computable real numbers, the decimal (or binary) representation is as good as the Cauchy representation and therefore as good as the signed digit representation. But the set of real numbers with polynomial time computable decimal name (or with polynomial time computable binary name) is a proper subset of the set of polynomial time computable real numbers, and it is not even closed under addition; see Ko [55].

Now we turn to functions.

Theorem 10.13 (Brent [26], Schröder [87, 88]). *For each of the following functions $f :\subseteq \mathbb{R}^m \to \mathbb{R}$ and domains $K \subseteq \mathbb{R}^m$, there exists a Turing machine M which, computes f in time $\mathcal{O}(t)$ with input lookahead l:*

function $f(x_1, \ldots, x_m)$	*domain* K	*time* $\mathcal{O}(t(n))$	*lookahead* $l(n)$
$-x_1$	$\mathbb{R}$	n	n
$x_1 + x_2$	$[-1,1] \times [-1,1]$	n	$n + c$
$x_1 \cdot x_2$	$[-1,1] \times [-1,1]$	$\mathcal{M}(n)$	$2n + c$
$x_1 \cdot x_2$	$[-1,1] \times [-1,1]$	$\mathcal{M}(n) \cdot \log_2(n)$	$n + c$
$1/x_1$	$[7/8, 2]$	$\mathcal{M}(n)$	$2n + c$
$1/x_1$	$[7/8, 2]$	$\mathcal{M}(n) \cdot \log_2(n)$	$n + c$
$\exp, \sin, \cos$	$[-1,1]$	$\mathcal{M}(n) \cdot \log_2(n)$	$n + c$

A machine computing a real function f may read more digits from the input ρ_{sd}-name p than necessary to define the result with precision 2^{-n}. Let $\mathrm{Dep}(p)(n)$ be the smallest number m such that m digits after the binary point suffice to define n digits after the binary point of a ρ_{sd}-name of $f\rho_{\mathrm{sd}}(p)$. Call a machine *k-input-optimal*, if for any input name p it reads at most the first $\mathrm{Dep}(p)(n) + k$ digits after the point for writing the first n digits after the point of the result. For some functions there is a trade-off between input information and computation time.

Theorem 10.14 (Weihrauch [107]). *There exists a real number function $f : [0,1] \to \mathbb{R}$ with the following properties:*

1. *The function f can be computed in time $\mathcal{O}(n)$ with input lookahead $4n$.*
2. *For any k, the function f can be computed by a k-input-optimal machine in polynomial time if, and only if, $\mathrm{P} = \mathrm{NP}$.*

If the input is itself the result of some (expensive) computation, it is desirable to use as few input digits as possible. However, as we have just seen, this might increase the computation time substantially. In case of a composition of functions, for optimizing the total computation time, one has to balance input lookahead and time complexity of the second function in an appropriate way.

We have seen that if $f : [0,1] \to \mathbb{R}$ is computable and has a continuous second derivative, then f' is computable as well (Corollary 8.7). In fact, the condition that the function has a continous second derivative can be replaced by the weaker condition that the derivative is Lipschitz continuous; see Pour-El and Richards [80].

Theorem 10.15 (Müller [75]). *Let t be a regular function with $\mathcal{M} \in \mathcal{O}(t)$. If $f : [0,1] \to \mathbb{R}$ is a function that is computable in time $\mathcal{O}(t)$ and that has a Lipschitz continuous derivative, then the derivative f' is also computable in time $\mathcal{O}(t)$.*

We conclude this section with a comment on the complexity of real number functions as it is defined in the real random access machine model, most notably by Blum et al. [11]. In this model, one assumes that a computer can process real numbers with infinite precision, and that each comparison and arithmetic operation takes one step. This is certainly not realistic. On the other hand, the idea to consider real numbers as

entities on which one can operate directly without having to refer to names or representations looks attractive. Brattka and Hertling [20] have shown that one can express the computability and polynomial time complexity notion for real number functions as defined in this section also via a real random access machine model. This model is a natural generalization of the random access machine model for computations over the natural numbers with logarithmic cost measure.

11 Computational Complexity of Sets and Operators over the Real Numbers

So far we have discussed the time complexity of real numbers and of real number functions. The next step would be to define the time complexity of subsets of $\mathbb{R}^k$. Before we do that, let us look from a more general perspective at the problem to define time complexity for arbitrary mathematical objects that one might wish to compute in computations over the real numbers. We had observed that the Cauchy representation was unsuitable even for defining the time complexity of real numbers. Why did we come to that conclusion? Well, for defining the time complexity of a real number one would certainly take a name with "minimal time complexity." The problem with the Cauchy representation is that for any real number there are too many Cauchy names and such a minimum does not exist. A sufficient condition on a representation δ of a set X would be that for each $x \in X$, the fiber $\delta^{-1}(x)$ should be compact. Since we wish to have uniform time bounds even for all objects in a compact set, e.g., for all real numbers in a closed, finite interval, it makes sense to demand more: that for any compact set $K \subseteq X$ the set $\delta^{-1}(K)$ is compact as well. Let us call a representation δ of a topological space X *proper* if for each compact $K \subseteq X$ the set $\delta^{-1}(K)$ is compact.

Proposition 11.1. *The signed-digit representation is a proper representation of the real numbers equivalent to the Cauchy representation ρ. In particular, it is admissible.*

Schröder [86, 92] has characterized the spaces that possess admissible representations that have compact fibers or are even proper. Weihrauch [110] has continued this study and has shown how one can introduce in a natural way computational complexity on computable metric spaces. Labhalla, Lombardi, and Moutai [61] suggested other approaches for defining computational complexity on computable metric spaces.

Now let us introduce time complexity of compact subsets of Euclidean spaces. The metric space of compact subsets of $\mathbb{R}$ (we consider the Hausdorff metric) possesses proper admissible representations. A name in such a representation should in addition have the property that prefixes of increasing length should describe the set with increasing precision, e.g., by describing the set with a sequence of "pixels." A problem with such a name is that in order to describe the set with precision 2^{-n} one will

in general need an exponential number of "pixels." Thus, simply writing down such a prefix requires exponential time. This does not look very natural. The problem seems to be that a name of a set describes the set as a whole. Often one may be happy to be able to describe some details of the set with high precision. This idea is realized in the following definition of "local time complexity" of a compact set, as defined by Rettinger and Weihrauch [82].

Definition 11.2. A compact set $K \subseteq [0,1]^2$ is called *(locally) computable in time $\mathcal{O}(t)$, for some total function $t : \mathbb{N} \to \mathbb{N}$* if there exists a function $f : (\{0,1\}^*)^3 \to \{0,1\}^*$ computable in time $\mathcal{O}(t)$ and defined on all triples $(0^n, \nu_{\mathbb{N}}^{-1}(i), \nu_{\mathbb{N}}^{-1}(j))$ with $n \in \mathbb{N}$, $i, j \in \{0, \ldots, 2^n\}$ and with

$$f(0^n, \nu_{\mathbb{N}}^{-1}(i), \nu_{\mathbb{N}}^{-1}(j)) = \begin{cases} 1 & \text{if } d_K(\frac{i}{2^n}, \frac{j}{2^n}) < 2^{-n}, \\ 0 & \text{if } d_K(\frac{i}{2^n}, \frac{j}{2^n}) > 2 \cdot 2^{-n}, \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases}$$

This definition realizes the idea that being able to compute a compact subset of $\mathbb{R}^2$ means being able to plot it with any desired precision, as in Proposition 5.7. Given a screen of print area of fixed resolution, say with k pixels, it requires $\mathcal{O}(k \cdot t(n))$ time to print a set $K \subseteq \mathbb{R}^2$ with (local) time complexity t and a zoom factor 2^n. This definition can be generalized straightforwardly to compact subsets of $\mathbb{R}^m$ for any $m \geq 1$.

Remark 11.3. Chou and Ko [34] (see also [54, 55]) have defined and analyzed several other notions of computational complexity of two-dimensional regions. For example, they call a subset $K \subseteq \mathbb{R}^2$ *polynomial-time approximable*, resp. *polynomial-time recognizable*, if a Turing machine, given a point $x \in \mathbb{R}^2$, decides whether x is in K within polynomial time and makes errors only on a subset of $\mathbb{R}^2$ of measure at most 2^{-n}, resp. makes errors only for points of distance at most 2^{-n} from the boundary of K. If a Turing machine of this kind never errs for points in K, then they call the set K *strongly polynomial-time approximable*, resp. *strongly polynomial-time recognizable*. If a set is locally computable in polynomial time, then it is strongly polynomial-time recognizable, but the converse is true if, and only if, $\mathrm{P} = \mathrm{NP}$ (Braverman [23]).

We had characterized the nonempty, closed, computable subsets of $\mathbb{R}^k$ also as those nonempty closed subsets that have a computable distance function. It looks natural to ask for the time complexity of the distance function of a compact subset of $\mathbb{R}^k$. The following result clarifies the relation between the notion in Definition 11.2 and the time complexity of the distance function.

Theorem 11.4 (Braverman [23]). *Let $K \subseteq \mathbb{R}^n$ be a nonempty compact subset.*

1. *If the distance function $d_K : \mathbb{R}^n \to \mathbb{R}$ of K is polynomial-time computable, then K is polynomial-time computable.*

2. *If* $n = 1$*, then the converse holds.*

3. *If* $n > 1$*, then the converse holds if, and only if,* $\mathrm{P} = \mathrm{NP}$.

Remark 11.5. A similar result for strongly polynomial-time recognizable sets with strongly polynomial-time recognizable complement and for the distance to the boundary of the set has been shown by Chou and Ko [35].

In view of the intuitively appealing meaning of the time complexity of compact sets as in Definition 11.2, we will stick to that definition.

Very interesting compact subsets of Euclidean spaces are fractal sets, in particular the Julia sets. For computability and complexity considerations, we will identify $\mathbb{C}$ with $\mathbb{R}^2$.

Definition 11.6. Let $f : \mathbb{C} \to \mathbb{C}$ be a polynomial function of degree ≥ 2.

- A point $z \in \mathbb{C}$ is called a *periodic point* of f, if there exists a $p \in \mathbb{N}$ such that $f^p(z) = z$.
- A periodic point $z \in \mathbb{C}$ of f is called *repelling*, if $|(f^p)'(z)| > 1$.
- The Julia set $J(f)$ of f is the closure of the set of repelling periodic points of f.
- A point $z \in \mathbb{C}$ is called *critical*, if $f'(z) = 0$.
- The function f is called *hyperbolic*, if $J(f)$ is disjoint from the closure of the orbits $\bigcup_{z \text{ critical}} \bigcup_{n=0}^{\infty} f^n(z)$.

In 1998 Zhong [118] showed that a Turing machine, given the coefficients of a hyperbolic polynomial f, computes the Julia set $J(f)$ of f; i.e., given in addition a point $z \in \mathbb{C}$ and a number $k \in \mathbb{N}$, it computes the distance of z from $J(f)$ with precision 2^{-k}, or it computes a function for $K := J(f)$ as described in Definition 11.2. Rettinger and Weihrauch [82] looked at the time complexity of Julia sets and showed the following first result.

Theorem 11.7 (Rettinger and Weihrauch [82]). *There is a Turing machine which, given a signed-digit name of a* $c \in \mathbb{C}$ *with* $|c| < \frac{1}{4}$*, computes in time* $\mathcal{O}(n^2 \cdot \mathcal{M}(n)) \subseteq \mathcal{O}(n^3 \cdot \log n \cdot \log\log n)$ *a function* f *as in Definition 11.2 for the Julia set of the quadratic polynomial* $z \mapsto z^2 + c$.

This result was generalized and strengthened in 2004 independently by Braverman [24] and Rettinger [81] to hyperbolic polynomial and hyperbolic rational functions, respectively, with the time bound $\mathcal{O}(n \cdot \mathcal{M}(n))$. Since then, Binder, Braverman, and Yampolsky have obtained a series of positive and negative results concerning computability and polynomial time computability of Julia sets; see [7] and the references therein.

Now we come to the time complexity of numerical operators, e.g., of type

$$F : \mathcal{C}[0,1] \to \mathcal{C}[0,1]$$

such as integration or differentiation. It is difficult to define a uniform notion of complexity for such operators because $\mathcal{C}[0,1]$ is not locally compact, and therefore, there is no obvious way to define a uniform notion of complexity (not even parameterized over $\mathbb{N}$ as we had suggested to do it for functions on $\mathbb{R}^k$; see Remark 10.7). But one can study the complexity of such operators restricted to compact subspaces $K \subseteq \mathcal{C}[0,1]$. In fact, this is often done in numerical analysis, although often only in an algebraic computation model; see Traub et al. [99]. It is likely that for many numerically stable problems over function spaces, the time complexity in the Turing machine model will not be very different from the "arithmetic" complexity in an algebraic computation model; see Woźniakowski [113]. But a thorough study of the time complexity in the sense of Weihrauch [110] or Labhalla et al. [61] of higher type numerical operators in the Turing machine model seems to be missing.

There is a different approach, due to Ko and Friedman, see e.g. [56], and covered by Ko's book [55]: if F is a numerical operator receiving real number functions as input, one can study the time complexity of $F(f)$ for polynomial time computable input f. This approach leads to the insight that the complexity of many numerical problems can be characterized by discrete complexity classes or, the other way around, to the insight that many famous open problems concerning discrete complexity classes can be found to be hidden in numerical problems. We formulate only three results and refer the reader to Ko [55] for many more results and many open problems in this context.

Theorem 11.8 (Friedman [39]). *The following two statements are equivalent:*

1. $\mathrm{P} = \mathrm{NP}$.
2. *For each polynomial-time computable* $f : [0,1] \to \mathbb{R}$*, the maximum function* $g : [0,1] \to \mathbb{R}$*, defined by*

$$g(x) := \max\{f(y) : 0 \leq y \leq x\}$$

for all $x \in [0,1]$*, is polynomial-time computable.*

Proof (Sketch of Proof). The direction "1.$\Longrightarrow$2." of the proof is based on the idea that

$$z \leq g(x) \iff (\exists y \in [0,x])\ z \leq f(y).$$

Using the Polynomial-Time Projection Theorem, this is (approximately) decidable in polynomial time if $\mathrm{P} = \mathrm{NP}$. By a binary search over z, one can determine $g(x)$ in polynomial time. □

Theorem 11.9 (Friedman [39]). *The following two statements are equivalent:*

1. $\mathrm{FP} = \#\mathrm{P}$.
2. *For each polynomial-time computable* $f : [0,1] \to \mathbb{R}$*, the integral function* $g : [0,1] \to \mathbb{R}$*, defined by*

$$g(x) := \int_0^x f(t)\,\mathrm{d}t$$

for all $x \in [0,1]$*, is polynomial-time computable.*

Here $\#\mathrm{P}$ denotes the class of functions that count the number of accepting computations of a nondeterministic polynomial-time Turing machine.

Proof (Sketch of Proof). For the proof of "1.$\Longrightarrow$2." one can guess a number of points (t, y) with $0 \le t \le x$ and then count those with $y \le f(t)$ to get an approximation for the integral $g(x)$ (in case f is positive). □

Theorem 11.10 (Ko [53]). *For the following three statements, the implications* $1 \Rightarrow 2$ *and* $2 \Rightarrow 3$ *hold true.*

1. $\mathrm{P} = \mathrm{PSPACE}$.
2. *For each polynomial-time computable* $f : [0,1] \times [-1,1] \to \mathbb{R}$ *that satisfies the Lipschitz condition*

$$|f(x, z_1) - f(x, z_2)| \le L \cdot |z_1 - z_2|$$

for some $L > 0$*, the unique solution* $y : [0,1] \to \mathbb{R}$ *of the differential equation*

$$y'(x) = f(x, y(x)), \qquad y(0) = 0$$

is polynomial-time computable.

3. $\mathrm{FP} = \#\mathrm{P}$.

Acknowledgments

The first author has been supported by the National Research Foundation of South Africa. The second author has been supported by the German Research Council (DFG).

References

1. O. Aberth. *Computable Analysis*. McGraw-Hill, New York, 1980.

2. K. Ambos-Spies, K. Weihrauch, and X. Zheng. Weakly computable real numbers. *Journal of Complexity*, 16(4):676–690, 2000.
3. A. Avizienis. Signed-digit number representations for fast parallel arithmetic. *IRE Transactions on Electronic Computers*, 10:389–400, 1961.
4. G. Baigger. Die Nichtkonstruktivität des Brouwerschen Fixpunktsatzes. *Arch. Math. Logik Grundlag.*, 25:183–188, 1985.
5. G. Barmpalias. The approximation structure of a computably approximable real. *The Journal of Symbolic Logic*, 68(3):885–922, 2003.
6. G. Beer. *Topologies on Closed and Closed Convex Sets*, volume 268 of *Mathematics and Its Applications*. Kluwer Academic, Dordrecht, 1993.
7. I. Binder, M. Braverman, and M. Yampolsky. On computational complexity of Siegel Julia sets. *Comm. Math. Phys.*, 264(2):317–334, 2006.
8. E. Bishop and D. S. Bridges. *Constructive Analysis*, volume 279 of *Grundlehren der Mathematischen Wissenschaften*. Springer, Berlin, 1985.
9. J. Blanck. Domain representations of topological spaces. *Theoretical Computer Science*, 247:229–255, 2000.
10. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer, New York, 1998.
11. L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.
12. J. M. Borwein and P. B. Borwein. *Pi and the AGM*. John Wiley & Sons, New York, 1987.
13. B. Branner. The Mandelbrot set. In R. L. Devaney and L. Keen, editors, *Chaos and Fractals. The Mathematics Behind the Computer Graphics*, volume 39 of *Proceedings of Symposia in Applied Mathematics*, pages 75–105, Providence, Rhode Island, 1989. American Mathematical Society, 1989.
14. V. Brattka. Computable invariance. *Theoretical Computer Science*, 210:3–20, 1999.
15. V. Brattka. Computable versions of Baire's category theorem. In J. Sgall, A. Pultr, and P. Kolman, editors, *Mathematical Foundations of Computer Science 2001*, volume 2136 of *Lecture Notes in Computer Science*, pages 224–235, Berlin, 2001. Springer. 26th International Symposium, MFCS 2001, Mariánské Lázně, Czech Republic, August 27–31, 2001.
16. V. Brattka. The inversion problem for computable linear operators. In H. Alt and M. Habib, editors, *STACS 2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 391–402, Berlin, 2003. Springer. 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27–March 1, 2003.
17. V. Brattka. Effective Borel measurability and reducibility of functions. *Mathematical Logic Quarterly*, 51(1):19–44, 2005.
18. V. Brattka. On the Borel complexity of Hahn-Banach extensions. In V. Brattka, L. Staiger, and K. Weihrauch, editors, *Proceedings of the 6th Workshop on Computability and Complexity in Analysis*, volume 120 of *Electronic Notes in Theoretical Computer Science*, pages 3–16, Amsterdam, 2005. Elsevier. 6th International Workshop, CCA 2004, Wittenberg, Germany, August 16–20, 2004.
19. V. Brattka and R. Dillhage. On computable compact operators on Banach spaces. In D. Cenzer, R. Dillhage, T. Grubba, and K. Weihrauch, editors, *Proceedings of the Third International Conference on Computability and Complexity in Analysis*, volume 167 of *Electronic Notes in Theoretical Computer Science*, Amsterdam, 2007. Elsevier. CCA 2006, Gainesville, Florida, November 1–5, 2006.

20. V. Brattka and P. Hertling. Feasible real random access machines. *Journal of Complexity*, 14(4):490–526, 1998.
21. V. Brattka and G. Presser. Computability on subsets of metric spaces. *Theoretical Computer Science*, 305:43–76, 2003.
22. V. Brattka and A. Yoshikawa. Towards computability of elliptic boundary value problems in variational formulation. *Journal of Complexity*, 22(6):858–880, 2006.
23. M. Braverman. *Computational Complexity of Euclidean Sets: Hyperbolic Julia Sets are Poly-Time Computable*. Master thesis, Department of Computer Science, University of Toronto, 2004.
24. M. Braverman. Hyperbolic Julia sets are poly-time computable. In V. Brattka, L. Staiger, and K. Weihrauch, editors, *Proceedings of the 6th Workshop on Computability and Complexity in Analysis*, volume 120 of *Electronic Notes in Theoretical Computer Science*, pages 17–30, Amsterdam, 2005. Elsevier. 6th International Workshop, CCA 2004, Wittenberg, Germany, August 16–20, 2004.
25. M. Braverman and S. Cook. Computing over the reals: Foundations for scientific computing. *Notices of the AMS*, 53(3):318–329, 2006.
26. R. Brent. Fast multiple-precision evaluation of elementary functions. *Journal of the Association for Computing Machinery*, 23(2):242–251, 1976.
27. D. Bridges and F. Richman. *Varieties of Constructive Mathematics*, volume 97 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1987.
28. L. Brouwer. *Collected Works, Vol. 1, Philosophy and Foundations of Mathematics*. North-Holland, Amsterdam, 1975. Heyting, A. (ed).
29. L. Brouwer. *Collected Works, Vol. 2, Geometry, Analysis, Topology and Mechanics*. North-Holland, Amsterdam, 1976. Freudenthal, H. (ed).
30. C. S. Calude, P. H. Hertling, B. Khoussainov, and Y. Wang. Recursively enumerable reals and Chaitin ω numbers. *Theoretical Computer Science*, 255:125–149, 2001.
31. G. Ceĭtin. Algorithmic operators in constructive metric spaces. *Tr. Mat. Inst. Steklov*, 67:295–361, 1962. (in Russian, English trans. in AMS Trans. 64, 1967).
32. D. Cenzer and J. B. Remmel. Index sets for computable differential equations. *Mathematical Logic Quarterly*, 50(4,5):329–344, 2004.
33. G. J. Chaitin. A theory of program size formally identical to information theory. *Journal of the Association for Computing Machinery*, 22:329–340, 1975.
34. A. Chou and K.-I. Ko. Computational complexity of two-dimensional regions. *SIAM Journal on Computing*, 24:923–947, 1995.
35. A. W. Chou and K.-I. Ko. The computational complexity of distance functions of two-dimensional domains. *Theoretical Computer Science*, 337:360–369, 2005.
36. R. G. Downey, D. R. Hirschfeldt, A. Nies, and F. Stephan. Trivial reals. In R. Downey, D. Decheng, T. S. Ping, Q. Y. Hui, and M. Yasugi, editors, *Proceedings of the 7th and 8th Asian Logic Conferences*, pages 63–102, Singapore, 2003. World Scientific. 7th Conference: Hsi-Tou, Taiwan, June 6–10, 1999; 8th Conference: Chongqing, China, August 29–September 2, 2002.
37. A. Edalat. Domains for computation in mathematics, physics and exact real arithmetic. *Bulletin of Symbolic Logic*, 3(4):401–452, 1997.
38. M. H. Escardó. PCF extended with real numbers. In K.-I. Ko and K. Weihrauch, editors, *Computability and Complexity in Analysis*, volume 190 of *Informatik Berichte*, pages 11–24. FernUniversität Hagen, Sept. 1995. CCA Workshop, Hagen, August 19–20, 1995.
39. H. Friedman. On the computational complexity of maximization and integration. *Advances in Mathematics*, 53:80–98, 1984.

40. G. Gherardi. Effective Borel degrees of some topological functions. *Mathematical Logic Quarterly*, 52(6):625–642, 2006.
41. A. Grzegorczyk. On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.
42. J. Hauck. Berechenbare reelle Funktionen. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 19:121–140, 1973.
43. J. Hauck. Konstruktive Darstellungen reeller Zahlen und Folgen. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 24:365–374, 1978.
44. P. Hertling. A topological complexity hierarchy of functions with finite range. Technical Report 223, Centre de recerca matematica, Institut d'estudis catalans, Barcelona, Barcelona, Oct. 1993. Workshop on Continuous Algorithms and Complexity, Barcelona, October, 1993.
45. P. Hertling. Topological complexity with continuous operations. *Journal of Complexity*, 12:315–338, 1996.
46. P. Hertling. *Unstetigkeitsgrade von Funktionen in der effektiven Analysis*. PhD thesis, Fachbereich Informatik, FernUniversität Hagen, 1996.
47. P. Hertling. An effective Riemann Mapping Theorem. *Theoretical Computer Science*, 219:225–265, 1999.
48. P. Hertling. A real number structure that is effectively categorical. *Mathematical Logic Quarterly*, 45(2):147–182, 1999.
49. P. Hertling. A Banach-Mazur computable but not Markov computable function on the computable real numbers. *Annals of Pure and Applied Logic*, 132(2-3):227–246, 2005.
50. P. Hertling. Is the Mandelbrot set computable? *Mathematical Logic Quarterly*, 51(1):5–18, 2005.
51. M. D. Hirsch. Applications of topology to lower bound estimates in computer science. In *From Topology to Computation: Proceedings of the Smalefest, Berkeley, CA, 1990*, pages 395–418, New York, 1993. Springer.
52. C.-K. Ho. Relatively recursive reals and real functions. *Theoretical Computer Science*, 210(1):99–120, 1999.
53. K.-I. Ko. On the computational complexity of ordinary differential equations. *Inform. Contr.*, 58:157–194, 1983.
54. K.-I. Ko. Approximation to measurable functions and its relation to probabilistic computation. *Annals of Pure and Applied Logic*, 30:173–200, 1986.
55. K.-I. Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.
56. K.-I. Ko and H. Friedman. Computational complexity of real functions. *Theoretical Computer Science*, 20:323–352, 1982.
57. G. Kreisel, D. Lacombe, and J. Shoenfield. Partial recursive functionals and effective operations. In A. Heyting, editor, *Constructivity in Mathematics*, Studies in Logic and the Foundations of Mathematics, pages 290–297, Amsterdam, 1959. North-Holland. Proc. Colloq., Amsterdam, Aug. 26–31, 1957.
58. C. Kreitz and K. Weihrauch. Theory of representations. *Theoretical Computer Science*, 38:35–53, 1985.
59. B. A. Kušner. *Lectures on Constructive Mathematical Analysis*, volume 60 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, Rhode Island, 1984.
60. A. Kučera and T. A. Slaman. Randomness and recursive enumerability. *SIAM J. Comput.*, 31(1):199–211, 2001.

61. S. Labhalla, H. Lombardi, and E. Moutai. Espaces métriques rationnellement présentés et complexité, le cas de l'espace des fonctions réelles uniformément continues sur un intervalle compact. *Theoretical Computer Science*, 250:265–332, 2001.
62. D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. *Comptes Rendus Académie des Sciences Paris*, 241:151–153, 1955. Théorie des fonctions.
63. S. Mazur. *Computable Analysis*, volume 33. Razprawy Matematyczne, Warsaw, 1963.
64. G. Metakides and A. Nerode. The introduction of non-recursive methods into mathematics. In A. Troelstra and D. v. Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*, volume 110 of *Studies in Logic and the foundations of mathematics*, pages 319–335, Amsterdam, 1982. North-Holland. Proceedings of the conference held in Noordwijkerhout, June 8–13, 1981.
65. G. Metakides, A. Nerode, and R. Shore. Recursive limits on the Hahn-Banach theorem. In M. Rosenblatt, editor, *Errett Bishop: Reflections on Him and His Research*, volume 39 of *Contemporary Mathematics*, pages 85–91, Providence, Rhode Island, 1985. American Mathematical Society. Proceedings of the memorial meeting for Errett Bishop, University of California, San Diego, September 24, 1983.
66. J. Miller and A. Nies. Randomness and computability: Open questions. *Bull. Symb. Logic*, 12(3):390–410, 2006.
67. J. S. Miller. *Pi-0-1 Classes in Computable Analysis and Topology*. PhD thesis, Cornell University, Ithaca, New York, 2002.
68. J. S. Miller. Degrees of unsolvability of continuous functions. *The Journal of Symbolic Logic*, 69(2):555–584, 2004.
69. R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, New Jersey, 1966.
70. Y. N. Moschovakis. Recursive metric spaces. *Fundamenta Mathematicae*, 55:215–238, 1964.
71. Y. N. Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1980.
72. A. Mostowski. On computable sequences. *Fundamenta Mathematicae*, 44:37–51, 1957.
73. J. M. Muller. *Elementary Functions*. Birkhäuser, Boston, 2nd edition, 2006.
74. N. T. Müller. Subpolynomial complexity classes of real functions and real numbers. In L. Kott, editor, *Proceedings of the 13th International Colloquium on Automata, Languages, and Programming*, volume 226 of *Lecture Notes in Computer Science*, pages 284–293, Berlin, 1986. Springer.
75. N. T. Müller. Polynomial time computation of Taylor series. In *Proceedings of the 22th JAIIO - Panel'93, Part 2*, pages 259–281, 1993. Buenos Aires, 1993.
76. J. Myhill. A recursive function defined on a compact interval and having a continuous derivative that is not recursive. *Michigan Math. J.*, 18:97–98, 1971.
77. C. A. Neff and J. H. Reif. An efficient algorithm for the complex roots problem. *Journal of Complexity*, 12:81–115, 1996.
78. V. Orevkov. A constructive mappping of the square onto itself displacing every constructive point (Russian). *Doklady Akademii Nauk*, 152:55–58, 1963. Translated in: Soviet Math. - Dokl., 4 (1963) 1253–1256.
79. M. B. Pour-El and J. I. Richards. The wave equation with computable inital data such that its unique solution is not computable. *Advances in Math.*, 39:215–239, 1981.
80. M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, Berlin, 1989.
81. R. Rettinger. A fast algorithm for Julia sets of hyperbolic rational functions. In V. Brattka, L. Staiger, and K. Weihrauch, editors, *Proceedings of the 6th Workshop on Computability and Complexity in Analysis*, volume 120 of *Electronic Notes in Theoretical*

Computer Science, pages 145–157, Amsterdam, 2005. Elsevier. 6th International Workshop, CCA 2004, Wittenberg, Germany, August 16–20, 2004.
82. R. Rettinger and K. Weihrauch. The computational complexity of some Julia sets. In M. X. Goemans, editor, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 177–185, New York, 2003. ACM Press. San Diego, California, June 9–11, 2003.
83. H. Rice. Recursive real numbers. *Proc. Amer. Math. Soc.*, 5:784–791, 1954.
84. H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.
85. N. Šanin. *Constructive Real Numbers and Constructive Function Spaces*, volume 21 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, 1968.
86. M. Schröder. Topological spaces allowing type 2 complexity theory. In K.-I. Ko and K. Weihrauch, editors, *Computability and Complexity in Analysis*, volume 190 of *Informatik Berichte*, pages 41–53. FernUniversität Hagen, 1995. CCA Workshop, Hagen, August 19–20, 1995.
87. M. Schröder. Fast online multiplication of real numbers. In R. Reischuk and M. Morvan, editors, *STACS 97*, volume 1200 of *Lecture Notes in Computer Science*, pages 81–92, Berlin, 1997. Springer. 14th Annual Symposium on Theoretical Aspects of Computer Science, Lübeck, Germany, February 27–March 1, 1997.
88. M. Schröder. Online computations of differentiable functions. *Theoretical Computer Science*, 219:331–345, 1999.
89. M. Schröder. Admissible representations of limit spaces. In J. Blanck, V. Brattka, and P. Hertling, editors, *Computability and Complexity in Analysis*, volume 2064 of *Lecture Notes in Computer Science*, pages 273–295, Berlin, 2001. Springer. 4th International Workshop, CCA 2000, Swansea, UK, September 2000.
90. M. Schröder. Effectivity in spaces with admissible multirepresentations. *Mathematical Logic Quarterly*, 48(Suppl. 1):78–90, 2002.
91. M. Schröder. Extended admissibility. *Theoretical Computer Science*, 284(2):519–538, 2002.
92. M. Schröder. Spaces allowing type-2 complexity theory revisited. *Mathematical Logic Quarterly*, 50(4,5):443–459, 2004.
93. A. Slisenko. Examples of a nondiscontinuous but not continuous constructive operator in a metric space. *Trudy Mat. Inst. Steklov*, 72:524–532, 1964. (in Russian, English trans. in AMS Trans. 100, 1972).
94. S. Smale. On the topology of algorithms, I. *Journal of Complexity*, 3:81–89, 1987.
95. R. Soare. Cohesive sets and recursively enumerable Dedekind cuts. *Pacific J. Math.*, 31:215–231, 1969.
96. E. Specker. Nicht konstruktiv beweisbare Sätze der Analysis. *The Journal of Symbolic Logic*, 14(3):145–158, 1949.
97. E. Specker. The fundamental theorem of algebra in recursive analysis. In B. Dejon and P. Henrici, editors, *Constructive Aspects of the Fundamental Theorem of Algebra*, pages 321–329, London, 1969. Wiley-Interscience.
98. V. Stoltenberg-Hansen, I. Lindström, and E. Griffor. *Mathematical Theory of Domains*, volume 22 of *Cambrige Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1994.
99. J. F. Traub, G. Wasilkowski, and H. Woźniakowski. *Information-Based Complexity*. Computer Science and Scientific Computing. Academic Press, New York, 1988.

100. A. Troelstra and D. v. Dalen. *Constructivism in Mathematics, Volume 1*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1988.
101. A. Troelstra and D. v. Dalen. *Constructivism in Mathematics, Volume 2*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1988.
102. A. M. Turing. On computable numbers, with an application to the "Entscheidungsproblem". *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.
103. A. M. Turing. On computable numbers, with an application to the "Entscheidungsproblem". A correction. *Proceedings of the London Mathematical Society*, 43(2):544–546, 1937.
104. V. Vassiliev. Cohomology of braid groups and the complexity of algorithms. *Funktsional. Anal. i Prilozhen.*, 22(3):15 – 24, 1989. Englische Übers. in *Functional. Anal. Appl.*, 22:182–190, 1989.
105. M. Washihara. Computability and Fréchet spaces. *Mathematica Japonica*, 42(1):1–13, 1995.
106. K. Weihrauch. *Computability*, volume 9 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1987.
107. K. Weihrauch. On the complexity of online computations of real functions. *Journal of Complexity*, 7:380–394, 1991.
108. K. Weihrauch. The TTE-interpretation of three hierarchies of omniscience principles. Informatik Berichte 130, FernUniversität Hagen, Hagen, Sept. 1992.
109. K. Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.
110. K. Weihrauch. Computational complexity on computable metric spaces. *Mathematical Logic Quarterly*, 49(1):3–21, 2003.
111. K. Weihrauch and N. Zhong. Is wave propagation computable or can wave computers beat the Turing machine? *Proceedings of the London Mathematical Society*, 85(2):312–332, 2002.
112. H. Weyl. Randbemerkungen zu Hauptproblemen der Mathematik. *Math. Zeitschrift*, 20:131–150, 1924.
113. H. Woźniakowski. Why does information-based complexity use the real number model? *Theoretical Computer Science*, 219:451–465, 1999.
114. M. Yasugi, T. Mori, and Y. Tsujii. Effective properties of sets and functions in metric spaces with computability structure. *Theoretical Computer Science*, 219:467–486, 1999.
115. X. Zheng. Recursive approximability of real numbers. *Mathematical Logic Quarterly*, 48(Suppl. 1):131–156, 2002.
116. X. Zheng and R. Rettinger. Weak computability and representation of reals. *Mathematical Logic Quarterly*, 50(4,5):431–442, 2004.
117. X. Zheng and K. Weihrauch. The arithmetical hierarchy of real numbers. *Mathematical Logic Quarterly*, 47(1):51–65, 2001.
118. N. Zhong. Recursively enumerable subsets of R^q in two computing models: Blum-Shub-Smale machine and Turing machine. *Theoretical Computer Science*, 197:79–94, 1998.
119. N. Zhong and K. Weihrauch. Computability theory of generalized functions. *Journal of the Association for Computing Machinery*, 50(4):469–505, 2003.
120. M. Ziegler and V. Brattka. Computability in linear algebra. *Theoretical Computer Science*, 326(1–3):187–211, 2004.