A Simple Introduction to Computable Analysis

Klaus Weihrauch
FernUniversität
D - 58084 Hagen

(corrected 2nd version, July 1995)

Contents

1 Introduction

During the last 60 years an extensive theory of computability and computational complexity has been developed (see e.g. Rogers [Rog 67], Odifreddi [Odi 89], Weihrauch [Wei 87], Hopcroft and Ullmann [HU 79], Wagner and Wechsung [WW 86]). Without doubt this "Type 1 theory" models the behaviour of real world computers for computations on discrete sets like natural numbers, finite words, finite graphs etc. quite adequately.

A large part of computers, however, is used for solving numerical problems. Therefore convincing theoretical foundations are indispensible also for computable analysis. Several theories for studying aspects of effectivity in analysis have been developed in the past (see chapter 10). Although each of these approaches has its merits, none of them has been accepted by the majority of mathematicians or computer scientists. Compared with Type 1 computability, foundations of computable analysis have been neglected in research and almost disregarded in teaching.

This paper is an introduction to "Type 2 Theory of Effectivity" (TTE). TTE is one among the existing theories of effective analysis. It extends ordinary Type 1 computability theory and connects it with abstract analysis. Its origin is a definition of computable real functions given by Grzegorczyk in 1955 [Grz 55], which is based on the definition of computable operators on the set ω^ω of sequences of natural numbers. Real numbers are encoded by fast (with "speed 1/n") converging Cauchy sequences of rational numbers, and these are encoded by sequences of natural numbers. A real function is computable in Grzegorczyk's sense, iff it can be represented by a computable operator on such encodings of real numbers. In the following years this kind of computability has been investigated by several authors (e.g. Grzegorczyk [Grz 57], Klaua [Kla 61], Hauck [Hau 73, Hau 76] and Wiedmer [Wie 80]). The computational complexity theory for real functions developed by Ko and Friedmann [KF 82, Ko 91] can be considered as a special branch.

The study of representations, i.e. functions from ω^ω onto sets, as objects of separate interest results in an essential generalization of Grzegorczyk's original definition and admits to find and justify natural computability definitions for functions on most of the sets used in ordinary analysis. Basic concepts are explained in Weihrauch and Kreitz [WK 84, Wei 85, KW 85, Wei 87]. The theory has been expanded in several papers by Hertling, Kreitz, Müller and Weihrauch ranging from topological considerations to investigation of concrete computational complexity [WK 86, KW 86, Mue 86, Mue 87, WK 91, Wei 91, Wei 93, Wei 92A, Wei 92B, HW 94]. As an interesting feature, continuity can be interpreted in this context as a very fundamental kind of effectivity or constructivity, and simple topological considerations explain a number of well known observations from effective analysis very satisfactorily.

This paper is not a complete presentation of TTE but only a technically and conceptually simplified selection from [Wei 94]. The main stress is put on basic concepts and on simple but typical applications, while the theoretical background is reduced to the bare essentials.

We assume that the reader has some basic knowledge in computability theory (Turing machines, computable functions, recursive sets, recursively enumerable sets).

There are several good introductions, e.g. the classical book by Hopcroft and Ullman [HU 79] or Bridges [Bri 94]. In addition to standard Calculus we use some simple concepts from topology (topological space, open and closed sets, continuous functions, metric space, Cauchy sequence, compact set). Any introduction to topology (e.g. [Eng 89]) may be used as a reference.

In the following we axplain some notations which will be used in this paper. By $f :\subseteq X \longrightarrow Y$ we denote a *partial function* from X to Y, i.e. a function from a subset of X, called the domain of f ($dom(f)$), to Y. The function $f :\subseteq X \longrightarrow Y$ is *total*, iff $dom(f) = X$; in this case we write $f : X \longrightarrow Y$ as usual. A finite alphabet is a non–empty finite set. In Section 2, Σ denotes any finite alphabet with $\{0, 1\} \subseteq \Sigma$. In the following section Σ is some fixed sufficiently large finite alphabet containing all the symbols we shall need. Let $\omega := \{0, 1, 2, \ldots\}$ be the set of natural numbers. As usual, Σ^* is the set of all finite words $a_1 \ldots a_k$ with $k \in \omega$ and $a_1, \ldots, a_k \in \Sigma$. The empty word is denoted by ε. Let $\Sigma^\omega := \{a_0a_1 \ldots \mid a_i \in \Sigma\} = \{p \mid p : \omega \longrightarrow \Sigma\}$ be the set of infinite sequences (or ω–sequences) with elements from Σ. We use suggestive informal notations for defining finite and infinite sequences over Σ. If $u = a_1 \ldots a_k$, $v = b_1 \ldots b_l$ and $p = c_0c_1 \ldots \in \Sigma^\omega$ ($a_i, b_i, c_i \in \Sigma$), then $uv := a_1 \ldots a_kb_1 \ldots b_l$, $up := a_1 \ldots a_kc_0c_1 \ldots \in \Sigma^\omega$, $u^m := a_1 \ldots a_ka_1 \ldots a_k \ldots a_1 \ldots a_k$ (m times), $u^\omega := uuu \ldots := a_1 \ldots a_ka_1 \ldots a_k \ldots \in \Sigma^\omega$. If $x = uvw \in \Sigma^*$ and $q = uvp \in \Sigma^\omega$ then u is a *prefix* and v is a *subword* of x and q. We extend the above notations to sets of finite or infinite sequences. For example, $01\Sigma^* = \{x \in \Sigma^* \mid 01$ is a prefix of $x\}$ and $\Sigma^*u\Sigma^\omega = \{p \in \Sigma^\omega \mid u$ is a subword of $p\}$.

In Chapter 2 we generalize computability from finite to infinite sequences of symbols and illustrate the definition by a number of examples. We introduce the Cantor topology and show that computable functions are continuous. We introduce notations and representations and define, how topological and computational concepts are transferred from sequences to named sets. In Chapter 3 we introduce standard representations of the real numbers (the *interval representation* and the *Cauchy representation*) and investigate the computability concepts induced by them on the real numbers. We give examples for computable and non–computable real numbers, we characterize the recursively enumerable subsets of $\mathbb{R}$ and prove computability of a number of real functions. In Chapter 4 we give reasons for selecting the interval and the Cauchy representation and for rejecting, e.g., the decimal representation. We prove that every computable real function is continuous, we formulate the thesis that every physically computable function is continuous and we prove that no injective and no surjective representation can be equivalent to the Cauchy representation. In Chapter 5 we introduce representations of the open and of the compact subsets of the real numbers. We prove effective versions of some well known classical properties, especially we prove a computational version of the Heine/Borel theorem on compact sets. We introduce representations of the classes $C(\mathbb{R})$ and $C[0; 1]$ of continuous real functions and discuss their effectivity in Chapter 6. We present some computational versions of well known properties and consider the determination of a modulus of continuity, of the maximum value, the derivative and the integral. Determination of zeros of continuous functions is considered in Chapter 7. We prove that the general problem can not even be solved continuously. Under certain restrictions we have a computable but non–extensional solution operator. A computable operator exists only on the set of continuous functions which have exactly one zero. In Chapter 8 we introduce as new concepts computation time and input lookahead of Type 2 machines with infinite output. In Chapter 9 we define the modified binary representation which is appropriate for introducing computational complexity of real functions. We determine bounds of time and input lookahead for addition, multiplication and, by an application of Newton's method, for inversion. Finally we define the complexity of compact sets, which can be interpreted as "plotter complexity". Some other approaches to effective analysis are discussed in Chapter 10.

2 Computability on Finite and Infinite Words, Naming Systems

In this Chapter, Σ is any finite alphabet, i.e. any finite non empty set. Turing machines are a convenient mathematical model for defining computability of wordfunctions $f :\subseteq (\Sigma^*)^k \longrightarrow \Sigma^*$. By the Church/Turing thesis, a word function is computable informally or by a physical device, if and only if it can be computed by a Turing machine. Moreover, Turing machines model time and storage complexity of physical computers rather realistically. (A standard reference is the book by Hopcroft and Ullman [HU 79]).

In this section we introduce our basic computational model for computable analysis, the *Type 2 machines*. We formulate a generalization of the Church/Turing thesis, we prove that computable functions on finite or infinite sequences are continuous and define recursively enumerable sets. We introduce notations and representations and define, how effectivity of elements, sets, functions and relations is transferred by naming systems. Many examples illustrate the definitions.

Roughly speaking, a Type 2 machine is a Turing machine for which not only finite but also infinite sequences of symbols may be considered as inputs or outputs. We give an informal definition of Type 2 machines and their semantics.

Definition 2.1 (*Type 2 machines*)

A Type 2 machine M is defined by two components:

(i) a Turing machine with k one–way input tapes ($k \geq 0$), a single one–way output tape and finitely many work tapes,

(ii) a type specification $(Y_1, \ldots, Y_k, Y_0)$ with $\{Y_0, \ldots, Y_k\} \subseteq \{\Sigma^*, \Sigma^\omega\}$.

The type specification expresses that $f_M :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ is the type of the function computed by the machine M. It tells which of the input and output tapes are provided for finite and which for infinite sequences. Notice, that input and output tapes are restricted to one–way (left to right).

Definition 2.2 (*semantics of Type 2 machines*)

The function $f_M :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ computed by the Type 2 machine M (the semantics of M) is defined as follows:

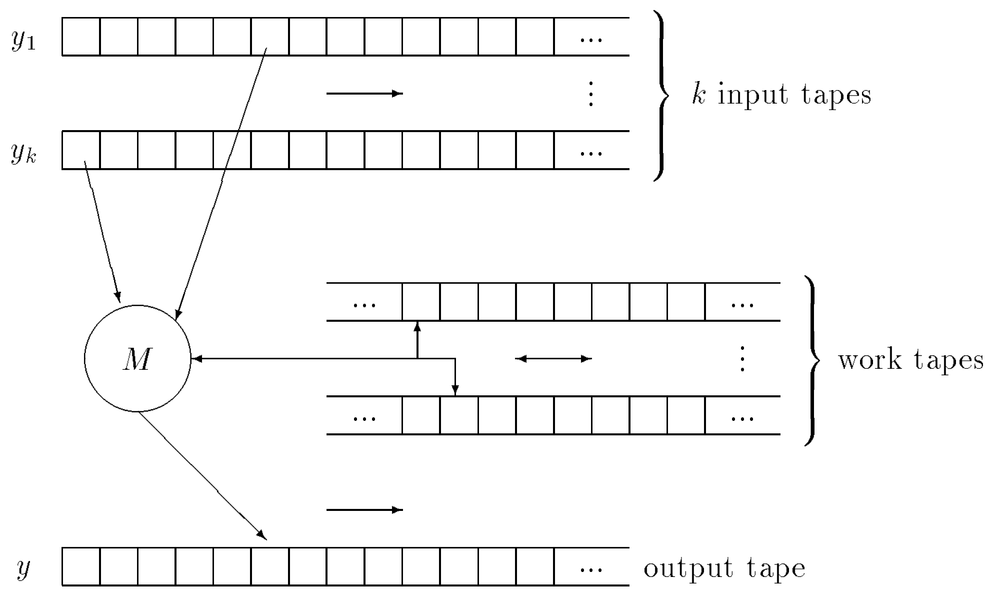
Case $Y_0 = \Sigma^*$ (finite output):

$f_M(y_1, \ldots, y_k) = w \in \Sigma^*$, iff M with input $(y_1, \ldots, y_k)$ halts with result w on the output tape.

Case $Y_0 = \Sigma^\omega$ (infinite output):

> $f_M(y_1, \ldots, y_k) = p \in \Sigma^\omega$, iff M with input $(y_1, \ldots, y_k)$ computes forever writing the sequence p on the output tape.

Notice, that in the case $Y_0 = \Sigma^*$ the result $f_M(y_1, \ldots, y_k)$ is undefined, if the machine writes only finitely many symbols on the output tape but does not halt. A Type 2 machine can be visualized by its underlying Turing machine.



$f_M(y_1, ..., y_k) = y$

Readers not familiar with Turing machines find a more detailed definition in Appendix A.

We use the Type 2 machines for defining *computability* of functions $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ ($\{Y_0, \ldots, Y_k\} \subseteq \{\Sigma^*, \Sigma^\omega\}$). The following definition generalizes the common definition of computable wordfunctions, since in the special case $Y_0 = \ldots = Y_k = \Sigma^*$ Type 2 machines are ordinary Turing machines.

Definition 2.3 (*Type 2 computability*)

> Let Σ be a finite alphabet. Assume $\{Y_0, \ldots, Y_k\} \subseteq \{\Sigma^*, \Sigma^\omega\}$ ($k \geq 0$). A function $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ is *computable*, iff $f = f_M$ for some Type 2 machine M. A sequence y is a computable element of Y_0, iff the 0–place function $f : \{()\} \longrightarrow Y_0$ with $f() = y$ is computable.

Since Turing machines and their halting computations are finite, they have physical realizations (of course, only if size and time do not exceed certain bounds). By definition, Type 2 machines may require infinite input and output tapes and may perform infinite computations which cannot be realized *actually*, since infinite tapes do not exist and infinite computations cannot be completed in reality. Notice, however, that for a computation of a Type 2 machine any finite portion of the output can be obtained already from a finite initial part of the possibly infinite computation, and for this only finite initial parts of the input tapes are relevant. This means that the behaviour of a Type 2 machine can be *approximated* adequately by its *behaviour in the finite*. In this sense also Type 2 machines and their computations can be realized physically. Therefore, any Type 2 computable function may be called "intuitively computable" or "physically computable".

Instead of Type 2 machines any other common computability model (e.g. FORTRAN or PASCAL programs) may be used for definition and study of the computable functions $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$, provided inputs and outputs are one–way (finite or infinite) files of symbols. Merely the definition of computational complexity depends crucially on the computability model. Below, we shall use Type 2 machines for this purpose.

Type 2 machines can be considered as a certain kind of *oracle Turing machines* (Rogers [Rog 67], Hopcroft and Ullman [HU 79]). Several other computable functions of higher types have been introduced, e.g. enumeration operators $\Phi_z : 2^\omega \longrightarrow 2^\omega$, *partial recursive operators* $F :\subseteq PF \longrightarrow PF$, *partial recursive functions* $F :\subseteq \omega^\omega \times \omega \longrightarrow \omega$ and $F :\subseteq 2^\omega \times \omega \longrightarrow \omega$, *partial recursive functionals* $F :\subseteq PF \longrightarrow \omega \cup \{*\}$ where $PF = \{f \mid f :\subseteq \omega \longrightarrow \omega\}$ (see Rogers [Rog 67] §§ 9.7, 9.8, 15.1 - 3) and *computable functions* $F :\subseteq \omega^\omega \longrightarrow \omega^\omega$ (Weihrauch [Wei 87]). Each of these definitions can be derived from our Type 2 computability and vice versa by using appropriate "natural" encodings. Therefore, it is very likely that every "intuitively computable" function $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ is computable by a Type 2 machine.

The above considerations support the following generalization of the Church/Turing thesis.

Generalized Church/Turing Thesis

A function $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ $(Y_0, \ldots, Y_k \in \{\Sigma^*, \Sigma^\omega\})$ is computable informally or by a physical device, if and only if it can be computed by a Type 2 machine.

Like Church's Thesis, also this more comprehensive thesis cannot be proved. In the definition of Type 2 machines we have restricted input and output tapes to be one–way. For input tapes and for output tapes with finite output this restriction is inessential, because a two–way input tape can be simulated by a one–way input tape and a work tape, and for halting computations a two–way output tape can be simulated by a work tape and a one–way output tape. The one–way output for infinite computations, however, is an essential restriction (see Example 4 below).

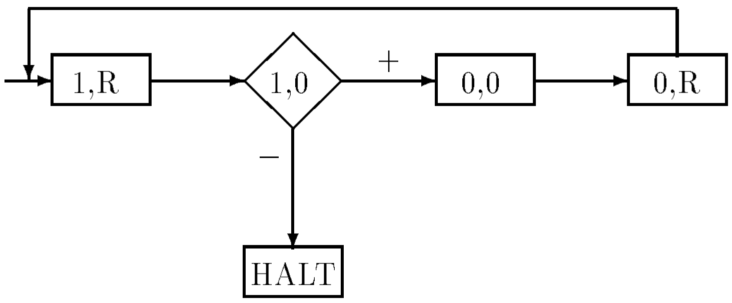
Among other proposed basic computational models for defining computability on

Type 2 objects like 2^ω, ω^ω etc. the Type 2 machines are particularly simple and concrete, they admit to explain the topological connection between classical analysis and computational theory in a very transparent way, and moreover they admit a direct definition of very realistic computational complexities as we shall show later on. We illustrate the definition of Type 2 computability by several examples.

Example 1
Let $\Sigma := \{0,1\}$ and define $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$ by

$$\begin{aligned} f(0^\omega) &:= \mathit{div} \\ f(0^i 1 p) &:= 0^i \text{ for all } i \in \omega \text{ and } p \in \Sigma^\omega. \end{aligned}$$

The following flowchart copies the leftmost zeros from the input tape 1 to the output tape 0. It halts, iff the input is not 0^ω.

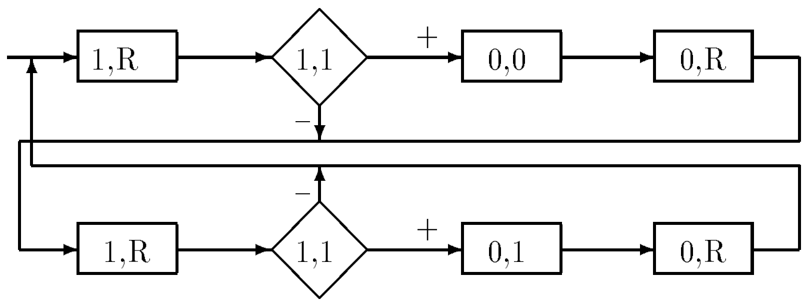


The flowchart together with the type specification $(\Sigma^\omega, \Sigma^*)$ defines a Type 2 machine which computes the function f.

Example 2
Let $\Sigma := \{0,1\}$ and define $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ by

$$f(p)(n) := \begin{cases} \mathit{div} & \text{if } \{i \mid p(i) = 1\} \text{ is finite, else} \\ 0 & \text{if } h(p,n) \text{ is even} \\ 1 & \text{if } h(p,n) \text{ is odd} \end{cases}$$

where $h(p,n)$ is the position of the $(n+1)$th one in p (i.e. $h(p,n)$ is that number i, for which $p(i) = 1$ and $\mathit{card}\{k < i \mid p(k) = 1\} = n$). The following flowchart together with the type specification $(\Sigma^\omega, \Sigma^\omega)$ defines a Type 2 machine which computes the function f.



From now on we shall no longer specify Turing machine flowcharts in full detail but give only informal descriptions. Type specifications will be given implicitly by the context.

Example 3

Consider the problem of dividing real numbers by 3, where the real numbers are represented by infinite decimal fractions (decimal expansions). The well–known paper and pencil method by reading the input left to right and writing the output left to right can be programmed easily by a Type 2 machine without work tapes. The nth output symbol $b_n \in \{0, \dots 9\}$ and the nth remainder $r_n \in \{0, 1, 2\}$ are determined by the symbol $a_n \in \{0, \dots 9\}$ and the previous remainder $r_{n-1} \in \{0, 1, 2\}$ as follows:

$$10 \cdot r_{n-1} + a_n = 3 \cdot b_n + r_n .$$

The sign and the decimal point must only be copied from the input to the output tape. A flowchart consisting of 3 sequences (one for each previous remainder $\in \{0, 1, 2\}$) of 10 consecutive tests (one for each symbol $\in \{0, \dots, 9\}$) plus write–statements etc. solves the problem. We omit a detailed flowchart.

Example 4

Consider the problem of *multiplying* real numbers by 3, where the real numbers are represented by infinite decimal fractions. The school method for multiplying finite decimal fractions adds intermediate results from right to left. It is also possible to perform the addition from left to right. In this case, however, from time to time carries may appear, which run from right to left switching nines to zeros.

right to left addition

```
0.  2  4  3  3  8  6  6  6  7  ·  3
    6
    1  2
          9
             9
             2  4
                1  8
                   1  8
                      1  8
                         2  1
0.  7  3  0  1  6  0  0  0  1
```

left to right addition

```
0.  2  4  3  3  8  6  6  6  7  ·  3
    6
    1  2
          9
             9
             2  4
                1  8
                   1  8
                      1  8
                         2  1
0.  7  2  9
       3  0  1  5  9  9
                6  0  0  0  1
```

This method with left to right addition can be applied also to infinite decimal fractions. It can be implemented easily on a modified Type 2 machine which has a two–way output tape.

Now we show that no Type 2 machine multiplies infinite decimal fractions by 3.

Assume that there is a Type 2 machine M which muliplies infinite decimal fractions by 3. Consider the input $p = 0.333\ldots = 0.3^\omega$. Then M must produce the output $q = 1.000\ldots = 1.0^\omega$ or the output $q = 0.999\ldots = 0.9^\omega$. Consider the case $q = 1.0^\omega$. There is a computation step in which M writes the first symbol 1 on the output tape. Up to this step M has read only the first k symbols (for some $k \in \omega$) from the input tape. Consider the input sequence $q' := 0.3^k0^\omega$. Since the first k symbols of q and q' coincide, also with input q' the machine M will write the symbol 1 as the first output. But since M is a multiplier, it must write 0.9^k0^ω on the output tape. This is a contradiction. The case $q = 0.9^\omega$ is handled accordingly.

Therefore, no Type 2 machine multiplies infinite decimal fractions by 3. Also the more general problem of multiplying two real numbers in decimal representation cannot be solved by a Type 2 machine. By Example 4 two–way output is strictly more powerful than one–way output. We continue with examples for non–computable functions.

Example 5

Let $\Sigma := \{0, 1\}$ and define $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$ by

$$f(p) := \begin{cases} 0 & \text{if } p = 0^\omega \\ 1 & \text{otherwise.} \end{cases}$$

We show that f is not computable. Assume that some Type 2 machine computes f. Consider the input $p := 00\ldots = 0^\omega$. Then for some number $k \in \omega$, M will produce the output 0 in k steps. Consider the input $p' := 0^k10^\omega$. Since the first k symbols of

p and p' coincide, and since M can read in k steps at most k symbols, M halts with output 0 also for input p'. Since $f(p') = 1$, M cannot compute the function f.

Example 6

The function f from Example 1 has no computable proper extension. Assume, on the contrary, that for some $w \in \Sigma^*$ the function $f' : \Sigma^\omega \longrightarrow \Sigma^*$, defined by $f(0^\omega) = w$, $f(0^i 1p) = 0^i$ for all $i \in \omega$ and $p \in \Sigma^\omega$, is computed by some Type 2 machine M. Then with input 0^ω for some $k \in \omega$, M will produce the output w in k steps. Especially, this implies $lg(w) \leq k$. Consider the input $q := 0^{k+1}10^\omega$. Since the first k symbols of 0^ω and q coincide and since M can read in k steps at most k symbols, M halts with output w also for input q. Since $lg(w) \leq k$ we obtain $f(q) := 0^{k+1} \neq w = f_M(q)$. Therefore, M cannot compute f.
In the same way it can be shown that also the function $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ from Example 2 has no computable proper extension.

In the proofs in Examples 4, 5 and 6 only the following fundamental finiteness property of computable functions $f :\subseteq \Sigma^\omega \longrightarrow Y$ ($Y \in \{\Sigma^*, \Sigma^\omega\}$) has been used:

Finiteness property (*for computable functions*)

> If $f(z) = y$, then any finite prefix of the output y is already determined by some finite portion of the input z.

This finiteness property is equivalent to *continuity* w.r.t. the *Cantor topology* on Σ^ω and the *discrete topology* on Σ^*.

Definition 2.4 (*Cantor topology on Σ^ω, discrete topology on Σ^**)

(1) $\tau_d := \{A \mid A \subseteq \Sigma^*\}$ is called the *discrete topology on Σ^**.

(2) $\tau_C := \{A\Sigma^\omega \mid A \subseteq \Sigma^*\}$ is called the *Cantor topology on Σ^ω*,
(Σ^ω, τ_C) is called the *Cantor space* (over Σ).

Every set $A \subseteq \Sigma^*$ is τ_d–open (i.e. $A \in \tau_d$). A set $U \subseteq \Sigma^\omega$ is τ_C–open (i.e. $U \in \tau_C$), iff there is some $A \subseteq \Sigma^*$ with ($p \in U \iff (\exists w \in A)$ w is a prefix of p) for all $p \in \Sigma^\omega$. If $p \in U$, already a finite prefix w of p suffices to prove this property. The topology τ_d can be generated from a *metric space*. For $p, q \in \Sigma^\omega$ define the distance

$$d(p,q) := \begin{cases} 0 & \text{if } p = q \\ 2^{-n} & \text{where } n \text{ is the length of the longest} \\ & \text{common prefix, otherwise.} \end{cases}$$

It is easy to show that (Σ^ω, d) is a metric space. A subset $X \subseteq \Sigma^\omega$ is an open ball, iff it is a closed ball, iff $X = w\Sigma^\omega$ for some word $w \in \Sigma^*$ ($w\Sigma^\omega = B(w0^\omega, 2 \cdot 2^{-n}) = Bc(w0^\omega, 2^{-n})$ where $n := lg(w)$). The set of open balls $\{w\Sigma^\omega \mid w \in \Sigma^*\}$ is a *basis* of the Cantor topology τ_C. On cartesian products $Y_1 \times \ldots \times Y_k$ ($Y_1, \ldots, Y_k \in \{\Sigma^*, \Sigma^\omega\}$) we consider the product topologies.

For functions $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ the finiteness property can be formulated as follows. Assume $f(z) = y$. Then for any open ball $B(y, \varepsilon)$ there is some open ball $B(z, \delta)$ such that $f(B(z, \delta)) \subseteq B(y, \varepsilon)$. But this means that f is continuous in z, i.e. the finitenness property is equivalent to continuity.

Theorem 2.5 (*computable $\Longrightarrow$ continuous*)

Every *computable function* $f :\subseteq Y_1 \times Y_2 \times \ldots \times Y_k \longrightarrow Y_0$ is *continuous.*

Proof

Let $f(y_1, \ldots y_k) = y_0$. Consider the case $Y_0 = \Sigma^\omega$. It suffices to show that for any neighbourhood $w_0\Sigma^\omega$ of y_0 there is some neighbourhood X of $(y_1, \ldots, y_k)$ with $f(X) \subseteq w_0\Sigma^\omega$. Let M be a Type 2 machine which computes f. Let $w_0\Sigma^\omega$ be a neighbourhood of y_0. Then M with input $(y_1, \ldots, y_k)$ writes the prefix w_0 of y_0 in finitely many steps. During this computation only the prefix w_i of the input y_i on Tape i can be read ($i = 1, \ldots, k$). Then $X := w_1Y_1 \times \ldots \times w_kY_k$ is an open neighbourhood of $(y_1, \ldots, y_k)$ with $f(X) \subseteq w_0\Sigma^\omega$. The case $Y_0 = \Sigma^*$ can be proved similarly.
$\square$

Therefore, for functions on Σ^* and Σ^ω continuity is a necessary condition for computability (only continuous functions can be computable). Continuity, i.e. the finiteness property of functions, is a very elementary constructivity property. In each of the examples 4, 5 and 6 we have proved discontinuity of the function under consideration. Of course there are also continuous functions which are not computable.

Example 7

Let $d : \omega \longrightarrow \omega$ be a total function with *range* $(d) \subseteq \{0, 1\}$ which is not computable. Then the functions $f :\subseteq \Sigma^* \longrightarrow \Sigma^\omega$, $g :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$ and $h : \Sigma^\omega \longrightarrow \Sigma^\omega$ are continuous but not computable, where:

$$\begin{array}{lcll} f(w)(n) & := & d(n) & \text{for all } w \in \Sigma^*,\ n \in \omega; \\ g(0^\omega) & := & div & \\ g(0^k1q) & := & d(k) & \text{for all } k \in \omega,\ q \in \Sigma^\omega; \\ h(q)(n) & := & d(n) & \text{for all } q \in \Sigma^\omega,\ n \in \omega. \end{array}$$

From a Type 2 machine for f, g or h one could construct a Turing machine computing the function d.

An important object in ordinary recursion theory (Rogers [Rog 67]) is an "effective Gödel numbering" $\varphi : \omega \longrightarrow P^{(1)}$ of the set $P^{(1)}$ of the computable functions $f :\subseteq \omega \longrightarrow \omega$. The theory of Type 2 computability can be deepened by introducing "effective" notations $\zeta^{ab} :\subseteq \Sigma^* \longrightarrow P^{ab}$ of the sets P^{ab} of the computable functions $f :\subseteq \Sigma^a \longrightarrow \Sigma^b$ and "effective" representations $\eta^{ab} :\subseteq \Sigma^\omega \longrightarrow F^{ab}$ of certain sets F^{ab} of continuous functions $f :\subseteq \Sigma^a \longrightarrow \Sigma^b$ $(a, b \in \{*, \omega\})$. Definitions and some properties are given in Appendix B. For details see [Wei 94]. These naming systems, however, will not be used in the following.

The composition of computable functions is computable or has a computable extension. For simplicity we consider only unary functions.

Theorem 2.6 (*composition of computable functions*))

Let $f :\subseteq Y_1 \longrightarrow Y_2$ and $g :\subseteq Y_2 \longrightarrow Y_3$ $(Y_1, Y_2, Y_3 \in \{\Sigma^*, \Sigma^\omega\})$ be computable.

- If $(Y_2, Y_3) \neq (\Sigma^\omega, \Sigma^*)$, then gf is computable.
- If $(Y_2, Y_3) = (\Sigma^\omega, \Sigma^*)$, then gf has a computable extension h such that $dom\,(gf) \cap dom\,(f) = dom\,(h) \cap dom\,(f)$.

Proof

Let M_f and M_g be Type 2 machines computing f and g, respectively. It is possible to construct from M_f and M_g a Type 2 machine M, which simulates alternately the computations of M_f and M_g taking in turn the output symbols of M_f as the input symbols for M_g: M_g is simulated until it requires the first input symbol, M_f is simulated until it produces the first output symbol, M_g is simulated until it requires the next input symbol, etc.. The computable function f_M has the desired properties. $\square$

As a simple consequence of Theorem 2.6, computable functions map computable elements to computable elements. A subset $A \subseteq \Sigma^*$ is recursively enumerable (r.e.), iff $A = dom\,(f)$ for some computable function $f :\subseteq \Sigma^* \longrightarrow \Sigma^*$, and A is recursive (or decidable), iff A and $\Sigma^* \setminus A$ are r.e. We generalize these basic definitions from recursion theory as follows:

Definition 2.7 (*r.e. and recursive sets*)

Consider $k \geq 0$ and $Y_1, \ldots, Y_k \in \{\Sigma^*, \Sigma^\omega\}$.

(1) A set $X \subseteq Y_1 \times \ldots \times Y_k$ is called *recursively enumerable (r.e.)*, iff $X = dom\,(f)$ for some computable function $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow \Sigma^*$.

(2) For $U \subseteq W \subseteq Y_1 \times \ldots \times Y_k$ we call U *r.e. in* W, iff $U = W \cap X$ for some r.e. set X.

(3) For $U \subseteq W \subseteq Y_1 \times \ldots \times Y_k$ we call U *recursive* (or *decidable*) *in* W, iff U and $W \setminus U$ are r.e. in W.

Assume $M = dom\,(f_M)$, where $f_M :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow \Sigma^*$ for some Type 2 machine M. This machine M is an "abstract proof system" for the set $X = dom(f_M)$. If $y \in X$ then M applied to input y halts. The finite computation can be considered as a proof for the property "$y \in X$" in this proof system. If $y \notin X$, then there is no such a proof.

If $y \in dom\,(f_M)$, then only a finite portion of the possible infinite input y can be read by M during its finite computation. Therefore, any r.e. set is open. Any open set $X \subseteq \Sigma^\omega$ has the form $A\Sigma^\omega$ for some $A \subseteq \Sigma^*$. It is easy to show that $X \subseteq \Sigma^\omega$ is r.e., iff $X = A\Sigma^\omega$ for some r.e. (even for some recursive) subset $A \subseteq \Sigma^*$. U is recursive in W, iff there is a computable function $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow \Sigma^*$ with $W \subseteq dom\,(f)$ and $U = f^{-1}\{\varepsilon\} \cap W$, i.e. $f(y) = \varepsilon \iff y \in U$ for all $y \in W$. The sets $U \subseteq \Sigma^\omega$ recursive in Σ^ω are particularily simple: U is recursive in Σ^ω, iff $U = A\Sigma^\omega$ for some *finite* set $A \subseteq \Sigma^*$. This follows from compactness of Σ^ω.

Finite or infinite sequences of symbols can be used as *names* of other objects like natural numbers, rational numbers, finite graphs, rational matrices, real numbers, subsets of ω etc.. Examples are the binary notation $\nu_{bin} :\subseteq \Sigma^* \longrightarrow \omega$ of the natural numbers and the decimal representation $\nu_{dec} :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ of the real numbers (where Σ is a sufficiently large alphabet). We introduce *naming systems* and *reducibilities* for comparing them.

Definition 2.8 (*notations, representations, reducibility*)

(1) A *naming system* of a set M is a *notation* or a *representation* of M, where a *notation* is a surjective function $\nu :\subseteq \Sigma^* \longrightarrow M$ (naming by finite strings) and a *representation* is a surjective function $\delta :\subseteq \Sigma^\omega \longrightarrow M$ (naming by infinite sequences).

(2) For functions $\gamma :\subseteq Y \longrightarrow M$ and $\gamma' :\subseteq Y' \longrightarrow M'$ with $Y, Y' \in \{\Sigma^*, \Sigma^\omega\}$ we call γ *reducible to* γ', $\gamma \le \gamma'$, iff $(\forall y \in dom\,(\gamma))\;\; \gamma(y) = \gamma' f(y)$ for some computable function $f :\subseteq Y \longrightarrow Y'$. We call γ and γ' *equivalent*, $\gamma \equiv \gamma'$, iff $\gamma \le \gamma'$ and $\gamma' \le \gamma$.
Topological reducibility $\le_t$ and *topological equivalence* $\equiv_t$ are defined accordingly by substituting "continuous" for "computable".

If $\gamma = \gamma' f$, we may say that the function f "translates" the naming system γ into the naming system γ' (examples: translation from PASCAL to ASSEMBLER, from English to German). For a naming system $\gamma :\subseteq Y \longrightarrow M$ there are informations about the elements of M, which can be obtained computationally from their names. Translation cannot increase this information. If $\gamma \leq \gamma'$ and $\gamma' \not\leq \gamma$, we may say that γ–names contain more computationally available information than γ'–names. As an example consider the following two notations γ_1 and γ_2 of Σ^*. Let $A \subseteq \Sigma^*$ r.e. and not recursive. Define $\nu_1(w) = w$ for all $w \in \Sigma^*$, $\nu_2(0w) = w$ if $w \in A$, $\nu_2(1w) = w$ if $w \notin A$, $\nu_2(x) = div$ otherwise. Then obviously, $\nu_2 \leq \nu_1$ but $\nu_1 \not\leq \nu_2$. The first symbol of any ν_2–name of a word w is the answer to the (undecidable) question "$w \in A$?". This is not the case for ν_1–names. We illustrate Definition 2.8 by an example.

Example 8

Let $d \in \omega$, $d \geq 2$, and let Σ be an alphabet with $\{., 0, 1, \dots, d-1\} \subseteq \Sigma$. For any $a \in \omega$, $2 \leq a \leq d$, define a notation $\nu_a :\subseteq \Sigma^\omega \longrightarrow \omega$ of the natural numbers and a representation $\delta_a :\subseteq \Sigma^\omega \longrightarrow R_{+0}$ of the non–negative real numbers as follows (where $\Sigma_a := \{0, \dots, a-1\}$):

$$\begin{aligned}
dom\,(\nu_a) &:= \Sigma_a^* \setminus \{\varepsilon\} \\
\nu_a(a_k \dots a_0) &:= a_k a^k + \dots + a_0 a^0 \quad (a_i \in \Sigma_a) \\
dom\,(\delta_a) &:= \Sigma_a^* . \Sigma_a^\omega \\
\delta_a(a_k \dots a_0 . a_{-1} a_{-2} \dots) &:= a_k a^k + \dots + a_0 a^0 + a_{-1} a^{-1} + \dots \quad (a_i \in \Sigma_a)
\end{aligned}$$

Let $P_a := \{e \in \omega \mid e \text{ is a prime factor of } a\}$. Then for any $a, b \in \{2, \dots, d\}$ the following properties hold::

(1) $\nu_a \equiv \nu_b$,

(2) $\nu_a \leq \delta_b$,

(3) $\delta_a \leq \delta_b$, if $P_b \subseteq P_a$,

(4) $\delta_a \not\leq_t \delta_b$, if $P_b \not\subseteq P_a$.

There is a Turing machine which translates a–adic numbers into b–adic numbers, i.e. $\nu_a \leq \nu_b$. By symmetry we have also $\nu_b \leq \nu_a$, hence $\nu_a \equiv \nu_b$. The computable function $f :\subseteq \Sigma^* \longrightarrow \Sigma^\omega$ with $f : w \mapsto w.0^\omega$ translates ν_b into δ_b. Since $\nu_a \leq \nu_b$ (by (1)) we have $\nu_a \leq \delta_b$. It is not very difficult to design a Type 2 machine which translates δ_a to δ_b if $P_b \subseteq P_a$. Consider $e \in P_b \setminus P_a$. Then $1/e = \delta_b(.c(b-1)(b-1)\dots) = \delta_b(.(c+1)00\dots)$ for some $c \in \Sigma_b$, but $1/e$ has a unique δ_a–name p, for which neither

$p \in \Sigma^* 0^\omega$ nor $p \in \Sigma^*(a-1)^\omega$. As in Example 4 or 5 it can be shown that there is no continuous translator which is correct for input p. Details are left to the reader.

A naming system $\gamma :\subseteq Y \longrightarrow M$ transforms effectivity concepts from Y to M. First we define computable points and open, r.e. and recursive subsets.

Definition 2.9

Let $\gamma_i :\subseteq Y_i \longrightarrow M_i$ $(i = 1, \ldots, k)$ be naming systems.

(1) $x \in M_1$ is γ_1–computable, iff there is a computable element $y \in Y_1$ with $\gamma_1(y) = x$.

(2) $X \subseteq M_1 \times \ldots \times M_k$ is $(\gamma_1, \ldots, \gamma_k)$–*open* (–*r.e.*, –*recursive*), iff

$$\{(y_1, \ldots, y_k) \in Y_1 \times \ldots \times Y_k \mid (\gamma_1(y_1), \ldots, \gamma_k(y_k))) \in X\}$$

is open (r.e., *recursive*) in $dom\,(\gamma_1) \times \ldots \times dom\,(\gamma_k)$.

For any naming system $\gamma :\subseteq Y \longrightarrow M$, the set $\tau_\gamma := \{X \subseteq M \mid X$ is γ–open $\}$ is called the *final topology* of γ.

Example 9

Let $\nu_{bin} :\subseteq \Sigma^* \longrightarrow \omega$ be the binary notation of ω. Every $n \in \omega$ is ν_{bin}–computable, every subset $A \subseteq \omega$ is ν_{bin}–open. A subset $A \subseteq \omega$ is ν_{bin}–r.e., iff it is r.e.. Let δ_{dec} be the representation of real numbers by infinite decimal fractions.

(1) Every rational number is δ_{dec}–computable. For a proof notice that the decimal names of the rational numbers are periodic.

(2) $\sqrt{2}$ is δ_{dec}–computable. A simple trial and error search by squaring finite decimal fractions yields a sequence $p \in \Sigma^\omega$ with $\delta_{dec}(p) = \sqrt{2}$.

(3) For any $X \subseteq \mathbb{R}$, X is open $\iff$ X is δ_{dec}–open.

We sketch a proof. Let $X \subseteq \mathbb{R}$ be open. Let $p \in \delta_{dec}^{-1}X$. Since X is open, there is some $n \in \omega$ with $[\delta_{dec}(p) - 10^{-n}; \delta_{dec}(p) + 10^{-n}] \subseteq X$. Let $w_p \in \Sigma^*$ be the prefix of p containing the first n digits after the decimal point. Then $\delta_{dec}(w_p\Sigma^\omega \cap dom\,(\delta_{dec})) \subseteq X$, i.e. $p \in w_p\Sigma^\omega \cap dom\,(\delta_{dec}) \subseteq \delta_{dec}^{-1}(X)$. This means, p has an open neighbourhood in $\delta_{dec}^{-1}X$. Therefore $\delta_{dec}^{-1}X$ is open in $dom\,(\delta)$. Let X be δ–open. Consider $x \in X$ and $x > 0$ (w.l.g.). There is some $p \in \Sigma^\omega$ with $\delta_{dec}(p) = x$ such that $q \neq 9^\omega$ whenever $p = wq$ with $w \in \Sigma^*$ and $q \in \{0, \ldots, 9\}^\omega$. Since $\delta_{dec}^{-1}X$ is open in $dom\,(\delta_{dec})$ there are some $w \in \Sigma^*$ and $q \in \{0, \ldots, 9\}^\omega$ with $p = wq$ and $w\Sigma^\omega \cap dom\,(\delta_{dec}) \subseteq \delta_{dec}^{-1}(X)$. Since $q \neq 9^\omega$ we obtain

$$x = \delta_{dec}(wq) \in [\delta_{dec}(w0^\omega); \delta_{dec}(w9^\omega)) \subseteq \delta_{dec}(w\Sigma^\omega) \subseteq X.$$

There is another $p' \in \Sigma^\omega$ with $\delta_{dec}(p') = x$ and $q \neq 0^\omega$ whenever $p' = wq$ with $w \in \Sigma^*$ and $q \in \{0, \ldots, 9\}^*$. As in the first case we conclude that there is some prefix w' of p' with

$$x \in (\delta_{dec}(w'0^\omega);\ \delta_{dec}(w'9^\omega)] \subseteq X.$$

Therefore $x \in I \subseteq X$ for some open interval I, This shows that X is open.

Notice that the final topology τ_γ of $\gamma :\subseteq Y \longrightarrow M$ is indeed a topology on M. Next, we define relative computability and continuity of functions and relations.

Definition 2.10 (*relatively effective relations and functions*)

For $i = 0, \ldots, k$ let $\gamma_i :\subseteq Y_i \longrightarrow M_i$ be naming systems.

(1) A relation $Q \subseteq M_1 \times \ldots \times M_k \times M_0$ is $(\gamma_1, \ldots, \gamma_k, \gamma_0)$–*computable* (–*continuous*), iff there is some computable (continuous) function $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ with

$$(\gamma_1(y_1), \ldots, \gamma_k(y_k), \gamma_0 f(y_1, \ldots, y_k)) \in Q$$

whenever $\exists x.(\gamma_1(y_1), \ldots, \gamma_k(y_k), x) \in Q$.

(2) A function $F \subseteq M_1 \times \ldots \times M_k \longrightarrow M_0$ is $(\gamma_1, \ldots, \gamma_k, \gamma_0)$–*computable* (–*continuous*), iff there is some computable (continuous) function $f :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ with

$$F(\gamma_1(y_1), \ldots, \gamma_k(y_k)) = \gamma_0 f(y_1, \ldots, y_k)$$

whenever $F(\gamma_1(y_1), \ldots, \gamma_k(y_k))$ exists.

Relative computability (continuity) of a relation Q can be considered as an effective version of the mere existence statement $(\forall x_1, \ldots, x_k)(\exists x_0)Q(x_1, \ldots, x_k, x_0)$. If Q is $(\gamma_1, \ldots, \gamma_k, \gamma_0)$–computable, some computable function f transforms any name $(y_1, \ldots, y_k)$ of $(x_1, \ldots, x_k)$ from the domain of Q into a name y_0 of some x_0 such that $Q(x_1, \ldots, x_k, x_0)$. Roughly speaking, for each $(x_1, \ldots, x_k)$ we can determine some x_0 with $Q(x_1, \ldots, x_k, x_0)$. If in (1), $(\gamma_1(y_1), \ldots, \gamma_k(y_k)) = (\gamma_1(y_1'), \ldots, \gamma_k(y_k'))$ implies $\gamma_0 f(y_1, \ldots, \gamma_k) = \gamma_0 f(y_1', \ldots, y_k')$, then there is a (computable or continuous) function $G :\subseteq M_1 \times \ldots \times M_k \longrightarrow M_0$ with $(x_1, \ldots, x_k, G(x_1, \ldots, x_l)) \in Q$. Such a function G is called a choice function of Q. A relation Q may be computable without having a continuous choice function (see Example 10 below). Definition 2.10(2) is a special case of 2.10(1) where Q is single–valued. A function F is $(\gamma_1, \ldots, \gamma_k, \gamma_0)$–continuous, iff some continuous function transforms any name $(y_1, \ldots, y_k)$ of some $(x_1, \ldots, x_k) \in dom(F)$ into some name y_0 of $F(x_1, \ldots, x_k)$. Notice that by definition every restriction F' of F is $(\gamma_1, \ldots, \gamma_k, \gamma_0)$–continuous (–computable) if F is $(\gamma_1, \ldots, \gamma_k, \gamma_0)$–continuous (–computable). A Type 2 machine "computing" some relation Q or some function F actually transforms merely sequences of symbols; it is the user who interprets these sequences as names of objects.

Example 10
Define the enumeration representation $En :\subseteq \Sigma^\omega \longrightarrow 2^\omega$ of the set of subsets of ω by

$$n \in En(p) :\iff 10^{n+1}11 \text{ is a subword of } p$$

for all $n \in \omega$ and $p \in \Sigma^\omega$, and let $\nu_{bin} :\subseteq \Sigma^\omega \longrightarrow \omega$ be the binary notation of ω. (We assume $\{0,1\} \subseteq \Sigma$.) Then the following properties hold.

(1) $A \subseteq \omega$ is En–computable $\iff$ A is r.e..

(2) $\{(A,n) \in 2^\omega \times \omega \mid n \in A\}$ is (En, ν_{bin})–r.e..

(3) $\{(A,n) \in 2^\omega \times \omega \mid n \notin A\}$ is not (En, ν_{bin})–open.

(4) $\{(A,n) \in 2^\omega \times \omega \mid n \in A\}$ is (En, ν_{bin})–computable.

(5) There is no (En, ν_{bin})–continuous function $f :\subseteq 2^\omega \longrightarrow \omega$ with $f(A) \in A$ for all $A \neq \emptyset$.

We sketch the proofs:

(1) This is a simple recursion theoretic exercise.

(2) Let M be a Type 2 machine which for input (p,w) searches in p for the subword $10^{n+1}11$, where $n := \nu_{bin}(w)$. M halts, iff such a subword has been found.

(3) We have $(En(0^\omega), \nu_{bin}(0)) \in Q_3 := \{(A,n) \mid n \notin A\}$. Assume, Q_3 is (En, ν_{bin})–open. Then for some k, $(En(q), \nu_{bin}(0)) \in Q_3$ for all $q \in 0^k\Sigma^\omega$. But $q := 0^k10110^\omega \in 0^k\Sigma^\omega$ and $(En(q), \nu_{bin}(0)) \notin Q_3$.

(4) Let M be a Type 2 machine with $f_M :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$, which searches in input $p \in \Sigma^\omega$ for the first appearence of a subword $10^{m+1}11$. If such a word is found then a word w with $\nu_{bin}(w) = m$ is written on the output tape.

(5) Suppose, there is some continuous function $g :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$ with $\nu_{bin}g(p) = f\,En(p)$ whenever $En\,(p) \neq \emptyset$. Then $g(10110^\omega) = 0$, $g(100110^\omega) = 1$. By continuity of g there is some k with $g(10110^k\Sigma^\omega) = \{0\}$, $g(100110^k\Sigma^\omega) = \{1\}$. Let $p := 10110^k100110^\omega$ and $q := 100110^k10110^\omega$. Then $0 = \nu_{bin}g(p) = f\,En(p) = f\{0,1\} = f\,En(q) = \nu_{bin}g(q) = 1$ (contradiction).

Notice that the set $Q = \{(A,n) \mid n \in A\}$ is (En, ν_{bin})–computable but has not even a (En, ν_{bin})–continuous choice function $f :\subseteq 2^\omega \longrightarrow \omega$.

The computability concepts induced on sets by naming systems (Def. 2.9, 2.10) remain unchanged, if the naming systems are replaced by equivalent ones, and correspondingly the induced topological properties remain unchanged, if the naming systems are replaced by topologically equivalent ones. For the proof only the fact that the computable and the continuous functions are closed under composition is needed.
On the other hand, non–equivalent naming systens induce different computability

theories on M. Therefore, the induced effectivity concepts on a set (according to Defs. 2.9 and 2.10) depend crucially on the underlying naming system.

In TTE, computability on a set M is introduced in two steps:

(1) definition of computable functions on finite or infinite sequences of symbols,

(2) definition of a naming system $\gamma :\subseteq Y \longrightarrow M$.

As for number functions we are not interested in arbitrary computability concepts on M but only in those which meet some intuition, which are "natural". In Step 1, which is independent of M, we choose the Type 2 computable functions, which are "effective" by our generalized Church/Turing thesis (see Chapter 2). "Effectiveness" of a naming system of a set M can be defined only relative to some structure on M. It is an essential feature of TTE that effectiveness of the introduced naming systems is justified by general principles.

3 Computability on the Real Numbers

In this chapter we introduce standard naming systems for the natural, the rational and the real numbers and study the induced computability. We give examples of computable real numbers, characterize the ρ_C–open, –r.e. and –recursive sets and prove computability of functions like addition and multiplication and of real analytic functions with computable power series.

From now on let Σ be a sufficiently large finite alphabet containing all the symbols we shall need. Let $\nu_{bin} :\subseteq \Sigma^* \longrightarrow \omega$ be the one–to–one binary notation (without leading zeros) of the natural numbers, and let $\nu_Q :\subseteq \Sigma^* \longrightarrow \omega$ be the one–to–one binary notation of the rational numbers by signed reduced fractions of binary numbers (for an exact definition see Appendix C). These notations and all the equivalent ones are usually called "effective". Are there other "effective" notations of the natural und rational numbers? In Appendix C we show, that the equivalence classes of ν_{bin} and ν_Q can be characterized by simple effectivity properties and a maximality principle. In the following text we shall use the abbreviations:

$$\overline{u} := \nu_{bin}(u) \quad \text{for all } u \in dom(\nu_{bin}),$$
$$\overline{u} := \nu_Q(u) \quad \text{for all } u \in dom(\nu_Q).$$

As an example, addition on ω is computable w.r.t. ν_{bin}, i.e. $f : \omega^2 \longrightarrow \omega$ with $f(x, y) := x + y$ is $(\nu_{bin}, \nu_{bin}, \nu_{bin})$–computable, in more detail: there is a computable function $g :\subseteq \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$ with $f(\overline{u}, \overline{v}) = \nu_{bin} g(u, v)$ for all $u, v \in Dom(\nu_{bin})$. Also multiplication, exponentiation, arithmetical subtraction and division, minimum and maximum are computable w.r.t. ν_{bin}. On the rational numbers addition, subtraction, multiplication, division, maximum and minimum are computable w.r.t. ν_Q. The most popular representation of the set $\mathbb{R}$ of the real numbers is that by infinite decimal fractions (decimal expansions), $\delta_{dec} :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$. Unfortunately, very simple functions like $x \mapsto 3x$ are not $(\delta_{dec}, \delta_{dec})$–computable (see Example 2.4). This already indicates that δ_{dec} in not adequate for a foundation of computability on $\mathbb{R}$, since real number multiplication should be computable. To overcome this problem we introduce two standard representations of $\mathbb{R}$, the interval representation and the Cauchy representation.

Definition 3.1 (*interval and Cauchy representations*)

Define two representations $\rho_I :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ (*interval representation*) and $\rho_C :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ (*Cauchy representation*) as follows:

$$\rho_I(p) = x, \quad \text{iff there are } u_0, v_0, u_1, v_1, \ldots \in dom(\nu_Q) \text{ with}$$
$$p = u_0\sharp v_0\sharp u_1\sharp v_1 \ldots \text{ and } x = \sup_{i\in\omega} \overline{u}_i = \inf_{i\in\omega} \overline{v}_i.$$
$$\rho_C(p) = x, \quad \text{iff there are } u_0, u_1, \ldots \in dom(\nu_Q) \text{ with}$$
$$p = u_0\sharp u_1\sharp \ldots, (\forall k)(\forall i > k)|\overline{u}_i - \overline{u}_k| < 2^{-k} \text{ and } x = \lim_{i\to\infty} \overline{u}_i.$$

If $p = u_0\sharp v_0\sharp u_1\ldots$ and $\rho_I(p) = x$ then x is the only point in the intersection of all closed intervals $[\overline{u}_i;\overline{v}_i]$. If $\rho_C(p) = x$ with $p = u_0\sharp u_1\sharp\ldots$, then $(\overline{u}_i)_{i\in\omega}$ is a Cauchy sequence of rational numbers converging to x with "speed" 2^{-i} and consequently, $|\overline{u}_k - x| \leq 2^{-k}$ for all $k \in \omega$. Therefore, we can associate with $u_0\sharp u_1\sharp\ldots$ the special sequence $(I_n)_{n\in\omega}$ of nested intervals where $I_n := [\overline{u}_n - 2^{-n};\overline{u}_n + 2^{-n}]$. Notice that we consider only these fast converging Cauchy sequences as names. First we compare the three representations δ_{dec}, ρ_I and ρ_C.

Lemma 3.2 (*relation between decimal and Cauchy representation*)

$$\delta_{dec} \leq \rho_I \equiv \rho_C$$

$$\rho_C \not\leq_t \delta_{dec}$$

Proof

$\delta_{dec} \leq \rho_I$: A Type 2 machine M can be constructed which with input $p \in \sigma a_k\ldots a_0.a_{-1}a_{-2}\ldots \in dom(\delta_{dec})$ $(\sigma \in \{+,-\}, a_i \in \{0,\ldots,9\})$ writes $u_0\sharp v_0\sharp u_1\sharp v_1\sharp\ldots$ on the output tape, where $\overline{u}_i = r$ and $\overline{v}_i = r + 10^{-i}$ if $\sigma = +$, $\overline{u}_i = -r - 10^{-i}$ and $\overline{v}_i = -r$ if $\sigma = -$, and $r \in Q$ is the rational value of the finite decimal fraction $a_k\ldots a_0.a_{-1}\ldots a_{-i}$. Then f_M translates δ_{dec} into δ_I, i.e. $\delta_{dec}(p) = \rho_I f_M(p)$ for all $p \in dom(\delta_{dec})$.

$\rho_I \leq \rho_C$: Let M be a Type 2 machine which with input $p \in u_0\sharp v_0\sharp u_1\sharp v_1\ldots \in dom(\rho_I)$ $(u_i, v_i \in dom(\nu_Q))$ writes $q := w_0\sharp w_1\sharp\ldots$ on the output tape, where for $i = 0,1,\ldots$ the word w_i is determined as follows. M searches for a pair of natural numbers (k,m) with $|\overline{v}_m - \overline{u}_k| < 2^{-i}$ and then sets $w_i = v_m$. Since $p \in dom(\rho_I)$, the search must be successful, its result guarantees $\overline{w}_i - 2^{-i} < \rho_I(p) \leq \overline{w}_i$, hence $|\overline{w}_i - \overline{w}_n| < 2^{-i}$ for all $n > i$. Therefore $\rho_I(p) = \rho_C(q)$.

$\rho_C \leq \rho_I$: Let M be a Type 2 machine which with input $p = w_0\sharp w_1\sharp\ldots \in dom(\rho_C)$ writes $u_0\sharp v_0\sharp u_1\sharp v_1\ldots$ on the output tape where $\overline{u}_i := \overline{w}_i - 2^{-i}$ and $\overline{v}_i := \overline{w}_i + 2^{-i}$. Then obviously $\rho_C(p) = \rho_I f_M(p)$.

$\rho_C \not\leq_t \delta_{dec}$: Assume, there is some continuous function $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ with $\rho_C(p) = \delta_{dec} f(p)$ for all $p \in dom(\rho_C)$. Since $\rho_C(1\sharp 1\sharp\ldots) = 1 \in \mathbb{R}$, $f(p) = 0.999\ldots$ or $f(p) = 1.000\ldots$. Consider the case $f(p) = 0.9^\omega$. Since f is continuous, there is some $n \in \omega$ such that $f((1\sharp)^n\Sigma^\omega) \subseteq 0.9\Sigma^\omega$. Let $\overline{u} := 1 + 2^{-n}$ and $q := (1\sharp)^n(u\sharp)^\omega$. Then $\rho_C(q) > 1$, but $f(q) \in 0.9\Sigma^\omega$, hence $\delta_{dec} f(q) \leq 1 < \rho_C(q)$. Therefore f does not translate q correctly. The case $f(p) = 1.0^\omega$ is handled accordingly.
□

By Lemma 3.2, the decimal–names contain more continuously accessible information than ρ_C–names. Below we shall give convincing arguments that not the decimal representation but the Cauchy–representation is adequate for defining computability on the real line. Since $\rho_C \equiv \rho_I$ we may use also ρ_I instead of ρ_C for investigating computability on $\mathbb{R}$, whenever appropriate.

Convention

In the following, ν_{bin}, ν_Q and ρ_C will be our standard naming systems of $\omega, \mathbb{Q}$ and $\mathbb{R}$, respectively. For simplicity, in connection with "computable", "r.e." and "recursive" we shall omit prefixes like ν_{bin}–, (ρ_C, ν_Q)– etc. and shall say "computable" instead of "(ν_{bin}, ν_{bin})–computable", "r.e." instead of $(\nu_Q, \rho_C, \nu_{bin})$–r.e. etc..

By Definition 2.9, the *computable real numbers* are (by the above convention) those numbers, which have computable ρ_C–names or computable ρ_I–names (Lemma 3.2).

Example 1 (*computable real numbers*)

(1) Every rational number is computable:
Consider $r \in \mathbb{Q}$. Define $u \in \Sigma^*$ by $\overline{u} = r$, define $q := u\sharp u\sharp \ldots = (u\sharp)^\omega \in \Sigma^\omega$. Then q is computable and $\rho_C(q) = r$.

(2) $\sqrt{2}$ is computable:
Define $f : \omega \longrightarrow \omega$ by $f(n) :=$ that $k \in \omega$ with $k^2 < 2 \cdot 2^{2n} \leq (k+1)^2$. Then f is computable. Let $\overline{u}_n := f(n) \cdot 2^{-n}$. Then $p = u_0\sharp u_1\sharp u_2 \ldots$ is computable and $\rho_C(p) = \sqrt{2}$.

(3) $\log_3 5$ is computable:
Define $f : \omega \longrightarrow \omega$ by $f(n) :=$ that $k \in \omega$ with $3^k < 5^n \leq 3^{k+1}$. Then f is computable. Let $\overline{u}_n := k/n$ and $\overline{v}_n := (k+1)/n$. Then $p = u_0\sharp v_0\sharp u_1\sharp v_1 \ldots$ is computable and $\rho_I(p) = \log_3(5)$.

(4) For $A \subseteq \omega$ define $x_A := \Sigma\{2^{-i} \mid i \in A\}$. Then

$$x_A \text{ is computable} \iff A \text{ is recursive.}$$

Assume that A is recursive. For $k \in \omega$ define $u_k \in \Sigma^*$ by $\overline{u}_k := \Sigma\{2^{-i} \mid i \in A, i \leq k\}$. Then $p = u_0\sharp u_1\sharp \ldots$ is computable with $x_A = \rho_C(p)$.

Assume that x_A is computable. For all $w = a_0 \ldots a_k$ ($k \in \omega, a_0, \ldots, a_k \in \{0,1\}$) let $x_w := \Sigma\{a_i \cdot 2^{-i} \mid i \leq k\}$. If $x_A = x_w$ for some $w \in \Sigma^*$ then x_A is computable by (1). Assume $x_A \neq x_w$ for all $w \in \Sigma^*$. By assumption, $x_A = \rho_C(p)$ for some computable $p = u_0\sharp u_1\sharp \ldots \in \Sigma^\omega$. For any $w \in \{0,1\}^*$ there is some $i \in \omega$ with $u_i + 2^{-i} < x_w$ or $x_w < u_i - 2^{-i}$. In the first case we have $x_A < x_w$, in the second case $x_w < x_A$. Therefore $W := \{w \mid x_w < x_A\}$ is decidable. Compute a sequence $y_0, y_1, \ldots$ of words inductively by $y_0 :=$ (1 if $x_1 < x_A$, 0 otherwise), $y_{k+1} :=$ (y_k1 if $x_{y_k1} < x_A$, y_k0 otherwise). Then 1 is the last symbol of y_k, iff $k \in A$. Therefore, A is recursive.

Further examples of computable real numbers can be obtained by applying computable functions to computable arguments (see below). The limit of any computable sequence of computable real numbers with computable *modulus of convergence* is computable:

Theorem 3.3 (*limit of computable sequence with computable convergence*)

Let $(y_i)_{i\in\omega}$ be a (ν_{bin}, ρ_C)–computable sequence of real numbers such that $(\forall i, j \geq m(n))|y_i - y_j| < 2^{-n}$ for some computable function $m : \omega \longrightarrow \omega$ (m is called a computable *modulus of convergence*). Then $x := \lim\limits_{i\to\infty} y_i$ is computable.

Proof

By assumption, for any $i, j \in \omega$ a word $u_{ij} \in dom(\nu_Q)$ can be computed such that $y_i = \rho_C(u_{i0}\sharp u_{i1}\sharp\ldots)$. Let $v_i := u_{m(i+1),i+2}$ for all $i \in \omega$. Then $q := v_0\sharp v_1\sharp\ldots$ is computable. For all $k > i$ we have

$$\begin{aligned}|\overline{v}_i - \overline{v}_k| &\leq |\overline{u}_{m(i+1),i+2} - y_{m(i+1)}| + |y_{m(i+1)} - y_{m(k+1)}| + |y_{m(k+1)} - \overline{u}_{m(k+1),k+2}|\\ &\leq 2^{-i-2} + \max(2^{-i-1}, 2^{-k-1}) + 2^{-k-2}\\ &< 2^{-i}\end{aligned}$$

and

$$\begin{aligned}|\overline{v}_i - x| &\leq |\overline{u}_{m(i+1),i+2} - y_{m(i+1)}| + |y_{m(i+1)} - x|\\ &\leq 2^{-i-2} + 2^{-i-1}\\ &\leq 2^{-i}.\end{aligned}$$

We obtain $x = \rho_C(q)$. Therefore, x is ρ_C–computable.
□

Example 2

Let $A \subseteq \omega$ be r.e. but not recursive. Define $x_A \in \mathbb{R}$ by $x_A := \Sigma\{2^{-i} \mid i \in A\}$. By Example 3.1, x_A is not computable. Since A is r.e. and not recursive, there is some total injective computable function $f : \omega \longrightarrow \omega$ with $A = range(f)$. Obviously, $x_A = \Sigma\{2^{-f(n)} \mid n \in \omega\}$. Define a sequence $s := (y_n)_{n\in\omega}$ by $y_n = \Sigma\{2^{-f(k)} \mid k \leq n\}$. Then s is (ν_{bin}, ρ_C)–computable (even (ν_{bin}, ν_Q)–computable) and increasing. Since its limit x_A is not computable, it cannot have a computable modulus of convergence by Theorem 3.3. The idea is from E. Specker [Spe 49].

The set of computable real numbers is a denumerable subset of $\mathbb{R}$, however it cannot be enumerated "effectively". We prove a positive version of this statement: For every computable enumeration of computable real numbers a computable number which is not enumerated can be determined.

Theorem 3.4

Let $(x_i)_{i\in\omega}$ be a (ν_{bin}, ρ_C)–computable sequence. Then a computable number x with $x \mathrel{\#} x_i$ for all $i \in \omega$ can be determined.

Proof

By diagonalization we construct a computable number x such that $x \mathrel{\#} x_i$ for all $i \in \omega$. For any $i \in \omega$ we can determine a sequence $q_i := u_{i0}\sharp u_{i1}\sharp \ldots$ with $x_i = \rho_C(q_i)$, therefore, there is a computable function $g :\subseteq \Sigma^* \longrightarrow \Sigma^*$ with $g(0^i) = u_{i,2i+2}$. We obtain $|\nu_Q g(0^i) - x_i| < \frac{1}{2} \cdot 3^{-i}$ for all $i \in \omega$. We compute $u_i, v_i \in dom(\nu_Q)$ for $i = 0, 1, \ldots$ as follows:

$$\overline{u}_0 := \nu_Q g(0^i) + 1, \;\; \overline{v}_0 := \overline{u}_0 + 1.$$

Assume u_{i-1} and v_{i-1} have been determined. Define

$$\begin{array}{ll} \overline{u}_i := \overline{u}_{i-1}, \overline{v}_i := \overline{u}_i + 3^{-i} & \text{if } \nu_Q g(0^i) \geq \overline{u}_{i-1} + \frac{3}{2} \cdot 3^{-i}, \\ \overline{u}_i := \overline{u}_{i-1} + 2 \cdot 3^{-i}, \overline{v}_i := \overline{v}_{i-1} & \text{otherwise.} \end{array}$$

The construction guarentees: $\overline{v}_i = \overline{u}_i + 3^{-i}$, $x_i \notin [\overline{u}_i; \overline{v}_i]$ and $[\overline{u}_{i+1}; \overline{v}_{i+1}] \subseteq [\overline{u}_i; \overline{v}_i]$. Therefore $x := \rho_I(u_0\sharp v_0\sharp u_1\sharp v_1\sharp \ldots)$ exists and $x \mathrel{\#} x_i$ for all $i \in \omega$. Additionally, the sequence $u_0\sharp v_0\sharp u_1\sharp v_1\sharp \ldots \in \Sigma^\omega$ is computable, hence x is ρ_I–computable, i.e. ρ_C–computable.
□

Next, we characterize the ρ_C–open, the r.e. and the recursive subsets of $\mathbb{R}$.

Theorem 3.5

For any $X \subseteq \mathbb{R}$

$$\begin{array}{llcl} (1) & X \text{ is } \rho_C\text{–open} & \Longleftrightarrow & X \text{ is open,} \\ (2) & X \text{ is } \rho_C\text{–r.e.} & \Longleftrightarrow & (\exists Y \subseteq dom(\nu_Q) \times dom(\nu_Q),\, Y \text{ r.e.}) \\ & & & X = \bigcup\{(\overline{u}; \overline{v}) \mid (u, v) \in Y\}, \\ (3) & X \text{ is } \rho_C\text{–recursive} & \Longleftrightarrow & X = \emptyset \text{ or } X = \mathbb{R}. \end{array}$$

Proof

(1) Let X be ρ_C–open. Consider $x \in X$. There are words $u_i \in dom(\nu_Q)$ with $\overline{u}_i - 2^{-i} < \overline{u}_{i+1} - 2^{-i-1} < x < \overline{u}_{i+1} + 2^{-i-1} < \overline{u}_i + 2^{-i}$ $(i \in \omega)$. We obtain $\rho_C(u_0\sharp u_1\sharp\ldots) = x$. Since $\rho_C^{-1}(X)$ is open in $dom(\rho_C)$, there is some k with $\rho_C(u_0\sharp\ldots\sharp u_k\sharp\Sigma^\omega) \subseteq X$. Since $x \in (\overline{u}_k - 2^{-k}; \overline{u}_k + 2^{-k}) \subseteq \rho_C(u_0\sharp\ldots\sharp u_k\sharp\Sigma^\omega)$, x has an open neighbourhood in X. Therefore, X is open. On the other hand, let X be open. Consider $p = u_0\sharp u_1\sharp\ldots \in \rho_C^{-1}(X)$. Since X is open, there is some $i \in \omega$ such that $[\rho_C(p) - 2\cdot 2^{-i}; \rho_C(p) + 2\cdot 2^{-i}] \subseteq X$. For any $q \in V := u_0\sharp u_1\sharp\ldots\sharp u_i\sharp\Sigma^\omega \cap dom(\rho_C)$ we have $|\overline{u}_i - \rho_C(q)| \leq 2^{-i}$, therefore $|\rho_C(q) - \rho_C(p)| \leq 2\cdot 2^{-i}$, hence $\rho_C(q) \in X$. Therefore V is an open neighbourhood of p in $\rho_C^{-1}(X)$. This shows that X is ρ_C–open.

(2) Let X be ρ_C–r.e. Then there is some r.e. set $W \subseteq \Sigma^*$ with $\rho_C^{-1}(X) = W\Sigma^\omega \cap dom(\rho_C)$. Let M be a Type 2 machine which for input $(u, v) \in \Sigma^* \times \Sigma^*$ works as follows: M searches systematically for some $w \in W$ and words $u_0, u_1, \ldots, u_k \in dom(\nu_Q)$ such that (cf. (1) of this proof):

$$\overline{u}_0 - 1 < \overline{u}_1 - 2^{-1} < \ldots < \overline{u}_k - 2^{-k} < \overline{u}_k + 2^{-k} < \ldots < \overline{u}_1 + 2^{-1} < \overline{u}_0 + 1,$$
w is a prefix of $u_0\sharp u_1\sharp\ldots\sharp u_k\sharp$,
$$\overline{u} = \overline{u}_k - 2^{-k},\ \overline{v} = \overline{u}_k + 2^{-k}.$$

M halts as soon as such words have been found. By the proof of (1) above, $Y := dom(f_M)$ has the desired properties. On the other hand, let $X = \bigcup\{(\overline{u}; \overline{v}) \mid (u, v) \in Y\}$ with r.e. Y. Let M be a Type 2 machine which for input $p = u_0\sharp u_1\sharp\ldots \in dom(\rho_C)$ works as follows: M searches systematically for some $k \in \omega$ and some $(u, v) \in Y$ with $[u_k - 2^{-k}; u_k + 2^{-k}] \subseteq (\overline{u}; \overline{v})$. M halts, iff the search has been successful. Obviously, $dom(f_M) \cap dom(\rho_C) = \{p \mid \rho_C(p) \in X\}$. Therefore X is ρ_C–r.e..

(3) If X is ρ_C–recursive, X is ρ_C–r.e. and $\mathrm{IR} \setminus X$ is ρ_C–r.e.. Therefore X and $\mathrm{IR} \setminus X$ are ρ_C–open and open by (1). $\emptyset$ and IR are the only sets X with this property, since the real line is connected.

□

By Theorem 3.5(1), the final topology of ρ_C is the usual topology τ_{IR} on IR generated by the open intervals. By (3) IR has no non–trivial ρ_C–recursive subsets. No non–trivial property of real numbers can be decided if only ρ_C–names are available. The characterizations hold accordingly for $X \subseteq \mathrm{IR}^n$ $(n \geq 2)$.

Lemma 3.6 (*some r.e. subsets of* $\mathrm{IR}, \mathrm{IR}^2$)

Let $a \in \mathrm{IR}$ be computable. The sets

$$\{x \in \mathrm{IR} \mid x > a\}, \{x \in \mathrm{IR} \mid x < a\}, \{x \in \mathrm{IR} \mid x \neq a\},$$
$$\{(x, y) \in \mathrm{IR}^2 \mid x < y\}, \{(x, y) \in \mathrm{IR}^2 \mid x \neq y\}$$

are recursively enumerable.

Proof

We consider only the most general case $x < y$. Let M be a Type 2 machine which for input $(p, q) = (u_0\sharp u_1\sharp \ldots, v_0\sharp v_1\sharp \ldots) \in dom(\rho_C) \times dom(\rho_C)$ searches for some $k \in \omega$ with $u_k + 2^{-k} < v_k - 2^{-k}$ and halts as soon as such a k has been found. Then $f_M(p, q)$ exists, iff $\rho_C(p) < \rho_C(q)$. The other proofs are left to the reader.
□

The complements of the above sets $\{x \mid x \leq a\}$ etc. are not r.e., since they are not open (Theorem 3.5.(1)). The r.e. subsets of $\mathbb{R}^n$ are closed under finite union and intersection. By Lemma 3.6, open intervals with computable boundaries are r.e.. Let $A \subseteq \omega$ be r.e. and not recursive. Then the interval $(0; x_A)$, where $x_A = \Sigma\{2^{-i} \mid i \in A\}$, is r.e. (for a proof use Theorem 3.5(2)), but by Example 1 its upper boundary is not computable. Many of the functions studied in "classical" Analysis are computable.

Theorem 3.7 (*some computable real functions*)

(1) The real functions $(x, y) \mapsto x + y$, $(x, y) \mapsto x \cdot y$, $(x, y) \mapsto \max(x, y)$ and $x \mapsto 1/x$ are computable.

(2) Let $(a_i)_{i\in\omega}$ be a (ν_{bin}, ρ_C)–computable sequence and let $R_0 > 0$ be the radius of convergence of the power series $\Sigma a_i x^i$. For each R with $0 < R < R_0$ the real function f_R defined by $f_R(x) = (\Sigma a_i x^i$ if $|x| \leq R$, *div* otherwise) is computable.

Proof

(1) We use the fact that the given functions are continuous and that their restrictions to $\mathbb{Q}$ (which is dense in $\mathbb{R}$) are (ν_Q, ν_Q)–computable.

$x + y$:

Let M be a Type 2 machine which for input (p, q), $p, q \in dom(\rho_C)$, $p = u_0\sharp u_1\sharp \ldots$, $q = v_0\sharp v_1\sharp \ldots$, writes the sequence $r := y_0\sharp y_1\sharp \ldots$ on the output tape, such that

$$\overline{y}_n = \overline{u}_{n+1} + \overline{v}_{n+1}$$

for all $n \in \omega$. Let $x = \rho_C(p)$ and $y = \rho_C(q)$. For all $n > k$ we have

$$|\overline{y}_n - \overline{y}_k| \leq |\overline{u}_{n+1} - \overline{u}_{k+1}| + |\overline{v}_{n+1} - \overline{v}_{k+1}| < 2 \cdot 2^{-k-1} = 2^{-k},$$
$$|\overline{y}_n - (x + y)| \leq |\overline{u}_{n+1} - x| + |\overline{v}_{n+1} - y| \leq 2 \cdot 2^{-k-1} = 2^{-k}.$$

We obtain $r = f_M(p, q) \in dom(\rho_C)$ and $\rho_C(r) = x + y$.

$x \cdot y$:
Let M be a Type 2 machine which for input (p,q), $p,q \in dom(\rho_C)$, $p = u_0\sharp u_1\sharp\ldots$, $q = v_0\sharp v_1\sharp\ldots$, writes the sequence $r := y_0\sharp y_1\sharp\ldots$ on the output tape, such that

$$\overline{y}_n = \overline{u}_{m+n} \cdot \overline{v}_{m+n}$$

for all $n \in \omega$ where m is the smallest natural number with

$$|\overline{u}_0| + 1 \leq 2^{m-1} \text{ and } |\overline{v}_0| + 1 \leq 2^{m-1}.$$

Let $x := \rho_C(p)$ and $y := \rho_C(q)$. For all $n \in \omega$ we have

$$|\overline{u}_n| \leq |\overline{u}_n - \overline{u}_0| + |\overline{u}_0| \leq 2^{m-1}$$

and correspondingly $|\overline{v}_n| \leq 2^{m-1}$. For all $k > n$ we have

$$\begin{aligned}|\overline{y}_n - \overline{y}_k| &\leq |\overline{u}_{m+n} \cdot \overline{v}_{m+n} - \overline{u}_{m+k} \cdot \overline{v}_{m+k}| \\ &\leq |\overline{u}_{m+n}(\overline{v}_{m+n} - \overline{v}_{m+k})| + |\overline{v}_{m+k}(\overline{u}_{m+n} - \overline{u}_{m+k})| \\ &< 2 \cdot 2^{m-1} \cdot 2^{-m-n} = 2^{-n}\end{aligned}$$

and correspondingly $|\overline{y}_n - x \cdot y| \leq 2^{-n}$. We obtain $r = f_M(p,q) \in dom(\rho_C)$ and $\rho_C(r) = x \cdot y$.

$\max(x,y)$:
Let M be a Type 2 machine which for input (p,q), $p,q \in dom(\rho_C)$, $p = u_0\sharp u_1\sharp\ldots$, $q = v_0\sharp v_1\sharp\ldots$, writes the sequence $r := y_0\sharp y_1\sharp\ldots$ on the output tape, such that

$$\overline{y}_n = \max(\overline{u}_n, \overline{v}_n)$$

for all $n \in \omega$. Let $x = \rho_C(p)$ and $y = \rho_C(q)$. Assume $k > n$ and $s := \max(\overline{u}_n, \overline{v}_n, \overline{u}_k, \overline{v}_k)$. If $s = \overline{u}_n$, then

$$|\overline{y}_n - \overline{y}_k| = \overline{u}_n - \max(\overline{u}_k, \overline{v}_k) \leq \overline{u}_n - \overline{u}_k < 2^{-k}.$$

By symmetry, for the other cases $s \in \{\overline{v}_n, \overline{u}_k, \overline{v}_k\}$ we obtain $|\overline{y}_n - \overline{y}_k| < 2^{-k}$ in the same way. Correspondingly $|\overline{y}_n - \max(x,y)| \leq 2^{-k}$ is proved. Therefore $r = f_M(p,q) \in dom(\rho_C)$ and $\rho(r) = \max(x,y)$.

$1/x$:
Let M be a Type 2 machine which for input $p = u_0\sharp u_1\sharp\ldots \in dom(\rho_C)$ and $\rho_C(p) = x \neq 0$ works as follows. First, M searches for the first $N \in \omega$ with $|\overline{u}_N| > 2 \cdot 2^{-N}$. As soon as such a number N has been found, M writes $v_0\sharp v_1\sharp\ldots$ on its output tape, where $\overline{v}_k = 1/\overline{u}_{2N+k}$ for all $k \in \omega$. Since $|u_i| > 2^{-N}$ for all $i \geq N$, v_k exists for all $k \in \omega$. For all k, n with $k < n$ we obtain

$$\begin{aligned}|\overline{v}_k - \overline{v}_n| &= |1/\overline{u}_{2N+k} - 1/\overline{u}_{2N+n}| \\ &= |\overline{u}_{2N+n} - \overline{u}_{2N+k}|/|\overline{u}_{2N+n}||\overline{u}_{2N+k}| \\ &< 2^{-2N-k} \cdot 2^N \cdot 2^N \leq 2^{-k}\end{aligned}$$

and correspondingly $|\overline{v}_k - 1/x| \leq 2^{-k}$. Therefore $r = f_M(p) \in dom(\rho_C)$ and $\rho_C(r) = 1/x$.

(2) It suffices to prove the theorem for rational numbers R. Let $R_1 \in \mathbb{Q}$ be some rational number with $R < R_1 < R_0$. By *Cauchy's estimate*, there is some number $M \in \omega$ with

$$|a_i| < M \cdot R_1^{-i}$$

for all $i \in \omega$. Given some x with $|x| \leq R$, for each n we shall approximate $f_R(x) = \sum_{i=0}^{\infty} a_i x^i$ by $d_n := \sum_{i=0}^{N} c_i b^i$, where N is sufficiently large and $b, c_0, \ldots, c_N$ are rational numbers where $|b - x|$ and the $|a_i - c_i|$ are sufficiently small such that $|f(x) - d_n| < 2^{-n-1}$. Let M be a Type 2 machine which for any input $p = u_0 \sharp u_1 \sharp \ldots \in dom(\rho_C)$ with $|x| \leq R$ (where $x = \rho_C(p)$) generates a sequence $q = v_0 \sharp v_1 \sharp \ldots$ $(v_i \in dom(\nu_Q))$, where v_n is computed as follows:

– M_0 determines some $N \in \omega$ such that
$$M \cdot (R/R_1)^{N+1} \cdot R_1/(R_1 - R) < 2^{-n-3}.$$
– M_0 determines some $b \in \mathbb{Q}$, $|b| \leq R_1$, with
$$|x - b| \cdot \sum_{i=1}^{N} M/R_1 \cdot i < 2^{-n-3}.$$
– For any $i = 0, \ldots, N$ the machine M_0 determines some $c_i \in \mathbb{Q}$ with
$$|a_i - c_i|\,|b|^i < 2^{-n-3}/(N+1).$$
– Define $\overline{v}_n := \sum_{i=0}^{N} c_i \cdot b^i$.

For all $x, b \in \mathbb{R}$ with $|x|, |b| \leq R_1$ and all $i \geq 1$ we have

$$|x^i - b^i| = |x - b| \cdot |x^{i-1} + x^{i-2}b + \ldots + b^{i-1}| \leq |x - b| \cdot i \cdot R_1^{i-1}.$$

We obtain for $|x| \leq R$:

$$\begin{aligned}
&|f_R(x) - \overline{v}_n| \\
&\leq |f_R(x) - \sum_{i=0}^{N} a_i x^i| + |\sum_{i=0}^{N} a_i x^i - \sum_{i=0}^{N} c_i b^i| \\
&\leq |\sum_{i=N+1}^{\infty} a_i x^i| + |\sum_{i=0}^{N} (a_i x^i - c_i b^i)| \\
&\leq \sum_{i=N+1}^{\infty} M \cdot R_1^{-i} \cdot R^i + \sum_{i=0}^{N} |a_i x^i - a_i b^i| + \sum_{i=0}^{N} |a_i b^i - c_i b^i| \\
&\leq M \cdot (R/R_1)^{N+1} \cdot R_1/(R_1 - R) + \sum_{i=1}^{N} |a_i||x - b| \cdot i \cdot R_1^{i-1} + \sum_{i=0}^{N} |a_i - c_i||b|^i \\
&< 2^{-n-3} + |x - b| \sum_{i=0}^{N} i \cdot M/R_1 + 2^{-n-3} \\
&< 3 \cdot 2^{-n-3} \\
&< 2^{-n-1}.
\end{aligned}$$

Consequently, for all $k > n$ we have $|\overline{v}_k - \overline{v}_n| < 2^{-n}$. Therefore $f_R(x) = \rho_C(v_0\sharp v_1\sharp\ldots)$.

□

In general, for computable $(a_i)_{i\in\omega}$ the function $f(x) = \Sigma a_i x^i$ is not computable on $\{x \mid |x| < R_0\}$ where R_0 is the radius of convergence.

Example 3
There is some computable injective function $h : \omega \longrightarrow \omega$ such that $0 \in \mathit{range}(h)$ and $A := \mathit{range}(h)$ is r.e. but not recursive. Define $c_n := 1 + 2^{-h(n)}$ and $a_n := c_n^n$ for all $n \in \omega$. Then the sequence $(a_n)_{n\in\omega}$ is computable, and the power series $\Sigma a_n x^n$ has radius of convergence 1. By Theorem 3.7, $f(x) := \Sigma a_n x^n$ is computable on every interval $[0; r]$ with $0 < r < 1$. We show that f is not computable on $[0; 1)$. If f is computable on $[0; 1)$, there is a computable function $M : \omega \longrightarrow \omega$ with $\Sigma a_n(1 - 2^{-k})^n \leq M(k)$. Define $g : \omega \longrightarrow \omega$ by

$$g(k) := \max\{n \mid h(n) \leq k\}.$$

We obtain for all $k \in \omega$

$$\begin{aligned} M(k+2) &\geq a_{g(k)}(1 - 2^{-k-2})^{g(k)} \\ &= ((1 + 2^{-hg(k)})(1 - 2^{-k-2}))^{g(k)} \\ &\geq ((1 + 2^{-k})(1 - 2^{-k-2}))^{g(k)} \\ &\geq (1 + 2^{-k-1})^{g(k)} \end{aligned}$$

therefore

$$g(k) \leq \log(M(k+2))/\log(1 + 2^{-k-1}).$$

Since M, log and division are computable (see below), $g(k) \leq H(k)$ for some computable function $H : \omega \longrightarrow \omega$. We obtain $k \in A \iff \exists n \leq H(k).h(n) = k$ by the definition of g. Therefore, A must be recursive (contradiction). We conclude that $\Sigma a_n x^n$ is not computable on $[0; 1)$.

Since the power series for e^x, $\sin x$, $\arctan x$, $\ln(1 + x)$ etc. are computable, these functions are computable by Theorem 3.7(2). Since the computable real functions are closed under composition, many other real functions are computable (at least on appropriate subsets of their domains),e.g. $x \mapsto -x$, $(x, y) \mapsto \min(x, y)$, $x \mapsto |x|$, any polynomial function with computable coefficients, $x \mapsto \sqrt{x}$, $x \mapsto \ln x$, $(x, y) \mapsto x^y$, $(x, y) \mapsto -1/(x^2 + y^2)$ etc. . Notice that every restriction of a computable function is computable. Since computable real functions map computable real numbers to computable real numbers, numbers like $e = e^1$, $\pi = 6\cdot$ arc $\sin(\frac{1}{2})$, $-1/\ln(e^2 - \pi)$, $\cos(\sqrt{2} - 1)$, π^2 etc. are computable.

The *join* of two computable real functions at a computable point is computable:

Lemma 3.8 (*join of two functions*)

Let $f_1, f_2 :\subseteq \mathbb{R} \longrightarrow \mathbb{R}$ be computable functions, let $a \in \mathbb{R}$ be computable with $f_1(a) = f_2(a)$. Then $f :\subseteq \mathbb{R} \longrightarrow \mathbb{R}$ defined by

$$f(x) := \begin{cases} f_1(x) & \text{if } x \leq a \\ f_2(x) & \text{otherwise} \end{cases}$$

is computable.

Proof

We only sketch a proof. Consider $i \in \{1, 2\}$. Since f_i is computable, there is a Type 2 machine M_i which computes f_i w.r.t. the Cauchy representation ρ_C. For any input p and any $n \in \omega$ we can compute an interval $I^i_{p,n}$ with rational boundaries such that

$$f_i\rho_C(p) \in I^i_{p,n}$$
$$\lim_{m\to\infty} \mathit{length}\,(I^i_{p,m}) = 0$$

if $p \in \mathit{dom}(f\rho_C)$.
There is some computable sequence $q = t_0\sharp t_1\sharp \ldots$ $(t_i \in \mathit{dom}(\nu_Q))$ with $a = \rho_C(q)$. Let M be a Type 2 machine which for input $p = u_0\sharp u_1\sharp \ldots \in \mathit{dom}(\rho_C)$ produces a sequence $v_0\sharp w_0\sharp v_1\sharp w_1\sharp \ldots$, where the words $v_n, w_n \in \mathit{dom}(\nu_Q)$ are defined as follows:

$$[\overline{v}_n; \overline{w}_n] := \begin{cases} I^1_{p,n} & \text{if } u_n + 2^{-n} < t_n - 2^{-n} \\ I^2_{p,n} & \text{if } t_n + 2^{-n} < u_n - 2^{-n} \\ J_{p,n} & \text{otherwise} \end{cases}$$

where $J_{p,n}$ is the smallest interval containing $I^1_{p,n}$ and $I^2_{p,n}$. For all $p \in \mathit{dom}(f\rho_C)$ we obtain $f\rho_C(p) = \rho_I f_M(p)$, therefore, f is computable.
□

Let us call a function $f :\subseteq \mathbb{R} \longrightarrow \mathbb{R}$ a *polygon*, iff there are real numbers $x_0, y_0, \ldots, x_n, y_n$ with

$$x_0 < x_1 < \ldots < x_n$$

and

$$f(x) := \begin{cases} div & \text{if } x < x_0 \text{ or } x > x_n \\ y & \text{where } (x,y) \text{ is on the straight line connecting} \\ & (x_{i-1}, y_{i-1}) \text{ and } (x_i, y_i), \text{ if } x_{i-1} \leq x \leq x_i. \end{cases}$$

The points $(x_0, y_0), \dots, (x_n, y_n)$ are called vertices of f. As a corollary of lemma 3.8 we obtain that every polygon function with computable vertices is computable.

Computability on the *complex plane* $\mathbb{C}$ is defined by identifying $\mathbb{C}$ with IR^2. For any function $f :\subseteq \mathbb{C} \longrightarrow \mathbb{C}$ there are two functions $f_1, f_2 :\subseteq \mathrm{IR}^2 \longrightarrow \mathrm{IR}$ defined by $f(x+iy) = f_1(x,y) + if_2(x,y)$. The function f is called computable, iff f_1 and f_2 are computable. Computability of complex addition, multiplication, division, $z \mapsto |z|$ and $z \mapsto \arg(z)$ follows from Theorem 3.7(1). The proof of Theorem 3.7(2) can easily be generalized to complex power series. Therefore, also complex functions like $\sin(z)$, e^z, $(w,z) \mapsto w^z$, $\ln(z)$ etc. are computable (on appropriate subsets of their domains).

We conclude with an example of a computable binary relation which has no computable choice function.

Example 4

Let $S := \{(x,n) \in \mathrm{IR} \times \omega \mid |x - n| < 1\}$. Then S as a relation is computable, more precisely (ρ_C, ν_{bin})–computable. But S has no continuous choice function, i.e. there is no (ρ_C, ν_{bin})–continuous function $f : \mathrm{IR} \longrightarrow \omega$ with $(x, f(x)) \in S$ for all $x \in \mathrm{IR}$. We prove both statements.

Let M be a Type 2 machine which for input $p = u_0 \sharp u_1 \sharp \ldots \in dom(\rho_C)$ determines some word w with $|\nu_{bin}(w) - \overline{u}_2| \leq 1/2$. Then $|\rho_C(p) - \nu_{bin} f_M(p)| < 1$ for all $p \in dom(\rho_C)$. Therefore, S is computable. Notice, that we cannot guarantee (ρ_C, ν_{bin})–extensionality of f_M, i.e. we cannot guarantee $\nu_{bin} f_M(p) = \nu_{bin} f_M(p')$ if $\rho_C(p) = \rho_C(p')$.

Assume that there is some (ρ_C, ν_{bin})–continuous function $f : \mathrm{IR} \longrightarrow \omega$ with $|x - f(x)| < 1$ for all $x \in \mathrm{IR}$. Then there is some continuous function $g :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$ with $|\rho_C(p) - \nu_{bin} g(p)| < 1$ for all $p \in dom(\rho_C)$. We have $f(0) = 0$ and $f(1) = 1$. Let $y := \inf\{x \mid f(x) = 1\}$. There is some $p = u_0 \sharp u_1 \sharp \ldots \in dom(\rho_C)$ such that $\rho_C(p) = y$ and $\rho_C(u_0 \sharp \ldots \sharp u_k \sharp \Sigma^\omega)$ is a neighbourhood of y for all $k \in \omega$. Consider the case $f(y) = 0$. Then $g(p) = 0$. By continuity of g there is some k with $g(u_0 \sharp \ldots \sharp u_k \sharp \Sigma^\omega) = \{0\}$. There is some $q \in u_0 \sharp \ldots \sharp u_k \sharp \Sigma^\omega$ with $f \rho_C(q) = 1$. For this q we must have $g(q) = 1$ (contradiction). The case $f(y) = 1$ is treated accordingly.

4 Effective Representation of the Real Numbers

In Section 3 we have introduced ad hoc the Cauchy representation $\rho_C :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ of the real numbers and studied the induced computability on $\mathbb{R}$. Since we are not interested in some arbitrary computability theory on $\mathbb{R}$, we need a good justification for the choice of the Cauchy representation (or some equivalent one).

In this section we explain why the Cauchy representation is topologically natural for the real line, and why it is computationally natural. We mention the concept of admissible representations and formulate the important continuity theorem for admissible representations. Finally we explain why several other representations of $\mathbb{R}$ cannot be natural.

We assume without further discussion (see Appendix C) that our notation $\nu_Q :\subseteq \Sigma^* \longrightarrow \mathbb{Q}$ of the rational numbers induces "the natural" computability on $\mathbb{Q}$. Let K_t be the set of all representations δ of $\mathbb{R}$ such that

$$\{(u, v, p) | \overline{u} < \delta(p) < \overline{v}\} \text{ is open in } \Sigma^* \times \Sigma^* \times dom(\delta).$$

A representation δ is in K_t iff

$$\overline{u} < \delta(p) < \overline{v} \iff \text{already a finite portion of } p \text{ guarantees } \overline{u} < \delta(p) < \overline{v}$$

or, more formally,

$$\overline{u} < \delta(p) < \overline{v} \iff (\exists w)(w \text{ is a prefix of } p \text{ and } \overline{u} < \delta(q) < \overline{v} \text{ for all } q \in w\Sigma^\omega).$$

This means that finite portions of p admit to "locate" $\delta(p)$ arbitrarily precisely by rational numbers from below and above on the real line. Representations not having this property don't seem to be very useful. In fact, the Cauchy representation δ_C, the interval representation δ_I (Def. 3.1) and also the decimal representation δ_{dec} are elements of K_t. Since $\rho_C \not\leq_t \delta_{dec}$, the class K_t does not consist of a single t–equivalence class, but ρ_C is distinguished by maximality in K_t:

Theorem 4.1 (*ρ_C is effective for the real line*)

Let K_t be the set of all functions $\delta :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ such that

$$\{(u, v, p) \mid \overline{u} < \delta(p) < \overline{v}\} \text{ is open in } \Sigma^* \times \Sigma^* \times dom(\delta).$$

Then for any function $\delta :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$

$$\delta \in K_t \iff \delta \text{ is continuous} \iff \delta \leq_t \rho_C.$$

Thus, ρ_C is except for equivalence the unique poorest continuous representation of

IR. If $\rho_C(p) = x$ then all true properties of the form "$\overline{u} < x < \overline{v}$" (and only these) can be obtained from finite portions of any ρ_C–name p of x. There is a surprising formal similarity of Theorem 4.1 to a well known theorem in recursion theory [Wei 87]. Let $\varphi : \omega \longrightarrow P^{(1)}$ be some "effective Gödel numbering" of the set $P^{(1)}$ of the partial recursive functions $f :\subseteq \omega \longrightarrow \omega$. Let K be the set of all numberungs $\psi : \omega \longrightarrow P^{(1)}$ such that $U_\psi := \{(i, x, y) \in \omega^3 \mid \psi_i(x) = y\}$ is r.e.. Then $\psi \in K \iff \psi \leq \varphi$. Notice that U_ψ is r.e., iff ψ satisfies the "universal Turing machine theorem". There is a computational version of Theorem 4.1 expressing that the Cauchy representation is not only topologically but also computationally sound:

Theorem 4.2 (ρ_C *is computationally effective*)

Let K_c be the set of all functions $\delta :\subseteq \Sigma^\omega \longrightarrow$ IR such that

$$\{(u, v, p) \mid \overline{u} < \delta(p) < \overline{v}\} \text{ is r.e. in } \Sigma^* \times \Sigma^* \times dom(\delta).$$

Then for any function $\delta :\subseteq \Sigma^\omega \longrightarrow$ IR

$$\delta \in K_c \iff \delta \leq \rho_C.$$

Thus, ρ_C is also maximal in the subclass $K_c \subseteq K_t$ w.r.t. computable reducibility. Notice that we have a definition of "δ is continuous" but no definition of "δ is computable". As in Theorem 4.1, " $\Longrightarrow$ " corresponds to the "smn–theorem" and " $\Longleftarrow$ " to the "utm–theorem".

Proof

Assume $\delta \in K_c$. By assumption, there is a Type 2 machine M_0 which for any input $(u, v, p) \in \Sigma^* \times \Sigma^* \times dom(\delta)$ halts, iff $\overline{u} < \delta(p) < \overline{v}$. Let M be a Type 2 machine which with input $p \in \Sigma^\omega$ tries to produce a sequence $u_0 \sharp u_1 \sharp \ldots$ ($u_i \in dom(\nu_Q)$) as follows. For computing u_n, by an exhaustive search M tries to find some $(u, v, m) \in \Sigma^* \times \Sigma^* \times \omega$ with $0 < \overline{v} - \overline{u} < 2^{-n}$ such that M_0 with input (u, v, p) halts in at most m steps. If this search is successful, M chooses $u_n := u$. Then $\delta(p) = \rho_C f_M(p)$ for all $p \in dom(\delta)$.

Assume $\delta \leq \rho_C$. By assumption, there is some Type 2 machine M with $\delta(p) = \rho_C f_M(p)$ for all $p \in dom(\delta)$. Let M' be a Type 2 machine, which with input (u, v, p) $(u, v, \in dom(\nu_Q)$, $p \in dom(\delta))$ works as follows. By simulating M with input p M' generates the sequence $f_M(p) = u_0 \sharp u_1 \sharp \ldots$ and halts as soon as some n is found with $\overline{u} < \overline{u}_n - 2^{-n}$ and $\overline{u}_n + 2^{-n} < \overline{v}$. Then

$$\{(u, v, p) \mid \overline{u} < \delta(p) < \overline{v}\} = \Sigma^* \times \Sigma^* \times dom(\delta) \cap dom(f_{M'}),$$

therefore, $\delta \in K_c$.
$\square$

Let $\tau_{\mathbb{R}}$ be the set of open subsets of $\mathbb{R}$. By Theorem 4.1 the representation $\rho_C :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ is *admissible* with final topology $\tau_{\mathbb{R}}$. Appendix D contains a short definition of admissible representations. Here we merely formulate the important continuity theorem. For a broad discussion see [KW 85, Wei 87, Wei 94].

Theorem 4.3 (*continuity*)

For $i = 0, \ldots, k$ let $\delta_i :\subseteq \Sigma^\omega \longrightarrow M_i$ be an admissible representation. For any function $F :\subseteq M_1 \times \ldots \times M_k \longrightarrow M_0$ we have:

$$F \text{ is continuous} \iff F \text{ is } (\delta_1, \ldots, \delta_k, \delta_0)\text{–continuous.}$$

Since every computable function on Σ^ω is continuous and since ρ_C is admissible, by Theorem 4.3., every computable real function is continuous. Because of its importance we prove this fact directly without using Theorem 4.3.

Theorem 4.4

Every computable real function is continuous.

Proof

Let $f :\subseteq \mathbb{R} \longrightarrow \mathbb{R}$ be computable. Then f is (ρ_I, ρ_C)–computable. There is a Type 2 machine M such that $f\rho_I(p) = \rho_C f_M(p)$ for all $p \in dom(f\rho_I)$. Let $O \subseteq \mathbb{R}$ be open and let $f(x) \in O$. We have to show that $f(I) \subseteq O$ for some open intervall I with $x \in I$. There are words $u_i, v_i \in dom(\nu_Q)$ with $\overline{u}_0 < \overline{u}_1 < \ldots$ and $\overline{v}_0 > \overline{v}_1 > \ldots$ such that $x = \rho_I(p)$ where $p = u_0\sharp v_0\sharp u_1\sharp v_1\sharp \ldots$. There are words $w_0, w_1, \ldots \in dom(\nu_Q)$ such that $f_M(p) = q$ where $q = w_0\sharp w_1\sharp \ldots \in dom(\rho_C)$. Since $\rho_C(q) = f(x) \in O$ and O is open, there is some $m \in \omega$ such that $\rho_C(w_0\sharp \ldots \sharp w_m\sharp\Sigma^\omega) \subseteq O$. For producing $w_0\sharp \ldots \sharp w_m\sharp$, the machine M reads at most $u_0\sharp v_0 \ldots \sharp u_k\sharp v_k\sharp$ for some k from the input tape. Let $x' \in (\overline{u}_k; \overline{v}_k)$. Then there is some $q \in \Sigma^\omega$ such that $x' = \rho_I(p')$ where $p' = u_0\sharp v_0 \ldots \sharp u_k\sharp v_k\sharp q$. By the behaviour of M, $f_M(p') \in w_0\sharp \ldots \sharp w_m\sharp\Sigma^\omega$, hence $\rho_C f_M(p') \in O$. We obtain $f(x') = f\rho_I(p') = \rho_C f_M(p') \in O$. Therefore $f(I) \subseteq O$ and $x \in I$ for $I := (\overline{u}_k; \overline{v}_k)$.

The general case $f :\subseteq \mathsf{IR}^n \longrightarrow \mathsf{IR}$ is proved accordingly.
□

At first glance very simple discontinuous real functions like the jump $j : x \mapsto (0$ if $x \leq 0, 1$ otherwise$)$ or the Gauss bracket $g : x \mapsto \lfloor x \rfloor$ (integer part of x) are intuitively computable. Clearly, these two functions are *easily definable* in our mathematical language, but "easily definable" does not mean "computable". This solves the seeming contradiction.

Some functions can be made computable by choosing appropriate representations. Consider a representation δ of the real numbers such that $p(0)$ determines the sign of x if $\delta(p) = x$. Then of course the jump is (δ, ν_{bin})–computable.
Every function $f :\subseteq \mathsf{IR} \longrightarrow \mathsf{IR}$ can be made (δ, ρ_C)–computable for some appropriate representation δ depending on f. Define $\delta(p(0)q(0)p(1)q(1)\ldots) = x :\iff \rho_C(p) = x$ and $\rho_C(q) = f(x)$. This "dirty trick" cannot be applied to two–place functions:

Lemma 4.5

There is no representation $\delta :\subseteq \Sigma^\omega \longrightarrow \mathsf{IR}$ such that the test $l : \mathsf{IR} \times \mathsf{IR} \longrightarrow \omega$, where $l(x, y) = (0$ if $x < y$, 1 otherwise$)$ is $(\delta, \delta, \nu_{bin})$–continuous.

Proof
Assume that there is some continuous function $f :\subseteq \Sigma^\omega \times \Sigma^\omega \longrightarrow \Sigma^*$ such that $l(\delta(p), \delta(q)) = \nu_{bin} f(p, q)$. Consider $z = \delta(p)$. We have $1 = l(\delta(p), \delta(p)) = \nu_{bin} f(p, p)$. Therefore, $f(p, p) = 1 \in \Sigma^*$. Since f is continuous, $f(w\Sigma^\omega, w\Sigma^\omega) = \{1\}$ for some prefix w of p. For any $x, y \in \delta(w\Sigma^\omega)$ we obtain $x \geq y$ hence $\{z\} = \delta(w\Sigma^\omega)$. Therefore, for any $z \in \mathsf{IR}$ there is some $w \in \Sigma^*$ with $\{z\} = \delta(w\Sigma^\omega)$. This, however, is impossible since $card(\Sigma^*) < card(\mathsf{IR})$.
□

We may interpret the result as follows: the function l is absolutely not computable by physical devices.

According to Theorem 4.1, the Cauchy representation is distinguished from other representations of the real numbers (except for topological equivalence), where the topology τ_{IR} on IR by the open intervals with rational boundaries is considered as the reference structure on IR. Theorems 4.2, 4.3 and 4.4 confirm, that the Cauchy representation induces the "natural" computability theory on the real line. Since the decimal representation δ_{dec} is not even t–equivalent to ρ_C, it is "unnatural". Remember also that by Example 2.4 the (computable) real function $x \mapsto 3x$ is not $(\delta_{dec}, \delta_{dec})$–computable. Are there representations in the equivalence class of ρ_C which are simpler than ρ_C? The next theorem excludes some obvious simplifications.

Theorem 4.6 (*restrictions for admissible representations of* $\mathbb{R}$)

(1) No total representation $\delta : \Sigma^\omega \longrightarrow \mathbb{R}$ is t–equivalent to ρ_C.

(2) No injective representation $\delta :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ is t–equivalent to ρ_C.

(3) Define the "naive" Cauchy representation of $\mathbb{R}$ by $\rho_n(p) = x$ iff there are $u_0, u_1, \ldots \in dom(\nu_Q)$ with $p = u_0 \sharp u_1 \sharp \ldots$ and $x = \lim_{i \to \infty} \overline{u}_i$.
Then ρ_n is not t–equivalent to ρ_C.

Proof

(1) One can show that Σ^ω with the Cantor topology is a compact metric space. If $\delta \equiv_t \rho_C$ then δ is continuous (Theorem 4.1). Since any continuous function maps compact sets to compact sets, also $\mathbb{R}$ must be compact, but $\mathbb{R}$ is not bounded.

(2) Assume that there is an injective representation $\delta :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ with $\delta \equiv_t \rho_C$. By Theorem 3.5 $X \subseteq \mathbb{R}$ is open $\iff$ X is ρ_C–open $\iff$ X is δ–open. We conclude that δ^{-1} is a continuous function. Any continuous function maps connected sets to connected sets. The set $\mathbb{R}$ is connected. But $\delta^{-1}\mathbb{R} = dom(\delta)$ is not connected: Let $p, q \in \delta^{-1}\mathbb{R}$, $p \neq q$. Then there is some $w \in \Sigma^*$ with $p \in w\Sigma^*$ and $q \notin w\Sigma^*$. Let $A := w\Sigma^\omega \cap \delta^{-1}\mathbb{R}$, $B := \Sigma^\omega \setminus w\Sigma^\omega \cap \delta^{-1}\mathbb{R}$. Then A and B are both open in $\delta^{-1}\mathbb{R}$ and non–empty, and $\delta^{-1}\mathbb{R} = A \cup B$ and $A \cap B = \emptyset$. Hence $dom(\delta)$ is not connected.

(3) Assume that there is some continuous function f with $\rho_n(p) = \rho_C f(p)$ for all $p \in dom(\rho_n)$. Let $p = (0\sharp)^\omega$. Then $\rho_n(p) = 0$. Let $f(p) = u_0 \sharp u_1 \sharp \ldots$. By continuity of f there is some k with $f((0\sharp)^k \Sigma^\omega) \subseteq u_0 \sharp u_1 \sharp \Sigma^\omega$. Then f is incorrect for $p' := (0\sharp)^k (1\sharp)^\omega$ since $\rho_C f((0\sharp)^k \Sigma^\omega) \subseteq [-1/2; 1/2]$ but $\rho_n(p') = 1$.
□

5 Open and Compact Subsets

Since the cardinality of $2^{\mathbb{R}}$, the power set of $\mathbb{R}$, is greater than the cardinality of Σ^ω, it has no representation. Therefore, in our approach we are not able to investigate computability of functions like $f :\subseteq 2^{\mathbb{R}} \longrightarrow \mathbb{R}$ with $f(X) = y :\iff y = supX$. We restrict our attention to the open subsets and to the compact subsets of $\mathbb{R}$, which have representations. We define a standard representation of $\tau_{\mathbb{R}}$, show that it is topologically and computationally effective and list some computability results. We introduce several effective representations of the set $K(\mathbb{R})$ of the compact subsets of $\mathbb{R}$, prove a computational version of the Heine/Borel theorem and give examples for computable operations on the set $K(\mathbb{R})$ of compact sets.

Definition 5.1 (*representation of* $\tau_{\mathbb{R}}$)

Define a representation δ_{op} of the set $\tau_{\mathbb{R}}$ of open subsets of $\mathbb{R}$ as follows:

$$\begin{aligned}\delta_{op}(p) = X, \quad & \text{iff there are words } u_0, v_0, u_1, v_1, \ldots \in dom(\nu_Q)\\ & \text{with } \overline{u}_i \leq \overline{v}_i \text{ for all } i \in \omega \text{ and } p = u_0\sharp v_0\sharp u_1\sharp v_1\sharp\ldots \text{ such that}\\ & X = \bigcup\{(\overline{u}_i;\overline{v}_i) \mid i \in \omega\}.\end{aligned}$$

for all $p \in \Sigma^\omega$ and $X \in \tau_{\mathbb{R}}$.

We use the convention $(a; a) := \emptyset$. A sequence $p \in \Sigma^\omega$ is a δ_{op}–name of X, iff p enumerates a set of open intervals with rational boundaries which exhausts X. Since every open subset X of $\mathbb{R}$ is the union of a set of open intervals with rational boundaries, the above function δ_{op} is surjective, i.e. it is a representation of $\tau_{\mathbb{R}}$. The equivalence class of δ_{op} can be defined by a simple effectivity property and a maximality principle (cf. Theorems 4.1, 4.2):

Theorem 5.2 (*effectivity of* δ_{op})

(1) Let K_t be the set of all functions $\delta :\subseteq \Sigma^\omega \longrightarrow \tau_{\mathbb{R}}$ such that

$$\{(u, v, p) \mid \overline{u} < \overline{v} \text{ and } [\overline{u}; \overline{v}] \subseteq \delta(p)\} \text{ is open in } \Sigma^* \times \Sigma^* \times dom(\delta).$$

Then for all functions $\delta :\subseteq \Sigma^\omega \longrightarrow \tau_{\mathbb{R}}$

$$\delta \in K_t \iff \delta \leq_t \delta_{op}.$$

(2) Let K_c be the set of all functions $\delta :\subseteq \Sigma^\omega \longrightarrow \tau_{\mathbb{R}}$ such that

$$\{(u, v, p) \mid \overline{u} < \overline{v} \text{ and } [\overline{u}; \overline{v}] \subseteq \delta(p)\} \text{ is r.e. in } \Sigma^* \times \Sigma^* \times dom(\delta).$$

Then for all functions $\delta :\subseteq \Sigma^\omega \longrightarrow \tau_{\mathbb{R}}$

$$\delta \leq K_c \iff \delta \leq \delta_{op}.$$

Thus, δ_{op} is except for equivalence the unique poorest representations δ of $\tau_{\mathbb{R}}$, for which every true property of the form "$[\overline{u};\overline{v}] \subseteq X$" can be obtained from a finite portion of any δ–name of X. We omit a proof of Theorem 5.2. A few examples for induced effectivity are listed in the following theorem.

Theorem 5.3 (*properties of* δ_{op})

(1) X is δ_{op}–computable $\iff$ X is ρ_C–r.e.

(2) $\{(x, X) \in \mathbb{R} \times \tau_{\mathbb{R}} \mid x \in X\}$ is (ρ_C, δ_{op})–r.e.

(3) Union and intersection on $\tau_{\mathbb{R}}$ are $(\delta_{op}, \delta_{op}, \delta_{op})$–computable.

(4) For $f : \mathbb{R} \longrightarrow \mathbb{R}$ define $H_f : \tau_{\mathbb{R}} \longrightarrow \tau_{\mathbb{R}}$ by $H_f(X) := f^{-1}(X)$ for all $X \in \tau_{\mathbb{R}}$. Then
- H_f is $(\delta_{op}, \delta_{op})$–continuous, if f is continuous,
- H_f is $(\delta_{op}, \delta_{op})$–computable, if f is computable.

Proof

(1) This is an immediate consequence of Theorem 3.5(2).

(2) Let M be a Type 2 machine which for inputs $p = w_0\sharp w_1\sharp \ldots \in dom(\rho_C)$ and $q = u_0\sharp v_0\sharp u_1\sharp v_1\sharp \ldots \in dom(\delta_{op})$ works as follows. M searches systematically for indices i, k with $\overline{u}_k < \overline{w}_i - 2^{-i}$ and $\overline{w}_i + 2^{-i} < \overline{v}_k$. M halts, iff such indices have been found. We obtain $\{(p, q) \mid \rho_C(p) \in \delta_{op}(q)\} = dom(f_M) \cap dom(\rho_C) \times dom(\delta_{op})$ (see Def. 2.9(2)).

(3) Consider only inputs of the form $p = u_0\sharp v_0\sharp u_1\sharp v_1\sharp \ldots \in dom(\delta_{op})$ and $q = w_0\sharp x_0\sharp w_1\sharp x_1\sharp \ldots \in dom(\delta_{op})$. For the case of union let M be a Type 2 machine which produces from p and q the output $u_0\sharp v_0\sharp w_0\sharp x_0\sharp u_1\sharp v_1\sharp w_1\sharp x_1\sharp \ldots$. For the case of intersection let M be a Type 2 machine, which produces a list of all intervals $(\overline{u};\overline{v})$ for which there are numbers i, k with $(\overline{u};\overline{v}) = (\overline{u}_i;\overline{v}_i) \cap (\overline{w}_k;\overline{x}_k)$.

(4) This follows from the more general Theorem 6.3 below.

□

A subset $X \subseteq \mathbb{R}$ is *compact*, iff X is closed and bounded. By the *Heine/Borel theorem*, X is compact, iff for every set $\beta \subseteq \tau_{\mathbb{R}}$ of open subsets of $\mathbb{R}$ with $X \subseteq \cup\beta$ there is some finite subset $\beta' \subseteq \beta$ with $X \subseteq \cup\beta'$. The characterization remains valid, if above $\tau_{\mathbb{R}}$ is replaced by the set of all open intervals with rational boundaries. The following four representations can be derived from these characterizations.

Definition 5.4 (*representations of the compact sets*)

Let $K(\mathrm{IR})$ be the set of all compact subsets of IR. Define a notation ι of the finite sets of open intervals with rational boundaries by

$$\begin{aligned}\iota(w) = \beta, \text{ iff } & \text{ there are words } u_1, v_1, \ldots, u_k, v_k \in dom(\nu_Q) \text{ with} \\ & w = u_1 \sharp v_1 \sharp \ldots \sharp u_k \sharp v_k \text{ and } \beta = \{(\overline{u}_1; \overline{v}_1), \ldots, (\overline{u}_k; \overline{v}_k)\}.\end{aligned}$$

Define representations κ_c, κ_{cb}, κ_w and κ of $K(\mathrm{IR})$ as follows:

(1) ("closed representation")

$$\kappa_c(p) = X, \text{ iff } \delta_{op}(p) = \mathrm{IR} \setminus X.$$

(2) ("closed bounded representation")

$$\begin{aligned}\kappa_{cb}(p) = X, \text{ iff } & \text{ there are } u \in dom(\nu_{bin}) \text{ and } q \in dom(\delta_{op}) \text{ with} \\ & p = u \sharp q, X = \mathrm{IR} \setminus \delta_{op}(q) \text{ and } X \subseteq [-\overline{u}; \overline{u}].\end{aligned}$$

(3) ("weak covering representation")

$$\begin{aligned}\kappa_w(p) = X, \text{ iff } & \text{ there are words } w_0, w_1, \ldots \in dom(\iota) \text{ with} \\ & p = w_0 ¢ w_1 ¢ \ldots \text{ such that for all } w \in dom(\iota) : \\ & X \subseteq \bigcup \iota(w) \iff (\exists i) w = w_i\end{aligned}$$

(4) ("strong covering representation")

$$\begin{aligned}\kappa(p) = X, \text{ iff } & p = w_0 ¢ w_1 ¢ \ldots \text{ as above such that} \\ & \{w_0, w_1, \ldots\} = \{w \mid X \subseteq \bigcup \iota(w) \text{ and } \forall I \in \iota(w). I \cap X \neq \emptyset\}\end{aligned}$$

If $\kappa_c(p) = X$, then p enumerates the complement of X. If $\kappa_{cb}(p) = X$, then p gives a bound of X and enumerates the complement of X. If $\kappa_w(p) = X$, then p enumerates all coverings of X with finitely many open intervals with rational boundaries. In the case of κ instead of κ_w, only the "minimal" coverings are enumerated by names.

The reducibilities between the four above representations are given by the following theorem.

Theorem 5.5 (*computational Heine/Borel theorem*)

(1) $\kappa_{cb} \leq \kappa_c$, $\kappa_c \not\leq_t \kappa_{cb}$

(2) $\kappa_w \equiv \kappa_{cb}$ (computational Heine/Borel theorem)

(3) $\kappa \leq \kappa_w$, $\kappa_w \not\leq_t \kappa$

Proof

(1) There is a Type 2 machine M with $f_M(u\sharp u_0\sharp u_1\sharp\ldots) = u_0\sharp u_1\sharp\ldots$ for $u, u_0, u_1, \ldots \in dom(\nu_Q)$. Then f_M translates κ_{cb} to κ_c. If $\kappa_c(p) = X$, then no finite prefix w of p contains any information about a bound of X, hence $\kappa_c \not\leq_t \kappa_{cb}$. More formally, assume that there is some continuous function $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ with $\kappa_c(p) = \kappa_{cb}f(p)$ for all $p \in dom(\kappa_c)$. There is some $p = u_0\sharp v_0\sharp u_1\sharp v_1\sharp\ldots$ with $\kappa_c(p) = \mathrm{IR} \setminus \delta_{op}(p) = \{0\}$. The sequence $f(p)$ has the form $u\sharp u_0'\sharp v_0'\sharp\ldots$. By continuity of f there is some prefix w of p with $f(w\Sigma^\omega) \subseteq u\sharp\Sigma^\omega$. But there is some $q \in w\Sigma^\omega \cap dom(\kappa_c)$ with $\kappa_c(q) \not\in [-\overline{u}; \overline{u}]$ (contradiction).

(2) We show $\kappa_w \leq \kappa_{cb}$. Assume $\kappa_w(p) = X$. Then p enumerates all coverings of X with finitely many intervals with rational boundaries. From the first such covering a bound for X can be determined easily. From the other coverings one can determine an enumeration of open intervals with rational boundaries which exhausts $\mathrm{IR} \setminus X$. We prove this more formally. Let $\nu_\Sigma : \omega \longrightarrow \Sigma^*$ be some standard bijective numbering of Σ^*. There is a Type 2 machine which for input $p = w_0¢w_1¢\ldots \in dom(\kappa_w)$ produces a sequence $u\sharp u_0\sharp v_0\sharp u_1\sharp v_1\sharp\ldots \in dom(\kappa_{cb})$ with the following properties:

$$\bigcup\iota(w_0) \subseteq [-\overline{u}; \overline{u}]$$

$$(u_i, v_i) = \begin{cases} (u, v) & \text{if } \nu_\Sigma(i) = 0^k\sharp u\sharp v \text{ with} \\ & (\overline{u}; \overline{v}) \cap \bigcup\iota(w_k) = \emptyset. \\ (0, 0) & \text{otherwise.} \end{cases}$$

Then $\kappa_w(p) = \kappa_{cb}f_M(p)$ for all $p \in dom(\kappa_w)$.
We show $\kappa_{cb} \leq \kappa_w$. Assume $\kappa_{cb}(p) = X$. Then from p we know some closed interval I with $X \subseteq I$ and an enumeration $I_0, I_1, \ldots$ of open intervals exhausting $\mathrm{IR} \setminus X$. Since I is compact,

$$X \subseteq \bigcup\iota(w) \text{ iff} I \subseteq \bigcup\iota(w) \cup I_0 \cup \ldots \cup I_k \text{ for some } k \in \omega.$$

Therefore, we can enumerate all words w with $X \subseteq \bigcup\iota(w)$. We prove this more formally. There is a Type 2 machine which for input $p := u\sharp u_0\sharp v_0\sharp u_1\sharp v_1\sharp\ldots \in dom(\rho_{cb})$ produces a sequence $q := w_0¢w_1¢\ldots \in dom(\kappa_w)$ with the following properties:

$$w_i := \begin{cases} w & \text{if } \nu_\Sigma(i) = 0^k¢w \text{ with} \\ & [-\overline{u}; \overline{v}] \subseteq \bigcup\iota(w) \cup (\overline{u}_0; \overline{v}_0) \cup \ldots \cup (\overline{u}_k; \overline{v}_k) \\ w' & \text{otherwise.} \end{cases}$$

where $[-\overline{u}; \overline{u}] \subseteq \bigcup\iota(w')$. Then $\kappa_{cb}(p) = \kappa_w f_M(p)$ for all $p \in dom(\kappa_{cb})$.

(3) Since κ is a restriction of κ_w, $\kappa \leq \kappa_w$ is trivial. $\kappa_w \not\leq_t \kappa$ follows from Theorem 5.6(1) below.

□

The equivalence $\kappa_w \equiv \kappa_{cb}$ can be considered as a computational version of the

Heine/Borel theorem. There is an "improvement" κ' of κ_{cb}, for which $\kappa' \equiv \kappa$, a stronger computational version of the Heine/Borel theorem, can be proved (see [KW 87]).

Every compact subset of IR has a maximum and a minimum, the compact sets are closed under union and intersection, and $f(X)$ is compact if f is continuous and X is compact. Effective versions of these facts are listed in the following theorem.

Theorem 5.6 (*computable operations on compact sets*)

(1) The function $\max : K(\mathsf{IR}) \longrightarrow \mathsf{IR}$ is (κ, ρ_C)–computable but not (κ_w, ρ_C)–continuous.

(2) Intersection and union are $(\kappa_w, \kappa_w, \kappa_w)$–computable and (κ, κ, κ)–computable.

(3) For $f : \mathsf{IR} \longrightarrow \mathsf{IR}$ define $H_f : K(\mathsf{IR}) \longrightarrow K(\mathsf{IR})$ by $H_f(X) := f(X)$ for all $X \in K(\mathsf{IR})$. Then

- H_f is (κ_w, κ_w)–continuous and (κ, κ)–continuous, if f is continuous,
- H_f is (κ_w, κ_w)–computable and (κ, κ)–computable, if f is computable.

Proof

(1) There is a Type 2 machine M which transforms any $p := w_0 ¢ w_1 ¢ \ldots \in dom(\kappa)$ into $q := u_0 \sharp v_0 \sharp u_1 \sharp v_1 \sharp \ldots \in dom(\rho_I)$, where the u_i, v_i are defined as follows:

$(\overline{u}_i; \overline{v}_i)$ is the greatest interval in $\iota(w_i)$

w.r.t. the order $(u, v) \leq (u', v') \iff (v < v'$ or $(v = v'$ and $u \leq u'))$. Then $\max \kappa(p) = \rho_I f_M(p)$. Assume, there is a continuous function $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ with $\max \kappa_w(p) = \rho_C f(p)$ for all $p \in dom(\kappa_w)$. There is some $p = x_0 ¢ x_1 ¢ \ldots$ with $\kappa_w(p) = [0; 1]$. Let $f(p) = u_0 \sharp u_1 \sharp \ldots$. Then $\rho_C f(p) = 1$ and $\overline{u}_2 \geq 3/4$. Since f is continuous, there is some $i \in \omega$ with $f(x_0 ¢ x_1 ¢ \ldots ¢ x_i ¢ \Sigma^\omega) \subseteq u_0 \sharp u_1 \sharp u_2 \sharp \Sigma^\omega$. There is some $q \in x_0 ¢ x_1 ¢ \ldots ¢ x_i ¢ \Sigma^\omega)$ with $\kappa_w(q) = \{0\}$. We obtain $\max \kappa_w(q) = 0$ but $\rho_C f(q) \geq 1/2$ (contradiction).

(2) The proof is left to the reader.

(3) This follows from the more general Theorem 6.3 below.

□

The definitions and theorems of this section can be generalized easily from IR to the n–dimensional Euklidean space IR^n. There are theorems similar to Theorem

5.2 characterizing the "effectivity" of κ_w and of κ. We don't go into more details here. We mention without proof, that κ is admissible and that the final topology $\tau_\kappa = \{X \subseteq K(\mathbb{R}) \mid \kappa^{-1}X$ is open in $dom(\kappa)\}$ of the representation κ is the Hausdorff topology on the set $K(\mathbb{R})$ of the compact subsets of $\mathbb{R}$ [Eng 89, Wei 94]. Especially Theorem 4.3 is applicable to κ.

6 Representations of Continuous Real Functions

Let us denote by $C(X)$ the set $\{f :\subseteq \mathsf{IR} \longrightarrow \mathsf{IR} \mid f$ continuous and $dom(f) = X\}$. We introduce explicitly standard represenations $\delta_{\mathbb{R}}$ of $C(\mathsf{IR})$ and δ_C of $C[0;1]$ and give sufficient reasons for their effectivity. As examples we consider modulus of continuity, maximum, differentiation and integration.

Definition 6.1 (*representation of* $C(\mathsf{IR})$)

Define a representation $\delta_{\mathbb{R}} :\subseteq \Sigma^\omega \longrightarrow C(\mathsf{IR})$ as follows.

$$\begin{aligned} \delta_{\mathbb{R}}(p) = f, \quad & \text{iff there are words } u_i, v_i, x_i, y_i \in dom(\nu_Q) \ \ (i \in \omega) \\ & \text{with } p = u_0 \sharp v_0 \sharp x_0 \sharp y_0 \cent u_1 \sharp v_1 \sharp x_1 \sharp y_1 \cent \ldots \\ & \text{such that for all rational numbers } a, b, c, d : \\ & f[a;b] \subseteq (c;d) \iff (\exists i)(a = \overline{u}_i, b = \overline{v}_i, c = \overline{x}_i, y = \overline{y}_i) \end{aligned}$$

for all $p \in \Sigma^\omega$ and $f \in C(\mathsf{IR})$.

Roughly speaking, $\delta_{\mathbb{R}}(p) = f$ iff p enumerates all $(a, b, c, d) \in \mathbb{Q}^4$ with $f[a;b] \subseteq (c;d)$. This representation has the following remarkable effectivity property (cf. Thms. 4.1, 4.2, 5.2).

Theorem 6.2 ($\delta_{\mathbb{R}}$ *is effective*)

(1) Let L_t be the set of all functions $\delta :\subseteq \Sigma^\omega \longrightarrow C(\mathsf{IR})$ such that the function *apply* : $C(\mathsf{IR}) \times \mathsf{IR} \longrightarrow \mathsf{IR}$, where $apply(f, x) := f(x)$, is (δ, ρ_C, ρ_C)–continuous. Then

$$\delta \in L_t \iff \delta \leq_t \delta_{\mathbb{R}}$$

for all functions $\delta :\subseteq \Sigma^\omega \longrightarrow C(\mathsf{IR})$.

(2) Let L_c be the set of all functions $\delta :\subseteq \Sigma^\omega \longrightarrow C(\mathsf{IR})$ such that *apply* is (δ, ρ_C, ρ_C)–computable. Then

$$\delta \in L_c \iff \delta \leq \delta_{\mathbb{R}}$$

for all functions $\delta :\subseteq \Sigma^\omega \longrightarrow C(\mathsf{IR})$.

Again there is a formal similarity with the characterization of "effective Gödel numberings" ψ of $P^{(1)}$:

ψ satisfies the universal Turing machine theorem $\iff \psi \leq \varphi$

(see the remarks after Theorem 4.1). We omit a proof of Theorem 6.2. Especially, we have $\delta_{\mathbb{R}} \in L_c$, i.e. the "universal function" *apply* $:\subseteq C(\mathrm{IR}) \times \mathrm{IR} \longrightarrow \mathrm{IR}$ of $\delta_{\mathbb{R}}$ is $(\delta_{\mathbb{R}}, \rho_C, \rho_C)$–computable. Some interesting properties are listed in the following theorem.

Theorem 6.3 (*some computable operations*)

(1) $f : \mathrm{IR} \longrightarrow \mathrm{IR}$ is (ρ_C, ρ_C)–computable $\Longleftrightarrow$ f is $\delta_{\mathbb{R}}$–computable.
(2) The function $H : C(\mathrm{IR}) \times \tau_{\mathbb{R}} \longrightarrow \tau_{\mathbb{R}}$, defined by $H(f, X) := f^{-1}X$, is $(\delta_{\mathbb{R}}, \delta_{op}, \delta_{op})$–computable.
(3) The function $G : C(\mathrm{IR}) \times K(\mathrm{IR}) \longrightarrow K(\mathrm{IR})$, defined by $G(f, X) := f(X)$, is $(\delta_{\mathbb{R}}, \kappa_w, \kappa_w)$–computable. and $(\delta_{\mathbb{R}}, \kappa, \kappa)$–computable.
(4) The composition $F : C(\mathrm{IR}) \times C(\mathrm{IR}) \longrightarrow C(\mathrm{IR})$, defined by $F(f, g) := f \circ g$, is $(\delta_{\mathbb{R}}, \delta_{\mathbb{R}}, \delta_{\mathbb{R}})$–computable.

We do not prove this theorem. It is well–known that continuous functions are uniformly continuous on compact subsets. We shall prove a computable version of this theorem. We call a function $m : \omega \longrightarrow \omega$ a *modulus of continuity* of a function $f :\subseteq \mathrm{IR} \longrightarrow \mathrm{IR}$ on $X \subseteq dom(f)$, iff for all $x, y \in X$ and $n \in \omega$:

$$|x - y| < 2^{-m(n)} \Longrightarrow |f(x) - f(y)| < 2^{-n}.$$

For the set $\omega^\omega = \{m \mid m : \omega \longrightarrow \omega\}$ we use the following standard representation $\delta_\omega :\subseteq \Sigma^\omega \longrightarrow \omega^\omega$:

$$\delta_\omega(p) = m :\Longleftrightarrow p = u_0\sharp u_1\sharp \ldots \text{ with } (\forall i)\nu_{bin}(u_i) = m(i)$$

for all $p \in \Sigma^\omega$ and $m \in \omega^\omega$.

Theorem 6.4 (*determination of a modulus of continuity*)

There is a computable function $h :\subseteq \Sigma^\omega \times \Sigma^* \longrightarrow \Sigma^\omega$ such that $\delta_\omega h(p, z)$ is a modulus of continuity of $\delta_{\mathbb{R}}(p)$ on $[-\overline{z}; \overline{z}]$ for all $p \in dom(\delta_{\mathbb{R}})$ and all $z \in dom(\nu_{bin})$.

Proof

Consider $N \in \omega$. If $f : \mathrm{IR} \longrightarrow \mathrm{IR}$ is continuous, for any $x \in [-N; N]$ and any $n \in \omega$ there are numbers $a_x, b_x, c, d \in \mathbb{Q}$ such that $x \in (a_x; b_x)$ and $f[a_x; b_x] \subseteq (c; d)$ and $d - c < 2^{-n-1}$. Obviously, $[-N; N] \subseteq \cup\{(a_x; b_x) \mid x \in [-N; N]\}$. Since $[-N; N]$ is compact, a finite subset of intervals suffices for covering $[-N; N]$. Therefore, for $n \in \omega$ there is a finite set of quadrupels (a_i, b_i, c_i, d_i) of rational numbers $(i = 1, \ldots, k)$ such that $[-N; N] \subseteq (a_1; b_1) \cup \ldots \cup (a_k; b_k)$ and $f[a_i; b_i] \subseteq (c_i, d_i)$ and $d_i - c_i < 2^{-n-1}$ (for $i = 1, \ldots, k$). Let $c := min\{b_i - a_i \mid i = 1, \ldots, k\}$. Assume

$-N \leq x \leq y \leq N$ and $|x-y| < c$. Then there are i, j with $x \in (a_i; b_i)$, $y \in (a_j; b_j)$ and $(a_i; b_i) \cap (a_j \cap b_j) \neq \emptyset$. Consequently $f(x) \in (c_i; d_i)$, $f(y) \in (c_j; d_j)$ and $(c_i; d_i) \cap (c_j \cap d_j) \neq \emptyset$. Therefore $|f(x) - f(y)| < 2^{-n}$. Let M be a Type 2 machine which for input $p = t_0 ¢ t_1 ¢ \ldots \in dom(\delta_{\mathrm{I\!R}})$ (where $t_i = u_i \sharp v_i \sharp x_i \sharp y_i$) and $z \in dom(\nu_{bin})$ produces a sequence $q = w_0 \sharp w_1 \sharp \ldots$ where w_n is defined as follows. M searches for a finite set $I \subset \omega$ of indices such that $\overline{y}_i - \overline{x}_i < 2^{-n-1}$ for all $i \in I$ and $[-\overline{z}; \overline{z}] \subseteq \cup\{(\overline{u}_i; \overline{v}_i) \mid i \in I\}$. (Such a set I exists.) M determines $m \in \omega$ with $2^{-m} \leq \min\{\overline{v}_i - \overline{u}_i \mid i \in I\}$. Then $w_n \in dom(\nu_{bin})$ is determined by $\nu_{bin}(w_n) = m$. By the above considerations, $\delta_\omega(q)$ is a modulus of continuity of $f = \delta_{\mathrm{I\!R}}(p)$ on $[-\overline{z}; \overline{z}]$.
□

Definition 6.1 and Theorems 6.2, 6.3 and 6.4 can be easily generalized from $C(\mathrm{I\!R})$ to $C(X)$ where $X \subseteq \mathrm{I\!R}$ is r.e.. Also generalizations from $\mathrm{I\!R}$ to $\mathrm{I\!R}^n$ are straightforward. Next we study the class $C[0;1]$ of the continuous functions $f :\subseteq \mathrm{I\!R} \longrightarrow \mathrm{I\!R}$ with $dom(f) = [0;1]$. We introduce a metric on $C[0;1]$ and define, as a generalization of ρ_C a standard Cauchy representation of $C[0;1]$.

For $f, g \in C[0;1]$ define the distance $d(f,g) := \max\{|f(x) - g(x)| | 0 \leq x \leq 1\}$. $(C[0;1], d)$ is a metric space. Let Pg be the set of all polygon functions $f \in C[0;1]$ with rational vertices. It is known that Pg is dense in $(C[0;1], d)$, i.e. for any $f \in C[0;1])$ and $n \in \omega$ there is some $g \in Pg$ with $d(f,g) < 2^{-n}$. The open ball $B(f,a)$ with centre $f \in C[0;1]$ and radius a can be visualized by a stripe of width $2a$ surrounding f.

Definition 6.5 (*Cauchy representation of* $C[0;1]$)

(1) Define a notation $\alpha :\subseteq \Sigma^* \longrightarrow Pg$ of the set Pg of all polygon functions with rational vertices from $C[0;1]$ by:

$$\begin{aligned} \alpha(w) = g, \quad & \text{iff there are } u_0, v_0, \ldots, u_k, v_k \in dom(\nu_Q) \text{ with} \\ & w = u_0 \sharp v_0 \sharp \ldots \sharp u_k \sharp v_k, \\ & 0 = \overline{u}_0 < \ldots < \overline{u}_k = 1 \text{ and } g \text{ is the polygon} \\ & \text{with the vertices } (u_0, v_0), \ldots, (u_k, v_k). \end{aligned}$$

(2) Define a representation $\delta_C :\subseteq \Sigma^\omega \longrightarrow C[0;1]$ of $C[0;1]$ by:

$$\begin{aligned} \delta_C(p) = f, \quad & \text{iff there are } w_0, w_1, \ldots \in dom(\alpha) \text{ with} \\ & p = w_0 ¢ w_1 ¢ \ldots, (\forall k)(\forall i > k) d(\alpha(w_i), \alpha(w_k)) < 2^{-k} \\ & \text{and } (\forall k) d(f, \alpha(w_k)) \leq 2^{-k}. \end{aligned}$$

Similar to the definition of the Cauchy representation of the real numbers ρ_C we consider in (2) only fast converging Cauchy sequences of rational polygon functions as names. Instead of $(\forall k) d(f, \alpha(w_k)) \leq 2^{-k}$ we can also write $f = \lim\limits_{k \to \infty} \alpha(w_k)$. If

$\delta_C(w_0¢w_1¢\ldots) = f$ then the graph of f is the intersection of all the closed balls $Bc(\alpha(w_k), 2^{-k})$. The representation is equivalent to the representation δ' obtained from $\delta_{\mathbb{R}} :\subseteq \Sigma^\omega \longrightarrow C(\mathsf{IR})$ by restricting the domains from IR to $[0;1]$:

Theorem 6.6

Define $\delta' :\subseteq \Sigma^\omega \longrightarrow C[0;1]$ by

$$\delta'(p)(x) := \begin{cases} \delta_{\mathbb{R}}(p)(x) & \text{if } 0 \le x \le 1 \\ div & \text{otherwise} \end{cases}$$

for all $p \in \Sigma^\omega$ and $x \in \mathsf{IR}$.
Then $\delta' \equiv \delta_C$.

As a consequence, Theorem 6.2 holds accordingly for δ_C instead of $\delta_{\mathbb{R}}$. $C[0;1]$ has other important dense subsets, e.g. the polynomial functions with rational coefficients or the trigonometric polynomials with rational coefficients. Standard notations of these dense subsets induce Cauchy representations which are equivalent to δ_C. For the functions from $C[0;1]$ a modulus of continuity can be computed from their δ_C–names. We mention without proof that the representation δ_C is admissible where the final topology is generated by the open balls of the metric space $(C[0;1], d)$. As a consequence the continuity theorem, Theorem 4.3, can be applied to δ_C.

Corollary 6.7 (*modulus of continuity*)

There is a computable function $g :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ such that $\delta_\omega g(p)$ is a modulus of continuity of $\delta_C(p)$ on $[0;1]$ for all $p \in dom(\delta_C)$.

Proof

By Theorem 6.6 there is a computable function $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ with $\delta_C(p) = \delta' f(p)$. The modulus of continuity of $\delta_{\mathbb{R}}(f(p))$ on $[-1;1]$ is a modulus of continuity of $\delta_C(p)$ on $[0;1]$. Define $g(p) = h(f(p), 1)$ with h from Theorem 6.4.
□

For a function $f \in C[0;1]$ the number $y = \max\{f(x) \mid x \in [0;1]\}$ is called the *maximum value* of f, and any x with $f(x) = y$ is called a *maximum point* of f. For functions f from $C[0;1]$ the maximum values can be determined effectively. Determination of maximum points will be reduced to the determination of zeros in Chapter 7.

Theorem 6.8 (*determination of maximum*)

The function $Max : C[0;1] \longrightarrow \mathrm{IR}$ defined by $Max(f) := \max\{f(x) \mid 0 \leq x \leq 1\}$ is (δ_C, ρ_C)–computable.

Proof

Let M be a Type 2 machine which for input $p = w_0¢w_1¢\ldots \in dom(\delta_C)$ determines a sequence $u_0\sharp u_1\sharp\ldots$ where $\overline{u}_n := Max(\alpha(w_{n+1}))$. Let $f := \delta_C(p)$, $f(x) = Max(f)$, $f_n := \alpha(w_{n+1})$, $f_n(x_n) = Max(f_n)$. Then for any $n \in \omega$

$$f_n(x_n) - 2^{-n-1} \leq f(x_n) \leq f(x) \leq f_n(x) + 2^{-n-1} \leq f_n(x_n) + 2^{-n-1},$$

therefore $|\overline{u}_n - Max(f)| \leq 2^{-n-1}$. We obtain $Max(f) = \rho_C\ (u_0\sharp u_1\sharp\ldots)$.
□

Especially, the maximum value of a computable function $f \in C[0;1]$ is computable.

We close this section with some remarks on differentiation and integration. By the next theorem differentiation on the set $C^1[0;1]$ of the continuously differentiable functions from $C[0;1]$ cannot be performed effectively, if δ_C is used as the naming system.

Theorem 6.9 (*non–effectivity of differentiation*)

The differentiation operator $Diff :\subseteq C[0;1] \longrightarrow C[0;1]$, defined by $Diff(f) = g$ iff g is the derivative of f (for all $f, g \in C[0;1]$), is not (δ_C, δ_C)–continuous.

Proof

Assume that $Diff$ is (δ_C, δ_C)–continuous. Since the continuity theorem 4.3 can be applied to δ_C, $Diff$ must be continuous. But this is false. Consider the functions $f, f_1, f_2, \ldots \in C^1[0;1]$ defined by $f(x) := 0, f_n(x) := \sin(n\pi x)/n$ for all $n \geq 1$ and $x \in [0;1]$. Then $(f_n)_{n\geq1}$ converges to f, but $(Diff(f_n))_{n\geq1}$ does not converge to $Diff(f)$.
□

Thus the δ_C–names of functions $f \in C^1[0;1]$ do not contain sufficiently much finitely accessible information in order to compute δ_C–names of the derivatives. On the other hand, the integration operator is computable.

Theorem 6.10 (*computability of integration*)

The integration operator $Int :\subseteq C(\mathbb{R}) \times \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$, defined by

$$Int(f, a, b) := \int_a^b f(x)dx,$$

is $(\delta_{\mathbb{R}}, \rho_C, \rho_C, \rho_C)$–computable.

We omit the proof. As a corollary the operator $Int_0 : C[0;1] \longrightarrow \mathbb{R}$, where $Int_0(f) = \int_0^1 f(x)dx$, is (δ_C, ρ_C)–computable.

7 Determination of zeros

Determination of zeros is an important task in numerical analysis. In this Chapter we study under which circumstances zeros of functions from $C[0;1]$ can be determined effectively.

For a continuous function $f :\subseteq \mathbb{R} \longrightarrow \mathbb{R}$ the set $\{x \in \mathbb{R} \mid f(x) \neq 0\}$ of the non–zeros is open, and for every open set X there is a continuous function $f : \mathbb{R} \longrightarrow \mathbb{R}$ such that X is the set of non–zeros. We prove a computable version of this fact (see Defs. 5.1, 6.1).

Theorem 7.1 (*characterization of the set of zeros*)

Let $S := \{(f, X) \in C(\mathbb{R}) \times \tau_{\mathbb{R}} \mid f^{-1}\{0\} = \mathbb{R} \setminus X\}$. Then

(1) S is $(\delta_{\mathbb{R}}, \delta_{op})$–computable,

(2) S^{-1} is $(\delta_{op}, \delta_{\mathbb{R}})$–computable.

Proof

(1) Since $\mathbb{R} \setminus \{0\}$ is δ_{op}–computable, the statement follows immediately from Theorem 6.3(2).

(2) For any $p = u_0\sharp v_0\sharp \ldots \in dom(\delta_{op})$ define $\delta(p) : \mathbb{R} \longrightarrow \mathbb{R}$ by

$$\delta(p)(x) := \sum_{n\in\omega} f_n(x) \cdot 2^{-n}$$

where

$$f_n(x) := \begin{cases} min(1, \overline{v}_n - x, x - \overline{u}_n) & \text{if } \overline{u}_n < x < \overline{v}_n \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mathbb{R} \setminus \delta_{op}(x) = \delta(p)^{-1}\{0\}$.

An easy estimation shows that the function $\delta :\subseteq \Sigma^\omega \longrightarrow C(\mathbb{R})$ has a (δ, ρ_C, ρ_C)–computable apply function. By Theorem 6.2(2) we obtain $\delta \leq \delta_{\mathbb{R}}$, i.e. there is a computable function $g :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ with $\delta(p) = \delta_{\mathbb{R}} g(p)$ for all $p \in dom(\delta_{op})$. Therefore, $(\delta_{op}(p),\ \delta_{\mathbb{R}} g(p)) \in S^{-1}$ for all $p \in dom(\delta_{op})$.
□

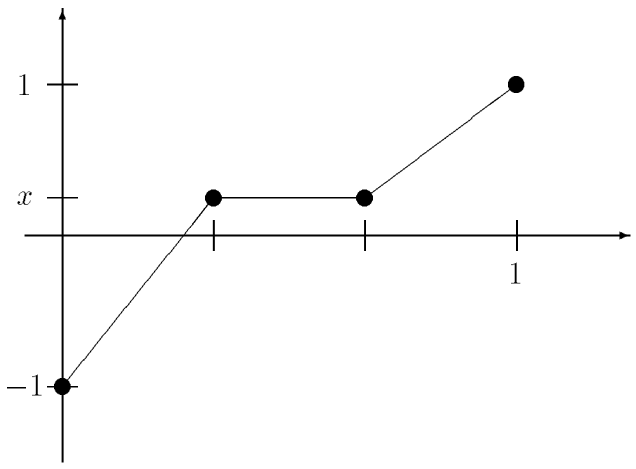
It can be shown that there is some δ_{op}–computable set $X \subseteq \mathbb{R}$ such that the Lebesgue measure $\mu(X)$ is less than 1/2 and $x \in X$ for every δ_C–computable real number [Spe 59, Wei 87, Wei 94]. Therefore by Theorem 7.1(2), there is a computable function with many zeros (e.g. in the interval $[0;1]$) but without any computable zero. As a consequence, the relation $R := \{(f,x) \in C[0;1] \times \mathbb{R} \mid f(x) = 0\}$ cannot be $(\delta_C, \rho_{\mathbb{R}})$–computable, since computable functions $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ map computable elements to computable elements. We prove that R is not even (δ_C, ρ_C)–continuous.

Theorem 7.2 (*impossibility of zero finding*)

Let $R := \{(f,x) \in C[0;1] \times \mathbb{R} \mid f(x) = 0\}$. Then R is not (δ_C, ρ_C)–continuous.

Proof

For any $x \in \mathbb{R}$ define the polygon function $G(x)$ by the vertices $(0,-1)$, $(1/3,x)$, $(2/3,x)$, $(1,1)$.



Define $\delta :\subseteq \Sigma^\omega \longrightarrow C[0;1]$ by $\delta(p) := G(\rho_C(p))$. Then the apply function of δ is (δ, ρ_C, ρ_C)–computable. Since Theorem 6.2 holds accordingly for δ_C, we obtain $\delta \leq \delta_C$, i.e. there is some computable function g with $G(\rho_C(p)) = \delta_C g(p)$ for all $p \in dom(\rho_C)$.
Now assume, that there is a continuous function h with $\delta_C(p)\rho_C h(p) = 0$ if $\delta_C(p)$ has a zero. Let $q = 0\sharp 0\sharp \ldots$. Then $\rho_C(q) = 0$ and $y := \rho_C hg(q)$ is a zero of $\delta_C g(q) = G\rho_C(q) = G(0)$. Obviously $1/3 \leq y \leq 2/3$. First we consider the case $y > 1/3$. There is a sequence $(q_i)_{i\in\omega}$ in Σ^ω with $\rho_C(q_i) = 2^{-i}$ and $\lim\limits_{i\to\infty} q_i = q$. Since $y_i := \rho_C hg(q_i)$ is a zero of $\delta_C g(q_i) = G\rho_C(q_i) = G(2^{-i})$, we have $y_i < 1/3$ for all $i \in \omega$. Since $\rho_C hg$ is continuous, we have

$$1/3 < y = \rho_C hg(q) = \rho_C hg(\lim_{i\to\infty} q_i) = \lim_{i\to\infty} \rho_C hg(q_i) = \lim_{i\to\infty} y_i \leq 1/3.$$

This is a contradiction. The case $y < 2/3$ is handled accordingly.
□

Notice, that even the very small subset $R' := R \cap \{G(x) \mid x \in R\} \times \mathbb{R}$ is not

(δ_C, ρ_C)–continuous. The contradiction has been derived by using the function $G(0)$ which is zero on an open interval. If we exclude such situations, and if we consider only functions which change their sign on $[0;1]$, we obtain a positive result. The following theorem is an effective version of a generalized intermediate value theorem from classical analysis: If $f : [0;1] \longrightarrow \mathbb{R}$ is continuous and changes its sign, then f has a zero.

Theorem 7.3 (*non extensional solution*)

Let $F_{nd} := \{f \in C[0;1] \mid (\exists x, y) f(x) \cdot f(y) < 0$ and $I \subseteq f^{-1}\{0\}$ for no open interval $I \subseteq [0;1]\}$. Let

$$R_{nd} := \{(f, x) \in F_{nd} \times \mathbb{R} \mid f(x) = 0\}.$$

Then

(1) R_{nd} is (δ_C, ρ_C)–computable,

(2) R_{nd} has no (δ_C, ρ_C)–continuous choice function.

Proof

(1) The following observations can be proved easily.

- Let $f \in F_{nd}$ and $a, b \in [0;1]$ with $f(a) \cdot f(b) < 0$. Then there are rational numbers $a', b' \in \mathbb{Q}$ with $a < a' < b' < b$, $(b' - a') \le (b - a)/2$, $f(a) \cdot f(a') > 0$, $f(a') \cdot f(b') < 0$ and $f(b') \cdot f(b) > 0$.
- The sets $\{(u, p) \mid \delta_C(p)(\overline{u}) > 0\}$ and $\{(u, p) \mid \delta_C(p)(\overline{u}) < 0\}$ are r.e. in $\Sigma^* \times dom(\delta_C)$.

There is a Type 2 machine M which for input $p \in dom(\delta_C)$ computes sequences $u_0, u_1, \ldots$ and $v_0, v_1, \ldots$ of elements of $dom(\nu_Q)$ and produces the output $q := u_0 \sharp u_1 \sharp \ldots \in dom(\rho_C)$ according to the following rules. First, M searches for words u_0, v_0 such that $\delta_C(p)(\overline{u}_0) \cdot \delta_C(p)(\overline{v}_0) < 0$. Assume u_{n-1} and v_{n-1} have been determined. Then M searches for words u_n, v_n with $\overline{u}_{n-1} < \overline{u}_n < \overline{v}_n < \overline{v}_{n-1}$, $(\overline{v}_n - \overline{u}_n) \le (\overline{v}_{n-1} - \overline{u}_{n-1})/2$, $\delta_C(p)(\overline{u}_{n-1}) \cdot \delta_C(p)(\overline{u}_n) > 0$, $\delta_C(p)(\overline{u}_n) \cdot \delta_C(p)(\overline{v}_n) < 0$ and $\delta_C(p)(\overline{v}_n) \cdot \delta_C(p)(\overline{v}_{n-1}) > 0$.

Assume $p \in \delta_C^{-1} F_{nd}$ is the input for M. By the above observations, M determines some $q = u_0 \sharp u_1 \sharp \ldots \in dom(\rho_C)$. Let $f := \delta_C(p)$ and $x := \rho_C(q)$. We prove $f(x) = 0$. We have $\lim\limits_{i \to \infty} \overline{u}_i = \lim\limits_{i \to \infty} \overline{v}_i = x$. Consider the case $f(\overline{u}_0) > 0$. Then $f(\overline{u}_i) > 0$ and $f(\overline{v}_i) < 0$ for all $i \in \omega$. By continuity of f we have $f(x) = f(\lim\limits_{i \to \infty} \overline{u}_i) = \lim\limits_{i \to \infty} f(\overline{u}_i) \ge 0$ and $f(x) = f(\lim\limits_{i \to \infty} \overline{v}_i) = \lim\limits_{i \to \infty} f(\overline{v}_i) \le 0$, therefore $f(x) = 0$. If $f(\overline{u}_0) < 0$, we obtain $f(x) = 0$ correspondingly.

(2) Suppose that there is a (δ_C, ρ_C)–continuous function $Z :\subseteq C[0;1] \longrightarrow \mathbb{R}$ such that $fZ(f) = 0$ for all $f \in F_{nd}$. For $x \in \mathbb{R}$ let $G(x)$ be the polygon function with the vertices $(0, -1)$, $(1/3, x)$, $(2/3, x - 1)$, $(1, 1)$. Then $G : \mathbb{R} \longrightarrow$

$C[0;1]$ is (ρ_C, δ_C)–computable, hence (ρ_C, δ_C)–continuous. Therefore, the function $ZG : \mathsf{IR} \longrightarrow \mathsf{IR}$ is (ρ_C, ρ_C)–continuous, i.e. continuous by Theorem 4.3, and has the property that $ZG(x)$ is a zero of $G(x)$ for all $x \in \mathsf{IR}$. Since continuous functions map intervals onto intervals, $I_1 := ZG(1/3; 3) \subseteq (0; 1/3)$ (since $ZG(2) = 1/9 \in I_1$ and $1/3 \notin I_1$) and $I_2 := ZG(-2; 2/3) \subseteq (2/3; 1)$ (since $2/3 \notin I_2$ and $ZG(-1) = 8/9 \in I_2$). This is contradiction.

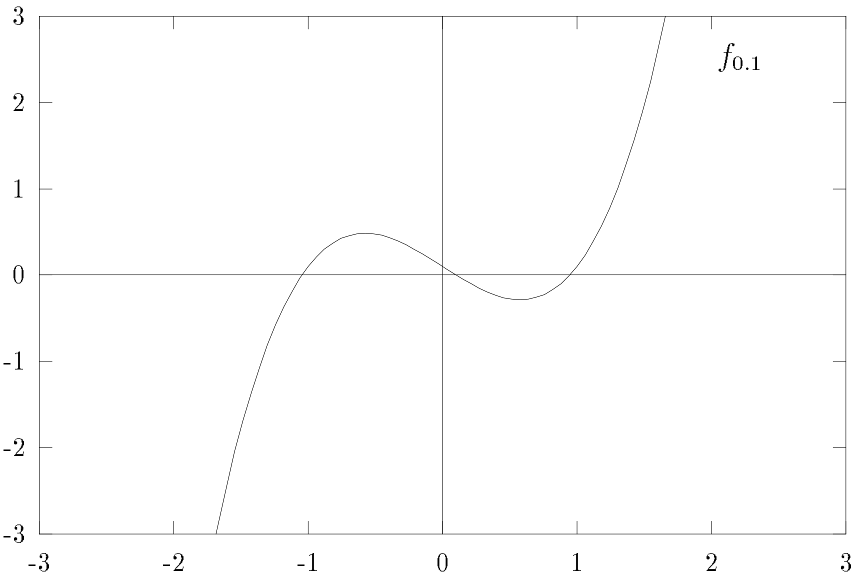
□

The following example illustrates Theorem 7.3.

Example 1
Consider the problem to determine a zero of the function $f_a :\subseteq \mathsf{IR} \longrightarrow \mathsf{IR}$ from a given number $a \in [-1; 1]$ where

$$f_a(x) = x^3 - x + a$$

for all $x \in \mathsf{IR}$. (Since for $a \in [-1; 1]$ the zeros of f_a are in the interval $[-2; 2]$, we may restrict the domains to $[-2; 2]$ and assume $f_a \in C[-2; 2]$ for all $a \in [-1; 1]$.)



By the method described in (1) of the proof of Theorem 7.3, for given $a \in [-1;1]$ one determines sequences of rational numbers $(a_i)_{i\in\omega}$ and $(b_i)_{i\in\omega}$ with

$$a_{i-1} < a_i < b_i < b_{i-1},\ f_a(a_i) < 0,\ f_a(b_i) > 0,\ b_i - a_i \le (b_{i-1} - a_{i-1})/2.$$

The sequence $(a_i)_{i\in\omega}$ converges with speed 2^{-i} to a zero x_a of f_a. If f_a has 3 zeros then it may depend on the given name $p \in \delta_C^{-1}\{a\}$ which zero (the leftmost or the rightmost) is determined. For every algorithm such dependence on the names must occur, since there is no continuous function $Z : [-1;1] \longrightarrow \mathrm{IR}$ with $f_a Z(a) = 0$ for all $a \in [-1;1]$. The proof is quite similar to that of Theorem 7.3(2).

As a corollary of Theorem 7.3(1) we obtain a computable version of the intermediate value theorem.

Corollary 7.4

Let $F_{iv} := \{f \in C[0;1] \mid f$ is increasing and $f(0)\cdot f(1) < 0\}$. The function $Z : C[0;1] \longrightarrow \mathrm{IR}$ with $dom(Z) = F_{iv}$ and $Z(f) :=$ (the zero of f) is (δ_C, ρ_C)–computable.

The function Z from this corollary can be extended to all continuous functions which have exactly one zero.

Theorem 7.5

Let $F_1 := \{f \in C[0;1] \mid f$ has exactly one zero$\}$. The function $Z : C[0;1] \longrightarrow \mathrm{IR}$ with $dom(Z) = F_1$ and $Z(f) :=$ (the zero of f) is (δ_C, ρ_C)–computable.

Proof

Let $\nu_\Sigma : \omega \longrightarrow \Sigma^*$ be some standard numbering of Σ^*. Let M be a Type 2 machine which for input $p = w_0 ¢ w_1 ¢ \ldots \in dom(\delta_C)$ produces a sequence $q = u_0 \sharp v_0 \sharp u_1 \sharp v_1 \sharp \ldots$ such that

$$(u_i, v_i) = \begin{cases} (u,v) & \text{if } \nu_\Sigma(i) = 0^k \sharp u \sharp v \text{ with } \overline{u} < \overline{v} \text{ and} \\ & |\alpha(w_k)(x)| > 2\cdot 2^{-k} \text{ for all } x \in [0;1] \setminus (\overline{u};\overline{v}) \\ (-1,10) & \text{otherwise.} \end{cases}$$

Then $\delta_C(p)\rho_I f_M(p) = 0$ whenever $\delta_C(p) \in F_1$.
□

Corollary 7.6

If $f \in C[0,1]$ is computable and $x \in [0;1]$ is an isolated zero of f, then x is computable.

Proof

Assume $0 < x < 1$. Then there are rational numbers r, s with $0 \leq r < x < s \leq 1$ such that x is the only zero of f in $[r;s]$. Define $f' \in C[0;1]$ by

$$f'(y) := \begin{cases} f(r) & \text{if } 0 \leq y < r \\ f(y) & \text{if } r \leq y \leq s \\ f(s) & \text{if } s \leq y < 1 \\ div & \text{otherwise.} \end{cases}$$

Then f' is computable (c.f. Lemma 3.8), and x is its only zero. By Theorem 7.5 we have $x = Z(f')$. Since Z is (δ_C, ρ_C)–computable and f' is δ_C–computable, $x = Z(f')$ is ρ_C–computable.
□

Although there is no general method of determining zeros for continuous functions it is possible to determine for $f \in C[0;1]$ and $n \in \omega$ some $x \in \mathbb{R}$ (even $x \in \mathbb{Q}$) with $|f(x)| < 2^{-n}$ (provided f has a zero).

Theorem 7.7 (*approximate zero*)

The relation

$$R := \{(f, n, s) \in C[0;1] \times \omega \times \mathbb{Q} \mid |f(s)| < 2^{-n}\}$$

is $(\delta_C, \nu_{bin}, \nu_Q)$–computable.

Proof

There is a Type 2 machine M which for inputs $p = w_0 ¢ w_1 ¢ \ldots \in dom(\delta_C)$ and $n \in \omega$ searches for some $k \in \omega$ and $u \in dom(\nu_Q)$ with $|\alpha(w_k)(\overline{u})| < 2^{-n-1}$. As soon as the search has been successful, M gives u as its output.
□

For every continuous increasing function $f \in C(\mathbb{R})$ the inverse function f^{-1} is continuous. We prove a computational version of this theorem (for simplicity only for functions f with $range(f) = \mathrm{IR}$; generalizations are straightforward).

Theorem 7.8 (*inverse function*)

The function $Inv :\subseteq C(\mathrm{IR}) \longrightarrow C(\mathrm{IR})$ with

$$Inv(f) := \begin{cases} f^{-1} & \text{if } f \text{ is increasing and } range(f) = \mathrm{IR} \\ div & \text{otherwise} \end{cases}$$

is $(\delta_{\mathbb{R}}, \delta_{\mathbb{R}})$–computable.

Proof

We generalize the method for determining zeros of continuous increasing functions. Since $(x, y) \mapsto x - y$ is computable on IR, by Theorem 6.2(2) the function $H : C(\mathrm{IR}) \times \mathrm{IR} \times \mathrm{IR} \longrightarrow \mathrm{IR}$ with $H(f, x, y) := f(x) - y$ is $(\delta_{\mathbb{R}}, \rho_C, \rho_C, \rho_C)$–computable. Let M be a Type 2 machine which for inputs $p \in dom(\delta_{\mathbb{R}})$ and $q \in dom(\rho_C)$ computes a sequence $r = u_0\sharp u_1\sharp\ldots$ as follows. For determining u_n, M searches for $u, v \in dom(\nu_Q)$ such that $H(\delta_{\mathbb{R}}(p), \overline{u}, \rho_C(q)) < 0$, $H(\delta_{\mathbb{R}}(p), \overline{v}, \rho_C(q)) > 0$ and $\overline{v} - \overline{u} < 2^{-n}$. As soon as the search has been successful, M chooses $u_n := u$. Since H is computable, the search can in fact be programmed by a Type 2 machine. The search is successful for every $n \in \omega$ and $q \in dom(\rho_C)$ if $\delta_{\mathbb{R}}(p)$ is increasing and has the range IR. Consider $f = \delta_{\mathbb{R}}(p) \in dom(Inv)$ and $y = \rho_C(q)$. Then

$$f\rho_C f_M(p, q) - y = 0, \text{ i.e. } f^{-1}(y) = \rho_C f_M(p, q).$$

Define $\delta :\subseteq \Sigma^\omega \longrightarrow C(\mathrm{IR})$ by $\delta(p) := (\delta_{\mathbb{R}}(p))^{-1}$. Then $\delta(p)\rho_C(q) = \rho_C f_M(p, q)$, i.e. the apply–function of δ is (δ, ρ_C, ρ_C)–computable. By Theorem 6.2(2) we obtain $\delta \leq \delta_{\mathbb{R}}$. This means that there is a computable function $g :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ with

$$Inv(\delta_{\mathbb{R}}(p)) = \delta(p) = \delta_{\mathbb{R}} g(p)$$

for all p with $\delta_{\mathbb{R}}(p) \in dom(Inv)$. Therefore Inv is $(\delta_{\mathbb{R}}, \delta_{\mathbb{R}})$–computable.
□

While for functions from $C[0; 1]$ maximum values can be computed by Theorem 6.8, the determination of maximum points is as difficult as the determination of zeros. This follows from the following observation:

– x is a zero of f, iff x is a maximum point of g where $g(x) = -|f(x)|$

– x is a maximum point of f, iff x is a zero of h where $h(x) = f(x) - Max(f)$.

Notice that Max is computable by Theorem 6.8 and that computability of $(x, y) \mapsto x - y$ and $x \rightarrow |x|$ can be derived from Theorem 3.7.

8 Computation Time and Lookahead on Σ^ω

Time and tape complexity are the most important computational complexity measures for Turing machine computations. They model time and storage requirement of digital computers quite realistically. In this section we introduce the time complexity for Type 2 machines M with $f_M :\subseteq (\Sigma^\omega)^m \longrightarrow \Sigma^\omega$. As a further important concept we define the *input lookahead* which measures the amount of information which is used during a computation. We prove, that co–r.e. sets are the natural classes with uniform time bound.

Let M be a Turing machine with $f_M :\subseteq (\Sigma^*)^m \longrightarrow \Sigma^*$. The computation time of M for input $(x_1, \ldots, x_m)$ is defined by

$$\begin{aligned} Time_M(x_1, \ldots, x_m) := \ & \text{the number of computation steps which } M \text{ with input} \\ & (x_1, \ldots, x_m) \text{ needs until it reaches a HALT statement.} \end{aligned}$$

A function $t : \omega \longrightarrow \omega$ is a time bound for M, iff

$$Time_M(x_1, \ldots, x_m) \leq t(\max_i lg(x_i)) \text{ for all } (x_1, \ldots, x_m) \in (\Sigma^*)^m.$$

Example 1
Consider the multiplication of natural numbers in binary notation. Using the school method, a Turing machine M can be constructed such that

– $\nu_{bin} f_M(u, v) = \nu_{bin}(u) \cdot \nu_{bin}(v)$ for all $u, v \in dom(\nu_{bin})$,

– $Time_M(u, v) \leq cn^2 + c$ where $n = \max(lg(u), lg(v))$ and $c \in \omega$ is a constant.

Therefore, M multiplies binary numbers in time t for some $t \in O(n^2)$.

Remember, for $f : \omega^m \longrightarrow \omega$

$$O(f) = \{g : \omega^m \longrightarrow \omega \mid (\exists c)(\forall \overline{x} \in \omega^m) g(\overline{x}) \leq cf(\overline{x}) + c\}.$$

For $t : \omega \longrightarrow \omega$,

$$\begin{aligned} TIME(t) := \{f_M \mid \ & M \text{ is a Turing machine and some } t' \in O(t) \\ & \text{is a time bound for } M\} \end{aligned}$$

is the *complexity class* of functions computable on Turing machines in Time $O(t)$.

The above definition of $Time_M$ cannot be used for machines with infinite output since valid computations never reach a HALT statement. We introduce as a further parameter a number $k \in \omega$ and measure the time until M has produced the output symbol $q(k)$ of its infinite output $q \in \Sigma^\omega$. Another important information is the *input lookahead*, i.e. the number of input symbols which M requires for producing the output sequence $q(0)\dots q(k)$. In the following, we consider only the case $Y = (\Sigma^\omega)^m$ for some $m \in \omega$.

Definition 8.1 (*time and input lookahead*)

Let M be a Type 2 machine with $f_M :\subseteq (\Sigma^\omega)^m \longrightarrow \Sigma^\omega$. For all $y \in (\Sigma^\omega)^m$ and $k \in \omega$ define *time* and *input lookahead* by:

$$\begin{aligned} Time_M(y)(k) := \ & \text{the number of steps which } M \text{ with input } y \\ & \text{needs until the } k\text{th output symbol has been written,} \\ Ila_M(y)(k) := \ & \text{the maximal } j \text{ such that } M \text{ with input } y \\ & \text{reads the } j\text{th symbol from some input tape during} \\ & \text{the first } Time_M(y)(k) \text{ computation steps.} \end{aligned}$$

Notice that $Time_M(y)(k)$ may exist for some but not for all $k \in \omega$ (in such a case $y \notin dom(f_M)$). Since reading an input symbol requires at least one computation step, $Ila_M(y)(k) \leq Time_M(y)(k)$. The input lookahead $Ila(y) : \omega \longrightarrow \omega$ is a modulus of continuity of the function $f_M :\subseteq (\Sigma^\omega)^m \longrightarrow \Sigma^\omega$ in the point $y \in dom(f_M)$.

While for a Turing machine $Time_M(y)$ is a natural number for any $y \in dom(f_M)$, for a Type 2 machine M with $f_M :\subseteq (\Sigma^\omega)^m \longrightarrow \Sigma^\omega$, the function $Time_M(y) : \omega \longrightarrow \omega$ determines the *computation time* of M with input $y \in dom(f_M)$ as a function of the *output precision*, and $Ila_M(y) : \omega \longrightarrow \omega$ determines the *amount* of *input information* used by M with input $y \in dom(f_M)$ as a function of the *output precision.*

For any Type 2 machine M, the properties $Time_M(y)(k) = t$ and $Time_M(y)(k) \leq t$ are decidable, and the properties $Ila_M(y)(k) = t$ and $Ila_M(y)(k) \leq t$ are r.e. in (y, k, t). A simple comterexample shows that $Ila_M(y)(k) = t$ and $Ila_M(y)(k) \leq t$ are not recursive in general. We shall consider bounds for time and input lookahead which are uniform for all $y \in X$ for some $X \subseteq (\Sigma^\omega)^m$. The sets $X \subseteq (\Sigma^\omega)^m$ such that $Time_M(y)$ has a computable bound uniform for all $y \in X$ can be characterized easily. A set $X \subseteq (\Sigma^\omega)^m$ is called co–r.e., iff $(\Sigma^\omega)^m \setminus X$ is r.e.

Theorem 8.2 (*uniform time on co–r.e. sets*)

Let M be a Type 2 machine with $f_M :\subseteq (\Sigma^\omega)^m \longrightarrow \Sigma^\omega$.

(1) If $X \subseteq dom(f_M)$ is co–r.e., then $(\forall y \in X)(\forall k) Time_M(y)(k) \leq t(k)$ for some computable function $t : \omega \longrightarrow \omega$.

(2) If $t : \omega \longrightarrow \omega$ is computable, then

$$X := \{y \in (\Sigma^\omega)^m \mid (\forall k) Time_M(y)(k) \leq t(k)\}$$

is co–r.e. and $X \subseteq dom(f_M)$.

Proof

For simplicity we consider only the case $m = 1$. The general case is proved accordingly. We use the important fact that the metric space (Σ^ω, d) is compact.

(1) Since X is co–r.e., there is some Type 2 machine N with $f_N :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$ such that $\Sigma^\omega \setminus X = dom(f_N)$. Consider $k \in \omega$. Then for any $p \in \Sigma^\omega$ there is some n such that

$$Time_N(p) = n \text{ or } Time_M(p)(k) = n.$$

Let n_p be the first such n and w_p the prefix of p of length n_p. Since $\Sigma^\omega = \cup\{w_p\Sigma^\omega \mid p \in \Sigma^\omega\}$ and Σ^ω is compact, there is a finite set $A \subset \Sigma^\omega$ with $\Sigma^\omega = \cup\{w_p\Sigma^\omega \mid p \in A\}$. Determine from $k \in \omega$ a number $t(k) \in \omega$ as follows: Search for a finite set W of words with $\Sigma^\omega = \cup\{w\Sigma^\omega \mid w \in W\}$ and $Time_N(w0^\omega) = lg(w)$ or $Time_M(w0^\omega)(k) = lg(w)$ for all $w \in W$. By the above considerations, such a set W exists. Define $t(k) := \max\{lg(w) \mid w \in W \text{ and } Time_M(w0^\omega)(k) = lg(w)\}$. Then $Time_M(p)(k) \leq t(k)$ for all $p \in X$. The function $t : \omega \longrightarrow \omega$ is computable.

(2) Let $t : \omega \longrightarrow \omega$ be computable. There is a Type 2 machine N which halts for input $p \in \Sigma^\omega$, iff $Time_M(p)(k) \not\leq t(k)$ for some $k \in \omega$. Then $dom(f_N) = \Sigma^\omega \setminus X$.

□

We shall call a sequence $p \in \Sigma^\omega$ computable in time $t : \omega \longrightarrow \omega$, iff there is a Type 2 machine M with $f_M :\subseteq (\Sigma^\omega)^0 \longrightarrow \Sigma^\omega$ such that $f_M(\,) = p$ and $Time_M(\,)(k) \leq t(k)$ for all $k \in \omega$.

9 Computational Complexity of Real Functions

In this section we introduce a new representation of the real numbers for measuring the time complexity of real functions. We prove bounds of time and input lookahead for addition, multiplication and, as an application of Newton's method, inversion. Finally we discuss the computational complexity of compact sets.

By the Main Theorem 4.3, a real function is continuous, iff it is determined by a continuous function on ρ_C–names. By definition, a real function is computable, iff it is determined by a computable function on ρ_C–names. We would like to call a real function computable in time $t : \omega \longrightarrow \omega$, iff it is determined by a function on ρ_C–names computable in time t.

Unfortunately, this definition is unreasonable. First, we observe, that any ρ_C–name p of a number $x \in \mathsf{IR}$ can be padded arbitrarily. Assume $p = u_0\sharp u_1\sharp \ldots$ and $\rho_C(p) = x$, and let $r : \omega \longrightarrow \omega$ be some function. Then some $q = w_0\sharp w_1\sharp \ldots$ with $\rho_C(q) = x$ can be determined easily such that $lg(w_i) \geq r(i)$ for all $i \in \omega$ (choose $w_i \in dom(\nu_Q)$ with very large numerator and denominator such that $|\overline{w}_i - \overline{u}_{i+1}| \leq 2^{-i-2}$). Let M be a Type 2 machine which computes a real function $g :\subseteq \mathsf{IR} \longrightarrow \mathsf{IR}$ on ρ_C–names. By padding the outputs of M, a machine M' can be constructed which computes g on ρ_C–names and operates in time $O(n)$. Therefore, every computable real function can be computed in time $O(n)$ on ρ_C–names.

To avoid this degeneracy, define temporarily: g is "computable in time $t : \omega \longrightarrow \omega$", iff some Type 2 machine M computes g on ρ_C–names such that $g(x)$ is determined with error $< 2^{-k}$ in at most $t(k)$ steps. But not even the identity $id : \mathsf{IR} \longrightarrow \mathsf{IR}$ is "computable in time t" for any $t : \omega \longrightarrow \omega$, since on the input tape arbitrarily redundant, i.e. padded, names are allowed.

We solve the problem by introducing a new representation $\rho :\subseteq \Sigma^\omega \longrightarrow \mathsf{IR}$ of the real numbers with $\rho \equiv \rho_C$, which does not allow padding. This representation is a generalization of the representation by infinite binary fractions, in which additionally the digit -1 may be used. We shall denote the digit -1 by $\overline{1} \in \Sigma$.

Definition 9.1 (*modified binary representation*)

Define $\rho :\subseteq \Sigma^\omega \longrightarrow \mathsf{IR}$ as follows (where $\overline{1}$ denotes the digit -1):

$$\begin{aligned} dom(\rho) := \ & \{a_n \ldots a_0 \cdot a_{-1}a_{-2} \ldots \mid n \geq -1, a_i \in \{\overline{1}, 0, 1\} \text{ for } i \leq n, \\ & a_n \neq 0 \text{ if } n \geq 0 \text{ and } a_n a_{n-1} \notin \{1\overline{1}, \overline{1}1\} \text{ if } n \geq 1\} \end{aligned}$$

$$\rho(a_n \ldots a_0 \cdot a_{-1}a_{-2} \ldots) := \Sigma\{a_i \cdot 2^i \mid i \leq n\}$$

Let $p = a_n \ldots a_0 \cdot a_{-1}a_{-2} \ldots \in dom(\rho)$ and $p[k] := a_n \ldots a_0 \cdot a_{-1} \ldots a_{-k}$ for $k \in \omega$. Then

$$\rho(p[k]0^\omega) \;=\; z \cdot 2^{-k} \text{ for some integer } z \in \mathbb{Z},$$

and for this number z:

$$\begin{aligned}
\rho(p[k]\Sigma^\omega) &= [z-1; z+1] \cdot 2^{-k} = [2z-2; 2z+2] \cdot 2^{-k-1} \\
\rho(p[k]\overline{1}\Sigma^\omega) &= [2z-2; 2z] \cdot 2^{-k-1} \\
\rho(p[k]0\Sigma^\omega) &= [2z-1; 2z+1] \cdot 2^{-k-1} \\
\rho(p[k]1\Sigma^\omega) &= [2z; 2z+2] \cdot 2^{-k-1}
\end{aligned}$$

Therefore p determines a sequence $(I_k)_{k\in\omega}$ of nested closed intervals, $I_k := \rho(p[k]\Sigma^\omega)$, such that:

- I_{k+1} is the left half of I_k if $a_{k+1} = \overline{1}$, the middle half I_k if $a_{k+1} = 0$ and the right half of I_k if $a_{k+1} = 1$,
- length $(I_k) = 2 \cdot 2^{-k}$,
- $\rho(p) = \cap\{I_k \mid k \in \omega\}$.

For reducing redundancy we have excluded the prefix 0, the prefix $1\overline{1}$, which can be replaced by 1, and the prefix $\overline{1}1$ which can be replaced by $\overline{1}$.Although the representation ρ is not injective (no representation equivalent to ρ_C can be injective by Theorem 4.5), the sets $\rho^{-1}X$ for compact $X \subseteq \mathbb{R}$ and especially the sets $\rho^{-1}\{x\}$ ($x \in \mathbb{R}$) are compact, i.e. "small". Remember that a subset $X \subseteq \Sigma^\omega$ of the Cantor space is compact, iff it is closed.

Theorem 9.2

(1) $\rho \equiv \rho_C$

(2) For any compact subset $X \subseteq \mathbb{R}$, $\rho^{-1}X \subseteq \Sigma^\omega$ is compact.

(3) For any κ_w–computable subset $X \subseteq \mathbb{R}$, $\rho^{-1}X$ is co–r.e. (see Def. 5.4.(3)).

Proof

(1) Translators from ρ to ρ_C and vice versa can be programmed easily.

(2) Let $(p_i)_{i\in\omega}$ be a sequence in $\rho^{-1}X$ converging to some $p \in \Sigma^\omega$. Since X is bounded, there is some $k \in \omega$ such that each p_i has the form $w_i.q_i$ with $lg(w_i) \leq k$. We conclude $p \in dom(\rho)$. The representation ρ is continuous since ρ_C is continuous and $\rho \equiv \rho_C$. By continuity, $p_i \to p \in dom(\rho)$ implies $\rho(p_i) \to \rho(p)$. Since $\rho(p_i) \in X$ for all i and since X is closed, we obtain $\rho(p) \in X$, hence $p \in \rho^{-1}X$. Therefore $\rho^{-1}X$ is closed and compact.

(3) We leave the proof to the reader.
$\square$

By Theorem 8.2 we know that the time of a Type 2 machine is uniformly bounded by a computable function on any co–r.e. subset of its domain which is especially compact. By Theorem 9.2(3), every (ρ, ρ)–computable real function has a uniform computable complexity bound on every κ_w–computable subset of its domain.

Definition 9.3

Let $f :\subseteq \mathbb{R}^m \longrightarrow \mathbb{R}$ be a computable function, let $X \subseteq dom(f)$ and let $s : \omega \longrightarrow \omega$ and $t : \omega \longrightarrow \omega$ be functions.
A Type 2 machine M computes f on X in time t with input lookahead s, iff

– $f(\rho(p_1), \ldots, \rho(p_m)) = \rho f_M(p_1, \ldots, p_m)$,

– $Time_M(p_1, \ldots, p_m)(n) \leq t(n)$,

– $Ila_M(p_1, \ldots, p_m)(n) \leq s(n)$

for all $n \in \omega$ whenever $(\rho(p_1), \ldots, \rho(p_m)) \in X$.

As a first example we consider addition on $\mathbb{R}$.

Lemma 9.4 (*addition*)

There is a Type 2 machine operating in time $O(k)$ with input lookahead $k + 2$ such that

$$\rho f_M(p, q) = \rho(p) + \rho(q)$$

for all $p, q \in .\{\overline{1}, 0, 1\}^\omega$.

Proof

Consider $p = .a_1a_2\ldots$ and $q = .b_1b_2\ldots$ $(a_i, b_i \in \{\overline{1}, 0, 1\})$. Define $r_{-1} := a_1 + b_1$. For $n \geq 0$ choose inductively $r_n \in \{-2, -1, 0, 1, 2\}$ and $c_n \in \{\overline{1}, 0, 1\}$ such that

$$2r_{n-1} + a_{n+2} + b_{n+2} = 4c_n + r_n.$$

If c_{n-1} with $|r_{n-1}| \leq 2$ exists then c_n and r_n with $|r_n| \leq 2$ exist. By induction, c_n and r_n exist for all $n \geq 0$. If $c_0 = 0$, define $f(p, q) := .c_1c_2\ldots$; if $c_0 \neq 0$, define $f(p, q) := c_0.c_1c_2\ldots$. Obviously, there is a Type 2 machine M, which produces

$f(p,q)$ in time $O(n)$ with input lookahead $\leq n+2$. We prove the correctness of M. By induction one shows easily

$$\sum_{i\leq n+2} a_i \cdot 2^{-i} + \sum_{i\leq n+2} b_i \cdot 2^{-i} = \sum_{i\leq n} c_i \cdot 2^{-i} + r_n \cdot 2^{-n-2}$$

for all $n \geq -1$. Consequently, $\rho(p) + \rho(q) = \rho f(p,q)$.
□

Theorem 9.5 (*addition*)

For every bounded subset $X \subseteq \mathrm{IR}^2$ there are constants c_1 and c_2 such that addition on X can be computed w.r.t. ρ by a Type 2 machine in time $c_1 \cdot n + c_1$ with input lookahead $n + c_2$.

Proof

There is some $m \in \omega$ such that $X \subseteq [-2^m; 2^m]^2$. If $\rho(p) \in [-2^m; 2^m]$ then $p = w.q$ for some $w \in \{\overline{1}, 0, 1\}^*$ with $lg(w) \leq m+1$. Let M be a Type 2 machine which for input (p,q) with $(\rho(p), \rho(q)) \in X$ shifts the points in p and q $m+1$ positions to the left, runs the machine from lemma 9.4 and shifts the point of the result $m+1$ positions to the right.
□

We reduce the multiplication of real numbers w.r.t. ρ to multiplication of binary integers by a doubling method. For obtaining good time estimations we need *regular* time bounds [FS 74, Mue 86]. As a tool we use the following improvement lemma, which we do not prove here.

Lemma 9.6 (*improvement lemma*)

Let $I := \rho(u.a_1 \ldots a_m \Sigma^\omega) \cap \rho(v.b_1 \ldots b_{m+k} \Sigma^\omega) \neq \emptyset$. Then there are $c_{m+1}, \ldots, c_{m+k} \in \{\overline{1}, 0, 1\}$ with $I \subseteq \rho(u.a_1 \ldots a_m c_{m+1} \ldots c_{m+k} \Sigma^\omega)$. A word $c_{m+1} \ldots c_{m+k}$ can be determined from u, v, $a_1 \ldots a_m$ and $b_1 \ldots b_{m+k}$ in time $O(n)$ where $n := lg(v) + m + k$.

We shall call a function $f : \omega \longrightarrow \omega$ *regular*, iff:

- f is non–decreasing and $(\exists n) f(n) \neq 0$ and
- there are numbers $n_0, c \in \omega$ with

$$2t(n) \leq t(2n) \leq ct(n) \text{ for all } n \geq n_0.$$

We state without proofs (see [Mue 86]) that for regular functions t:

- $n \in O(t)$,
- $t \in O(n^k)$ for some $k \in \omega$,
- $t(cn+c) \in O(t)$ for every $c \in \omega$,
- $\sum\{t(2^k) \mid k \leq \lceil log_2 n \rceil\} \in O(t)$.

Most of the commonly used bounds for complexity classes like polynomials, $n \cdot log^2 n$, $n \cdot log\, n \cdot log\ log\, n, \ldots$ are regular. In the following let $Mb : \omega \longrightarrow \omega$ be any regular upper time bound for binary integer multiplication on Turing machines. For example by Schönhage's method [Sch 71], $n \cdot log\, n \cdot log\ log\, n$ is such a bound.

Lemma 9.7 (*multiplication*)

There is a Type 2 machine N operating in time $O(Mb)$ with input lookahead $\leq 2n$ such that

$$\rho f_N(p,q) = \rho(p) \cdot \rho(q)$$

for all $p, q \in .\{\bar{1}, 0, 1\}^\omega$.

Proof

Consider $p = .a_1 a_2 \ldots$ and $q = .b_1 b_2 \ldots$ $(a_i, b_i \in \{\bar{1}, 0, 1\})$. N produces the output sequence $r = .c_1 c_2 \ldots$ in stages as follows.

Stage 0

Let $x_2 := \rho(.a_1 b_1 0^\omega)$, $y_2 := \rho(.b_1 b_2 0^\omega)$. Define $c_1 := \{1$ if $x_2 y_2 > 0$, 0 if $x_2 y_2 = 0$, $\bar{1}$ if $x_2 y_2 < 0\}$.

Stage n $(n \geq 1)$

Let $k := 2^n$. N multiplies the finite (generalized) binary fractions $.a_1 \ldots a_{k+2}$ and $.b_1 \ldots b_{k+2}$ and rounds the result to $.e_1 \ldots e_k$. Then, according to lemma 9.6, N improves the result $.c_1 \ldots c_{k/2}$ from Stage $n-1$ with $.e_1 \ldots e_k$ to $.c_1 \ldots c_k$.

We prove the correctness of the machine N. Define $x := \rho(p)$, $y := \rho(q)$, $x_m := \rho(.a_1 \ldots a_m 0^\omega)$, $y_m := \rho(.b_1 \ldots b_m 0^\omega)$ for $m \geq 1$.
The definition of c_1 guarantees $\rho(.a_1 a_2 \Sigma^\omega) \cdot \rho(.b_1 b_2 \Sigma^\omega) \subseteq \rho(.c_1 \Sigma^\omega)$, hence $xy \in \rho(.c_1 \Sigma^\omega)$.

Consider $n \geq 1$ and $k = 2^n$. If $.e_1 \ldots e_k$ is a rounding of $x_{k+2} \cdot y_{k+2}$, then

$$|\rho(.e_1 \ldots e_k 0^\omega) - x_{k+2} \cdot y_{k+2}| \leq 2^{-k-1}.$$

Furthermore,

$$\begin{aligned} |xy - x_{k+2}y_{k+2}| &\le |x - x_{k+2}| \cdot |y| + |x_{k+2}| \cdot |y - y_{k+2}| \\ &\le 2^{-k-2} + (1 - 2^{-k-2}) \cdot 2^{-k-2} \\ &\le 2^{-k-1} - 2^{-2k-2} \end{aligned}$$

The triangle inequality yields $|\rho(.e_1 \ldots e_k 0^\omega) - xy| < 2^{-k}$, hence $xy \in \rho(.e_1 \ldots e_k \Sigma^\omega)$. An induction with application of Lemma 9.6 shows $xy = \rho(.c_1 c_2 \ldots)$. For determining c_1, N uses the symbols a_1 and b_1, for determining the symbols c_i for $2^{n-1}+1 \le i \le 2^n$, N uses the symbols a_j and b_j with $j \le 2^n + 2$. Therefore N works with input lookahead $\le 2k$. We estimate the computation time for Stage n. Since $\rho(.a_1 \ldots a_{k+2}0^\omega)$ can be written as $2^{-k-2}(\nu_{bin}(u) - \nu_{bin}(v))$ with $lg(u), lg(v) \le k+2$, N can determine the product $x_{k+2} \cdot y_{k+2}$ in at most $c_1 Mb(k) + c_1$ steps. The other computations require at most $c_2 \cdot k + c_2$ steps. Therefore for any $m \ge 2$ the word $.c_1 \ldots c_m$ is determined by N in at most

$$s(m) := \sum \{c_1 \cdot Mb(2^i) + c_1 + c_2 \cdot 2^i + c_2 \mid i \le \lceil log\, m \rceil\}$$

steps. Since Mb is regular, $c_1 \cdot Mb(j) + c_1 + c_2 \cdot j + c_2 \in O(Mb)$ and (again by regularity of Mb) $s \in O(Mb)$.
□

By reduction to Lemma 9.7 one proves easily:

Theorem 9.8 (*multiplication*)

For every bounded subset $X \subseteq \mathbb{R}^2$ there is a Type 2 machine M which performs multiplication on X in time $O(Mb)$ with input lookahead $2n + c$ for some constant c.

The above multiplication algorithm uses a "doubling" method. The time can be bounded by $t(n) := f(2^0) + f(2^1) + \ldots + f(2^{\lceil log_2 n \rceil})$. If f is regular then $t \in O(f)$. A general case where a doubling method can be used is Newton's method for determining zeros.

By Newton's method, a zero y of a function f is determined as the limit of a sequence $(x_n)_{n \in \omega}$, where $x_{n+1} = x_n - f(x_n)/f'(x_n)$. If in some neighbourhood of y, $f'(x) \neq 0$ and $f''(x)$ is bounded, the sequence $(x_n)_{n \in \omega}$ converges "quadratically", if x_0 is sufficiently near to y_0. We consider the computation of $x \mapsto 1/x$ as a simple but important example. For $a > 0$ let $f(x) := 1/x - a$. Then $1/a$ is the zero of f. Simple computations show that $x_{n+1} = x_n(2 - ax_n)$ is the Newton recursion equation in this case and that $|x_{n+1} - 1/a| = |a| \cdot |x_n - 1/a|^2$ (quadratic convergence).

Lemma 9.9 (*inversion*)

There is a Type 2 machine M operating in time $O(Mb)$ with input lookahead $k+2$ for $k \leq 5$ and $2k-3$ for $k \geq 6$ and

$$\rho f_M(p) = 1/\rho(p)$$

for all $p \in 1.\{0,1\}^3\{\bar{1},0,1\}^\omega$.

Proof

Consider $p = 1.a_1a_2\ldots$. A Type 2 machine M produces the output sequence $r = 1.c_1c_2\ldots$ in stages according to the following rules:

Stage 0:

From $a_1 \ldots a_6$ determine $c_1 \ldots c_4$ such that
$x \in [7/8;2] \cap \rho(1.a_1 \ldots a_6\Sigma^\omega) \Longrightarrow 1/x \in \rho(1.c_1 \ldots c_4\Sigma^\omega)$.
Define $z_0 := \rho(1.c_1 \ldots c_4 0^\omega)$.

Stage n $(n \geq 1)$:

$k_n := 2^n + 3$, $r_n := \rho(1.a_1 \ldots a_{k_n+3}0^\omega)$, $y_n := z_{n-1}(2 - r_n z_{n-1})$,
$1.e_1 \ldots e_{k_n} :=$ a rounding of y_n to k_n digits,
$z_n := \rho(1.e_1 \ldots e_{k_n}0^\omega)$.

Let $1.c_1 \ldots c_{k_n}$ be the improvement of the result from Stage $n-1$ with $1.e_1 \ldots e_{k_n}$ by Lemma 9.6.

We have to prove the correctness of the machine and to make time and input lookahead estimations. By the restriction for p we have $7/8 \leq \rho(p) \leq 2$. Let $a := \rho(p)$. If $x \in [7/8;2]$, then $1/x \in \rho(1.\Sigma^\omega)$. The interval $I := \rho(1.a_1 \ldots a_6\Sigma^\omega) \cap [7/8;2]$ has length $\leq 2^{-5}$. A simple numerical calculation shows that its image J w.r.t. $x \mapsto 1/x$ has length $\leq 2^{-4}$. Therefore, digits $c_1, \ldots, c_4$ exist with $1/a \in J \subseteq \rho(1.c_1 \ldots c_4\Sigma^\omega)$. The machine M contains a finite table for determining $c_1 \ldots c_4$ from $a_1 \ldots a_6$. As a result $|z_0 - 1/a| \leq 2^{-4} = 2^{-k_0}$, where $k_0 := 2^0 + 3$. Consider $n \geq 1$ and assume that $z_{n-1} = \rho(1.e_1 \ldots e_{k_{n-1}}0^\omega)$ has been determined such that $|z_{n-1} - 1/a| \leq 2^{-k_{n-1}}$. If $x_n := z_{n-1}(2 - az_{n-1})$, then $|x_n - 1/a| \leq |a|2^{-2k_{n-1}} \leq 2^{-k_n-2}$. Since $|r_n - a| \leq 2^{-k_n-3}$ and $|z_{n-1}| < 5/4$ (since $1/a \leq 8/7$), $|x_n - y_n| \leq z_{n-1}^2 \cdot 2^{-k_n-3} \leq 2^{-k_n-2}$. By the rule for rounding, $|y_n - z_n| \leq 2^{-k_n-1}$. We obtain $|z_n - 1/a| \leq |z_n - y_n| + |y_n - x_n| + |x_n - 1/a| \leq 2^{-k_n}$.
By induction, $|\rho(1.c_1 \ldots c_{k_n}0^\omega) - 1/a| \leq 2^{-k_n}$, therefore $\rho(1.c_1c_2\ldots) = 1/a$.
We estimate the input lookahead of the machine. Simple numerical estimations show that c_1 can be determined from a_1a_2, c_1c_2 from $a_1 \ldots a_4$ and $c_1c_2c_3$ from $a_1 \ldots a_5$. Furthermore, $c_1 \ldots c_4$ is determined from $a_1 \ldots a_6$ and for $n \geq 1$ e_j for $k_{n-1}+1 \leq j \leq k_n$ is determined from $a_1 \ldots a_{k_n+3}$. From this we conclude that the input lookahead of M is $\leq k+2$ if $k \leq 5$ and $\leq 2k-3$ if $k \geq 6$. For counting input lookaheads observe that the input and the output begin with "1.". Since Mb is regular, Stage n can be

computed in $c \cdot Mb(2^n) + c$ steps. Summation yields a time bound in $O(Mb)$ for M.
□

Theorem 9.10 (*inversion*)

For every compact subset $X \subseteq \mathrm{IR}$ with $0 \notin X$ there is a Type 2 machine M which computes $x \mapsto 1/x$ on X in time $O(Mb)$ and input lookahead $2n + c$ (where c depends on X).

Proof (*outline*)
There is some $m \in \omega$ such that $2^{-m} < |x| < 2^m$ for all $x \in X$. We consider the case $x > 0$ w.l.g.. Assume $\rho(p) = x$. Then the first digit of p, which is different from 0, is 1. By at most $m + 4$ applications of the transformations $1\overline{1} \mapsto 01$, $10\overline{1} \mapsto 011$ and $100\overline{1} \mapsto 0111$ from p some $z \in \mathbb{Z}$ with $|z| \leq m$ and $q = 1.a_1a_2a_3q$ with $2^z \cdot \rho(q) = x$ and $a_1, a_2, a_3 \in \{0, 1\}$ can be determined. Some $r \in \Sigma^\omega$ with $\rho(r) = 1/\rho(q)$ can be determined in $Mb(n)$ time with lookahead $2k + c_1$ by Lemma 9.9. Finally the binary point of r is shifted by z positions.
□

As a final application we define recursiveness and computational complexity for subsets $X \subseteq \mathrm{IR}^n$. A subset $A \subseteq \omega$ is recursive, iff the characteristic function $cf_A : \omega \longrightarrow \omega$, $cf_A(x) = (1 \text{ if } x \in A, 0 \text{ otherwise})$, is computable. The direct generalization to subsets of IR, "$X \subseteq \mathrm{IR}$ is recursive, iff its characteristic function $cf_X : \mathrm{IR} \longrightarrow \omega$ is computable", is useless, since by Theorem 3.5 $cf_\emptyset$ and cf_{IR} are the only characteristic functions which are computable. If we consider ω as a metric subspace of the real line, a subset $A \subseteq \omega$, $A \neq \emptyset$, is recursive, iff the function $d_A : \omega \longrightarrow \mathrm{IR}$ is (ν_{bin}, ρ)–computable, where $d_A(x) := \min\{|x - a| | a \in A\}$. This characterization has a useful generalization.

Definition 9.11 (*complexity of compact sets*)

For any $A \subseteq \mathrm{IR}^n$, $A \neq \emptyset$ and A compact, define:

(1) $d_A : \mathrm{IR}^n \longrightarrow \mathrm{IR}$ by $d_A(x) := \inf\{|x - a| | a \in A\}$,

(2) A is recursive, iff d_A is computable,

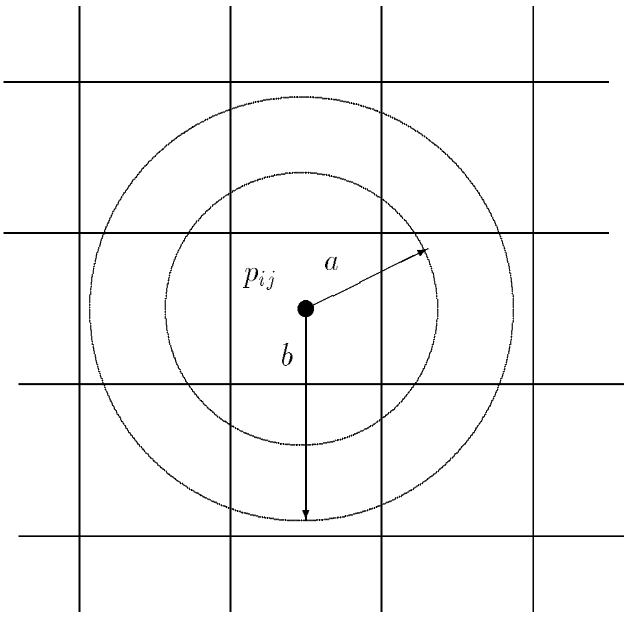
(3) A is computable in time t, iff d_A is computable in time t.

Simple subsets of IR^n such as the cube $[0; 1]^n$, the unit ball, every ball with computable centre and computable radius as well as its sphere and every convex polygon with computable vertices are computable. We mention without a proof that for

$A \subseteq \mathsf{IR}$, $A \neq \emptyset$ and A compact, A is recursive, iff A is κ–computable, where κ is the representation from Definition 5.4.

This definition of *recursive* corresponds to *located* in constructive analysis [BB 85]. The function $d_A : \mathsf{IR}^n \longrightarrow \mathsf{IR}$ of A may be called the "localizer" of A. If $n = 2$, any Type 2 machine computing d_A can be used by a plotter for producing approximate pictures of the figure $A \subseteq \mathsf{IR}^2$. Let M be some Type 2 machine computing $d_A : \mathsf{IR}^2 \longrightarrow \mathsf{IR}$ for some compact set $A \subseteq [0;1]^2$. Suppose we have a screen divided into $2^n \times 2^n$ pixels. For $i, j \in \{1, \ldots, 2^n\}$ the plotter determines the colour of the pixel $P_{ij} = ((i-1)\cdot 2^{-n}; i\cdot 2^{-n}] \times ((j-1)\cdot 2^{-n}; j\cdot 2^{-n}]$ as follows: By simulating the machine M it computes rational numbers a and b such that $d_A((i-1/2)\cdot 2^{-n}, (j-1/2)\cdot 2^{-n}) \in [a;b]$ and $b - a \leq 2^{-n-1}$. The pixel P_{ij} is set to black, if $a < 3 \cdot 2^{-n-1}$, to white otherwise. The construction guarantees:

$$\begin{aligned} & A \cap P_{ij} \neq \emptyset \\ \Longrightarrow\quad & P_{ij} \text{ is black} \\ \Longrightarrow\quad & P_{i'j'} \cap A \neq \emptyset \text{ for some } i', j' \text{ with } |i - i'| \leq 1 \text{ and } |j - j'| \leq 1. \end{aligned}$$



The pixel P_{ij} is set to black, if the annulus contains some point $x \in A$.

Consequently, the nth approximation $A_n := \bigcup\{P_{ij}$ is black$\}$ of A covers A, i.e. $A \subseteq A_n$, but it surrounds A very narrowly, since a pixel P_{ij} is white if neither the pixel itself nor any of its immediate neighbours intersect A. In fact, the Hausdorff distance $d_H(A_n, A)$ is not greater then $2 \cdot 2^{-n}$.

The kind of computational complexity of real functions introduced here is sometimes called *bit complexity*. Many interesting results on bit complexity of real functions have already been obtained, see e.g. [Bre 76, KF 82, Ko 91, Mue 86, Mue 87, Sch 90].

10 Other Approaches to Effective Analysis

The approach to computability in analysis presented in this paper (TTE) connects abstract analysis with Turing machine computability. Computability is defined explicitly on finite and infinite sequences of symbols. Computable functions turn out to be continuous. Computability and continuity are transferred to other sets by means of notations and representations where sequences serve as names of objects. Admissible representations, which formalize the concept of approximating sequences, lead to very natural computability on various sets used in analysis. The basic machine model admits to introduce realistic computational complexity in analysis.

As already mentioned there are several other approaches to study effectivity in analysis some of which are listed in the following.
Numerical analysis can be considered as the oldest discipline with this aim. Today, numerical algorithms are usually programmed (in FORTRAN, ALGOL, ...) and realized on computers. Such realizations can at most approximate the intended real functions since they operate on the finite set of floating point numbers supplied by the machines. No mathematical theory of computability or computational complexity is used.

The real RAM (*real random access machine*) is a mathematical machine model formalizing the intuitive concept of algorithm used in numerical analysis and computational geometry [BSS 89, PS 90]. Since many TTE–computable functions are not real RAM–computable, and since there are real RAM–computable functions which are absolutely not computable by a physical device (see Lemma 4.5), the real RAM model is certainly not adequate for generalizing Church's computability thesis from the natural numbers to the real numbers [Sma 92]. Non–continuous functions computable by real RAM's can be ordered by *levels* of discontinuity and classified by *degrees of discontinuity* [HW 94].

Interval Analysis controls errors which usually occur, if floating point numbers are used for performing real computations [Moo 79, Abe 88]. Although no formal definition of computability is considered, it is very closely related to a definition of computable real functions given by Grzegorczyk [Grz 57] by means of computable functions on intervals.

A computational model extending the real RAM is used in IBC (*information based complexity*) [TWW 88]. For defining computable operators, functions are inserted into programs as "black boxes" or "oracles". A typical question in IBC is: How many evaluations $f(x_i)$ are needed for determining the integral of a function $f \in C$ (for some given class C) with precision $\varepsilon > 0$?

Pour–El and Richards [PR 88] generalize a further characterization of the computable real functions (a real function is computable, iff it has a computable uniform modulus of continuity and transforms computable sequences of real numbers to computable sequences of real numbers) given by Grzegorczyk [Grz 57] to functions on

Banach spaces. They study especially solution operators of differential equations from physics.

Logical approaches are another way to formalize effectivity in analysis. The methods for proving theorems are restricted to "constructive" ones, especially no indirect proofs are allowed (see [Bee 85, BR 87, Tro 92] for detailed discussions and further references). A very far advanced theory is Bishop's remarkable constructive analysis [Bis 67, BB 85]. Most of his concepts can be transferred to TTE, if sets are interpreted by (adequate) naming systems and routines by computable or continuous functions. It should however be mentioned that such logical approaches do not admit to define computational complexity.

Computational complexity in analysis has been investigated in different ways. While in the real RAM model and in the IBC approach one evaluation of a real function is considered as a single step, the "bit complexity" models count the number of Turing machine operations for approximating a result with a given error 2^{-k} [Bre 76, KF 82, Mue 86, Mue 87, Sch 90, Ko 91, Wei 91]. TTE embeds these definitions into a general frame.

Computable analysis based on Grzegorczyk's definition via operators is sometimes called the *Polish approach.* There is another definition introduced by Ceitin [Cei 59, Kus 85, Abe 80] called *Russian approach.* The Russian approach considers only computable real numbers. Computability is introduced by an "effective" notation. We explain this more precisely in terms of TTE. For any $w \in \Sigma^*$ let $\xi_w^{*\omega}$ be the function $f :\subseteq \Sigma^* \longrightarrow \Sigma^\omega$ computed by the Type 2 machine with program w (see Appendix B). For any representation $\delta :\subseteq \Sigma^\omega \longrightarrow M$ of a set M we derive a notation $\nu_\delta :\subseteq \Sigma^* \longrightarrow M_\delta$ of the set M_δ of the δ–computable elements of M (Def. 2.9(1)) by

$$\nu_\delta(w) := \delta(\xi_w^{*\omega}(\varepsilon)).$$

We may say that $\nu_\delta(w)$ is the element $x \in M_\delta$, computed by the program w relative to the representation δ.
Let $\rho :\subseteq \Sigma^\omega \longrightarrow \mathbb{R}$ be the representation from Def. 9.1. Then $\mathbb{R}_\rho$ is the set of computable real numbers. In the Russian approach, a function $f :\subseteq \mathbb{R}_\rho \longrightarrow \mathbb{R}_\rho$ is called computable, iff it is (ν_ρ, ν_ρ)–computable. Correspondingly, computability is introduced on other sets like the r.e. subsets of $\mathbb{R}$ and the computable elements of $C[0;1]$. The underlying representations are not defined explicitly but used implicitly.

We discuss the relation between the Polish und the Russian approach. Let $\delta :\subseteq \Sigma^\omega \longrightarrow M$ be a representation and $\nu_\delta :\subseteq \Sigma^* \longrightarrow M_\delta$ the derived notation. With each function $f :\subseteq M_\delta \longrightarrow M_\delta$ we can associate a function $\overline{f} :\subseteq M \longrightarrow M$ by $graph(\overline{f}) := graph(f)$.

The Russian and the Polish approach would be (essentially) equivalent, if

$$f \text{ is } (\nu_\delta, \nu_\delta)\text{–computable} \iff \overline{f} \text{ is } (\delta, \delta)\text{–computable.}$$

The implication " $\Longleftarrow$ " can be proved easily. The implication " $\Longrightarrow$ " does not hold in general, but for some important special cases:

Theorem (*Ceitin*)

Let $f :\subseteq \mathrm{IR}_\rho \longrightarrow \mathrm{IR}_\rho$ be a function such that

$(*)$ $\nu_\rho(X)$is dense in $dom(f)$ for some r.e. set $X \subseteq dom(\nu_\rho)$.

Then

$$f \text{ is } (\nu_\rho, \nu_\rho)\text{–computable} \iff \overline{f} \text{ is } (\rho, \rho)\text{–computable.}$$

Remember that (ρ, ρ)–computability implies continuity. The condition $(*)$ cannot be omitted but might be weakened. The theorem can be generalized to computable metric spaces with Cauchy representation [Cei 59, KLS 59, Mos 64, Wei 87].
The other case is the Myhill/Shepherdson theorem [MS 55]. We formulate it in the framework of TTE. Let $PF := \{h \mid h :\subseteq \omega \longrightarrow \omega\}$ be the set of all partial number functions. Define a representation $\delta :\subseteq \Sigma^\omega \longrightarrow PF$ of PF by $\delta(p) = h$ iff p enumerates the graph of h (more precisely, $110^{i+1}10^{j+1}11$ is a subword of $p \iff h(i) = j$). Notice that PF_δ is the set $P^{(1)}$ of the partial recursive functions and ν_δ is equivalent to φ, the standard numbering of $P^{(1)}$.

Theorem (*Myhill/Shepherdson*)

For any total function $f : P^{(1)} \longrightarrow P^{(1)}$:

$$f \text{ is } (\nu_\delta, \nu_\delta)\text{–computable} \iff \overline{f} \text{ is } (\delta, \delta)\text{–computable.}$$

The theorem can be generalized to computable CPO's (cf. [Wei 87]).
Seemingly, no other cases, in which " $\Longrightarrow$ " holds, are known. Therefore the relation between the Polish and the Russian approach to computable analysis is not yet fully understood. It is well-known from computability theory that for notations like $\xi^{*\omega}$ the smn–function is easily computable (at most in polynomial time). As a consequence, for each (ν_δ, ν_δ)–computable function $f :\subseteq M_\delta \longrightarrow M_\delta$ there is some easily computable function $g \subseteq \Sigma^* \longrightarrow \Sigma^*$ with $f\nu_\delta(w) = \nu_\delta g(w)$ for all $w \in dom(f\nu_\delta)$. Therefore, the Russian approach has no complexity theory.

The references given in this paper, especially in Chapter 10, are by no means complete. Many other authors have contributed considerably to the development of effective analysis. I apologize to all those whom I did not mention.

Appendix A (*Type 2 machines and their semantics*)

A *Type 2 machine* M is defined by:

(i) an *input/output alphabet* Σ and a *tape alphabet* Γ with $\Sigma \subseteq \Gamma$ and $B \in \Gamma \setminus \Sigma$,

(ii) a sequence $(Y_1, \ldots, Y_k, Y_0)$ with $\{Y_0, \ldots, Y_k\} \subseteq \{\Sigma^*, \Sigma^\omega\}$ (specifying the function type $f_M :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$),

(iii) finitely many *Turing tapes*, each with a read/write head, indexed by $0, 1, \ldots, n$ $(k \leq n)$,

(iv) a finite *flowchart* F with the properties given below.

Only the following statements are admitted in a flowchart F of a Type 2 machine (where $0 \leq i \leq n$ and $a \in \Gamma$).

- (i, R) (move the head on Tape i one position to the right),
- (i, L) (move the head on Tape i one position to the left),
- (i, a) (write a on the square scanned by the head on Tape i),
- (i, a)? (binary branching: is a the symbol on the square scanned by the head on Tape i?),
- HALT.

Additionally, for Tapes $i \in \{1, \ldots, k\}$ (the input tapes) only statements (i, a)? and (i, R) (*read only one–way input*) and for Tape 0 (the output tape) only statement sequences $(0, a)(0, R)$ with $a \in \Sigma$ (*write only one–way output*) are admitted.

The *semantics* of a Type 2 machine is defined via computation sequences of configurations. As for ordinary Turing machines, a *configuration* of the Type 2 machine M is determined by the label of the statement in the flowchart F to be executed next and the inscription and head position for each Tape i $(0 \leq i \leq n)$. A configuration K' is the *successor* of a configuration K, $K \vdash K'$, iff K' is obtained from K by executing the statement at the label of K and going to the next label. Let the *output inscription* of a configuration, *out* (K), be the longest word $w \in \Sigma^*$ immediately to the left of the head on Tape 0. K is a *final configuration*, iff its label has the statement HALT. A *computation* is a finite or infinite sequence $K_0, K_1, \ldots$ of configurations with $K_i \vdash K_{i+1}$ $(i = 0, 1, \ldots)$.
Now, we define the function $f_M :\subseteq Y_1 \times \ldots \times Y_k \longrightarrow Y_0$ computed by the Type 2 machine M.
Consider $(y_1, \ldots, y_k) \in Y_1 \times \ldots \times Y_k$.
The *initial configuration* $K(y_1, \ldots, y_k)$ is determined as follows:

- The label is the initial label of the flowchart.
- Tape m $(1 \leq m \leq k)$ has the inscription y_m, the remaining squares have the inscription B and the head is positioned on the first square to the left of the inscription y_m.
- All the squares on the remaining tapes have the inscription B.

Case $Y_0 = \Sigma^*$:

For $w \in \Sigma^*$ we define:
$f_M(y_1, \ldots, y_k) = w$, iff there is a finite computation $K_0, K_1, \ldots, K_t$ such that $K_0 = K(y_1, \ldots, y_k)$, K_t is a final configuration and $w = \mathit{out}\,(K)$.

Case $Y_0 = \Sigma^\omega$:

For $p \in \Sigma^\omega$ we define:
$f_M(y_1, \ldots, y_k) = p$, iff there is an infinite computation $K_0, K_1, \ldots$ such that $K_0 = K(y_1, \ldots, y_k)$, $\mathit{out}\,(K_i)$ is a prefix of p for all $i \in \omega$ and the sequence $(\mathit{length\ out}\,(K_i))_{i\in\omega}$ is unbounded.

Appendix B (*Effective naming systems of sets of functions*)

First we introduce pairing functions, which are a useful tool also in Type 2 computability.

Definition B1

(1) For $k \in \omega$ and $x = a_1 \ldots a_k$ $(a_1, \ldots, a_k \in \Sigma)$ define

$$\tilde{x} := a_1 0 a_2 0 \ldots a_k 0.$$

(2) For $x, y \in \Sigma^*$ and $p, q \in \Sigma^\omega$ define

$$\begin{aligned} < x, y > &:= \tilde{x} 11 \tilde{y} \in \Sigma^* \\ < x, p > &:= < p, x > := \tilde{x} 11 p \in \Sigma^\omega \\ < p, q > &:= p(0)q(0)p(1)q(1)\ldots \end{aligned}$$

(3) For $k \geq 3$ and $z_1, \ldots, z_k \in \Sigma^* \cup \Sigma^\omega$ define

$$< z_1, \ldots, z_k > := << z_1, \ldots, z_{k-1} >, z_k > .$$

The above tuple functions are injective and computable, and the projections of their inverses are computable. As a generalization of the “effective Gödel numbering” $\varphi : \omega \longrightarrow P^{(1)}$ [Rog 67, Wei 87] we introduce notations $\xi^{ab} : \Sigma^* \longrightarrow P^{ab}$ $(a, b \in \{*, \omega\})$.

Definition B2

(1) Let $\nu_{FD} : \Sigma^* \longrightarrow FD$ be some standard notation of all flowcharts of Type 2 machines with one input tape.

(2) For $a, b \in \{*, \omega\}$ let P^{ab} be the set of all computable functions $f :\subseteq \Sigma^a \longrightarrow \Sigma^b$.

(3) For $a, b \in \{*, \omega\}$ define the notation $\xi^{ab} : \Sigma^* \longrightarrow P^{ab}$ by: $\xi^{ab}(x)$ is the function $f :\subseteq \Sigma^a \longrightarrow \Sigma^b$ computed by the flowchart $\nu_{FD}(x)$.

The representations ξ^{ab} have a computable universal function and satisfy the "smn–theorem". This can be expressed as follows:

Theorem B3

Consider $\nu :\subseteq \Sigma^* \longrightarrow P^{ab}$. Then

$$utm(\nu) \iff \nu \leq \xi^{ab},$$

where $utm(\nu)$ holds, iff there is a computable function $u :\subseteq \Sigma^* \times \Sigma^a \longrightarrow \Sigma^b$ such that $u(x, y) = \nu(x)(y)$ for all $x \in dom(\nu)$ and $y \in \Sigma^a$.

Theorem 3 expresses the kind of effectivity of the notations ξ^{ab}. For continuous functions effective representations can be introduced. A subset of a topological space is called a G_δ–set iff it is a countable intersection of open sets.

Definition B4

$$\begin{aligned}
F^{**} &:= \{f \mid f :\subseteq \Sigma^* \longrightarrow \Sigma^*\} \\
F^{*\omega} &:= \{f \mid f :\subseteq \Sigma^* \longrightarrow \Sigma^\omega\} \\
F^{\omega *} &:= \{f \mid f :\subseteq \Sigma^\omega \longrightarrow \Sigma^*,\ f \text{ is continuous and } dom(f) \text{ is open}\} \\
F^{\omega\omega} &:= \{f \mid f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega,\ f \text{ is continuous and } dom(f) \text{ is } G_\delta\}
\end{aligned}$$

F^{*b} ($b \in \{*, \omega\}$) is the set of all continuous functions $f :\subseteq \Sigma^* \longrightarrow \Sigma^b$. The sets $F^{\omega *}$ and $F^{\omega\omega}$ represent all continuous functions by the following lemma.

Lemma B5

Every continuous function $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^*$ has an extension in $F^{\omega *}$. Every continuous function $f :\subseteq \Sigma^\omega \longrightarrow \Sigma^\omega$ has an extension in $F^{\omega\omega}$.

We define representations of the function sets F^{ab}.

Definition B6

For $a, b \in \{*, \omega\}$ define $\eta^{ab} : \Sigma^\omega \longrightarrow F^{ab}$ by

$$\eta^{ab}(q)(y) := \begin{cases} \xi^{\omega b}(x) < p, y > & \text{if } q =< x, p > \text{ with } x \in \Sigma^* \text{ and } p \in \Sigma^\omega \\ div & \text{otherwise.} \end{cases}$$

The functions $\eta^{ab} : \Sigma^\omega \longrightarrow F^{ab}$ are in fact surjective and satisfy the following effectivity theorem.

Theorem B7

Consider $\delta :\subseteq \Sigma^\omega \longrightarrow F^{ab}$. Then

$$utm(\delta) \iff \delta \leq \eta^{ab},$$

where $utm(\delta)$ holds, iff there is a computable function $u :\subseteq \Sigma^\omega \times \Sigma^a \longrightarrow \Sigma^b$ such that $u(x, y) = \delta(x)(y)$ for all $x \in dom(\delta)$ and $y \in \Sigma^a$.

More details and proofs can be found in [Wei 94].

Appendix C (*Notations of ω and* $\mathbb{Q}$)

Definition C1 (*the notations ν_{bin} of ω and ν_Q of* $\mathbb{Q}$)

(1) The notation $\nu_{bin} :\subseteq \Sigma^* \longrightarrow \omega$ of w is defined by $dom(\nu_{bin}) := \{0\} \cup 1\{0,1\}^*$ and $\nu_{bin}(a_k \dots a_0) = a_k \cdot 2^k + \dots + a_0 \cdot 2^0$.

(2) The notation $\nu_Q :\subseteq \Sigma^* \longrightarrow \mathbb{Q}$ of the rational numbers is defined by

$$\nu_Q(u) := \nu_{bin}(u)$$

for all $u \in dom(\nu_{bin})$,

$$\nu_Q(\text{“}-u\text{”}) := -\nu_{bin}(u)$$

for all $u \in dom(\nu_{bin}) \setminus \{0\}$,

$$\begin{aligned} \nu_Q(\text{“}u/v\text{”}) &:= \nu_{bin}(u)/\nu_{bin}(v), \\ \nu_Q(\text{“}-u/v\text{”}) &:= -\nu_{bin}(u)/\nu_{bin}(v) \end{aligned}$$

for all $u, v \in dom(\nu_{bin})$ with $u \neq 0$, $v \notin \{0,1\}$ such that $\nu_{bin}(u)$ and $\nu_{bin}(v)$ have no common divisor. $\nu_Q(u)$ is undefined for all other $u \in \Sigma^*$.

We shall write $\overline{u}$ instead of $\nu_Q(u)$ for all $u \in don(\nu_Q)$.

A reasonable notation of the set ω of the natural numbers should at least have an r.e. domain, and the test “$n = 0$?” and upwards and downwards counting should be computable on names. The class of these notations ordered under reducibility has a maximum, the notation ν_{bin}.

Lemma C1 (*effectivity of* ν_{bin})

For all notations $\nu :\subseteq \Sigma^* \longrightarrow \omega$ of ω such that $dom(\nu)$ is r.e. we have:

$$\nu \leq \nu_{bin} \iff \{w \mid \nu(w) = 0\} \text{ and } \{(u,v) \mid \nu(u) + 1 = \nu(v)\} \text{ are r.e. .}$$

Roughly speaking, ν_{bin} is the, except for equivalence unique, poorest notation of ω with r.e. domain, for which the zero–test and counting are computable. The proof is not difficult, we omit it. Also the notation ν_Q can be characterized by an effectivity requirement and maximality.

Lemma C2 (*effectivity of* ν_Q)

For all notations $\nu :\subseteq \Sigma^* \longrightarrow \mathbb{Q}$ of the rational numbers $\mathbb{Q}$ we have:

$$\begin{aligned} \nu \leq \nu_Q \iff \quad & \{(u,v,w,x) \mid \nu(u) \cdot \nu_{bin}(v) = \nu_{bin}(w) - \nu_{bin}(x)\} \\ & \text{is r.e. in } dom(\nu) \times \Sigma^* \times \Sigma^* \times \Sigma^*. \end{aligned}$$

The proof is very easy.

Appendix D (*Admissible representations*)

We introduce a class of very natural representations, called *admissible*. Let M be a set and let $\sigma \subseteq 2^M$ be a set of subsets of M. We say that σ identifies the points of M, iff $M = \cup\sigma$ and $\{Q \in \sigma \mid x \in Q\} = \{Q \in \sigma \mid y \in Q\} \Longrightarrow x = y$ for all $x, y \in M$. (That means, each $x \in M$ can be identified by those properties $Q \in \sigma$ which hold for x.)

Definition D1

Let M be a set and let $\nu :\subseteq \Sigma^* \longrightarrow \sigma$ be a notation of a set $\sigma \subseteq 2^M$, which identifies the points of M. The *standard representation* $\delta_\nu :\subseteq \Sigma^\omega \longrightarrow M$ of M derived from ν is defined by

$$\delta_\nu(p) = x \text{ iff } \{w \mid x \in \nu(w)\} = Enw(p)$$

for all $p \in \Sigma^\omega$ and $x \in M$, where

$$Enw(p) := \{a_1 \dots a_k \in \Sigma^* \mid 110a_1 0 \dots 0a_k 11 \text{ is a subword of } p\}$$

Thus, $Enw(p)$ is the set of all words $w \in \Sigma^*$ enumerated by $p \in \Sigma^\omega$, and p is a δ_ν–name of x, iff p enumerates the set of all words w with $x \in \nu(w)$. Roughly speaking, a name of x is a complete list (in arbitrary order, possibly with repetitions) of those properties $Q \in \sigma$ which hold for x. We illustrate the definition by examples.

Example 1

(1) $M := \mathbb{R}$, $x \in \nu(w) :\Longleftrightarrow \overline{w} = \nu_Q(w) < x$,

(2) $M := \mathbb{R}$, $x \in \nu(w) :\Longleftrightarrow (w = \text{“}u\sharp v\text{”} \text{ with } \overline{u} < x < \overline{v})$,

(3) $M := 2^\omega$, $A \in \nu(w) :\Longleftrightarrow \nu_{bin}(w) \in A$,

(4) $M := \tau_{\mathbb{R}} :=$ the set of open subsets of $\mathbb{R}$,
$O \in \nu(w) :\Longleftrightarrow (w = u¢v \text{ with } [\overline{u}; \overline{v}] \subseteq O)$,

(5) $M = \Sigma^\omega$, $p \in \nu(w) :\Longleftrightarrow w$ is a prefix of p.

Every set $\sigma \subseteq 2^M$, which identifies the points of M, is a subbase of a T_0–topology τ on M (Engelking [Eng 89]), and any subbase of a T_0–topology on M identifies points on M. The topology $\tau \subseteq 2^M$ is defined from σ by:

$$\tau := \{\bigcup \alpha \mid \alpha \subseteq \beta\}$$

where

$$\beta := \{Q_1 \cap \ldots \cap Q_n \mid n \geq 1, Q_1, \ldots, Q_n \in \sigma\}$$

is a *base* of the topology τ. In Example 1(2), τ is the usual topology $\tau_{\mathbb{R}}$ of the real line, in Example 1(5), τ is the Cantor topology on Σ^ω. The representation δ_ν and the topology τ generated by σ as a subbase are very closely related:

Theorem D2

Let τ be the topology on M generated by the subbase $\sigma = range(\nu)$ from Definition D1 . Then

(1) $X \in \tau \iff \delta_\nu^{-1}X$ is open in $dom(\delta_\nu)$ for all $X \subseteq M$.

(2) δ is continuous $\iff \delta \leq_t \delta_\nu$ (for all functions $\delta :\subseteq \Sigma^\omega \longrightarrow M$).

By (1), τ is the *final topology* of δ_ν, by (2), δ_ν is the "greatest" or "poorest" (except for equivalence) continuous representation of the T_0–space (M, τ). In (2), " $\Longrightarrow$ " corresponds to the smn–theorem and " $\Longleftarrow$ " to the utm–theorem from ordinary recursion theory. Theorems 4.1, 5.2(1) and 6.2(1) are special cases of Theorem D2(2). We call representations which are t–equivalent to some standard representation *admissible* w.r.t. τ or τ*–admissible*.

Definition D3

Let (M, τ) be a topological T_0–space with denumerable subbase. A representation $\delta :\subseteq \Sigma^\omega \longrightarrow M$ of M is called τ*–admissible*, iff

$$\delta' \text{ is continuous } \iff \delta' \leq_t \delta$$

for all functions $\delta' :\subseteq \Sigma^\omega \longrightarrow M$.

By Theorem D2, every T_0–space (M, τ) with denumerable base has a τ–admissible representation which is unique except for t–equivalence. In Example 1(2) we obtain $\delta_\nu \equiv \rho_C$, hence ρ_C is $\tau_{\mathbb{R}}$–admissible; In Example 1(5) we obtain $\delta_\nu \equiv id_{\Sigma^\omega}$, hence id_{Σ^ω} is τ_C–admissible. Let τ be the topology induced by the metric on a separable metric space (M, d). Then (M, τ) is a T_0–space with denumerable subbase which has a τ–admissible representation. The Cauchy representation (for examples see Def. 3.1 and Def. 6.5) is τ–admissible. More details can be found in [Wei 87, Wei 94].
For spaces with admissible representations, a function is (topologically) continuous, iff it is continuous w.r.t. the representations (see Def. 2.10(2)). This is stated in Theorem 4.3.

References

Abe 80 Aberth, O.:
Computable analysis, McGraw–Hill, New York, 1980

Abe 88 Aberth, Oliver:
Precise numerical analysis, Brown Publishers, Dubuque, Iowa, 1988

BB 85 Bishop,E.; Bridges, D.S.:
Constructive Analysis, Springer-Verlag, Berlin, Heidelberg, 1985

Bee 85 Beeson, M.J.:
Foundations of constructive mathematics, Springer–Verlag, Berlin, Heidelberg, 1985

Bis 67 Bishop, Errett:
Foundations of constructive analysis, McGraw–Hill, New York, 1967

Bre 76 Brent, R.P.:
Fast multiple precision evaluation of elementary functions, J. ACM 23, 242 - 251 (1976)

BR 87 Bridges, D.S.: Richman, F.:
Varieties of constructive mathemetics, Cambridge University Press, Cambridge, 1987

Bri 94 Bridges, Douglas:
Computability, Springer Verlag, New York, Berlin, 1994

BSS 89 Blum, L.; Shub, M.; Smale, S.:
On a theory of computation and complexity over the real numbers, Bull. Amer. Math. Soc. 21, 1 - 46, 1989

Cei 59 Ceitin, G.S.:
Algorithmic operators in constructive complete separable metric spaces (in Russian), Doklady Akad, Nauk 128, 49 - 52 (1959)

Eng 89 Engelking, Ryszard.:
General topology, Heldermann, Berlin, 1989

FS 74 Fischer, M.J.; Stockmeyer, L.:
Fast On–line integer multiplication, Journal of Computer and System Sciences 9, 317 - 331, 1974

Grz 55 Grzegorczyk, A.:
Computable functionals, Fundamenta mathematica 42, 168 - 202, (1955)

Grz 57 Grzegorczyk, A.:
On the definition of computable real continuous functions, Fund. Math. 44, 61 - 71 (1957)

Hau 73 Hauck, J.:
Berechenbare reelle Funktionen, Zeitschrift f. math. Logik und Grdl. Math. 19, 121 - 140 (1973)

Hau 76 Hauck, J.:
Berechenbare reelle Funktionenfolgen, Zeitschrift f. math. Logik und Grdl. Math. 22, 265 - 282 (1976)

HU 79 Hopcropft, John E.; Ullman, Jeffrey D.:
Introduction to automata theory, languages, and computation, Addison–Wesley, Reading, MA, 1979

HW 94 Hertling, P., Weihrauch, K.:
Levels of degeneracy and exact lower bounds for geometric algorithms, Proceedings of the Sixth Canadian Conference on Computational Geometry, 237 - 242, Saskatoon, 1994

Kla 61 Klaua, D.:
Konstruktive Analysis, Deutscher Verlag der Wissenschaften, Berlin, 1961

KF 82 Ko, K. Friedman, H.:
Computational complexity of real functions, Theoretical Computer Science 20, 323 - 352, 1982

KLS 59 Kreisel, G.; Lacombe, D.; Shoenfield, J.:
Partial recursive functionals and effective operations. Constructivity in Mathematics (A. Heyting, ed.), 195 - 207, North-Holland, Amsterdam, 1959

Ko 91 Ko, Ker–I:
Complexity theory of real functions, Birkhäuser, Bosten, Basel, Berlin, 1991

Kus 85 Kushner, B.A.:
Lectures on constructive mathematical analysis, American Mathematical Society, Providence, 1985 (English translation from Russian original, 1973)

KW 85 Kreitz,Ch.; Weihrauch,K.:
Theory of representations, Theoretical Computer Science 38, 35 - 53 (1985)

KW 87 Kreitz,Ch.; Weihrauch,K.:
Compactness in constructive analysis revisited, Informatik–Berichte, Fernuniversität Hagen (1984) and Annals of Pure and Applied Logic 36, 29 –38, 1987

Moo 79 Moore, Ramon E.:
Methods and applications of interval analysis, SIAM, Philadelphia, 1979

Mos 64 Moschavakis, Y.N.:
Recursive metric spaces, Fundamenta mathematicae LV, 215 - 238, (1964)

MS 55 Myhill, J. Shepherdson, J.C.:
Effective operations on partial recursive functions, Zeitschrift für mathamatische Logik und Grundlangen der Mathematik 1, 310 - 317 (1955)

Mue 86 Müller,N.Th.:
Subpolynomial complexity classes of real functions and real numbers. In: Lecture notes in Computer Science 226, Springer–Verlag, Berlin, Heidelberg, 284 - 293 (1986)

Mue 87 Müller, N.Th.:
Uniform computational complexity of Taylor series. In: Lecture notes in Computer Science 267,, Springer–Verlag, Berlin, Heidelberg, 435 - 444 (1987)

Odi 89 Odifreddi, Piergiorgio:
Classical recursion theory, North–Holland, Amsterdam, 1989

PR 88 Pour–El, Marian B.;Richards, Jonathan I.:
Computablility in Analysis and Physics, Springer–Verlag, Berlin, Heidelberg, 1988

PS 90 Preparata, F.P.; Shamos, I.S.:
Computational Geometry, Springer–Verlag, New York, Berlin, Heidelberg, 1990

Rog 67 Rogers, Hartley Jr.:
Theory of recursive funcions and effective computability, McGraw–Hill, New York, 1967

Sch 71 Schönhage, Arnold:
Schnelle Multiplikation großer Zahlen, Computing 7, 281 - 292, 1971

Sch 90 Schönhage, Arnold:
Numerik analytischer Funktionen und Komplexität, Jahresberichte der Deutschen Mathematiker–Vereinigung 92, 1 - 20, 1990

Sma 92	Smale, Stephen: Theory of computation, in: C. Cassacuberta et al. (eds.), Mathematical research today and tomorrow, Springer–Verlag, Berlin 1992
Spe 49	Specker, Ernst: Nicht konstruktiv beweisbare Sätze der Analysis, The Journal of Symbolic Logic 14.3, 145 - 158, 1949
Spe 59	Specker, Ernst: DerSatz vom Maximum in der Rekursiven Analysis, in: A. Heyting (ed.) Constructivity in Mathematics, North–Holland, Amsterdam, 1959
Tro 92	Troelstra, A.S.: Comparing the theory of representations and constructive mathematics, in: proceedings of the conference Computer Sicence Logic '91, Lecture Notes in Computer Science, Springer–Verlag, Berlin, 1992
TWW 88	Traub, J.F.; Wasilkowski, G.W.: Wozniakowski, H.: Information–based Complexity, Academic press, New York, 1988
Wei 85	Weihrauch, Klaus: Type 2 recursion theory, Theoretical Computer Science 38, 17 - 33 (1985)
Wei 87	Weihrauch, Klaus: Computability, Springer–Verlag, Berlin, Heidelberg, 1987
Wei 91	Weihrauch, Klaus: On the complexity of online computations of real functions, Journal of Complexity 7, 380 – 394, 1991
Wei 92A	Weihrauch, Klaus: The degrees of Discontinuity of some translators between representations of the real numbers, Informatik–Berichte Nr. 129, Fernuniversität 1992
Wei 92B	Weihrauch, Klaus: The TTE–Interpretation of three Hierarchies of Omniscience principles, Informatik–Berichte Nr. 130, Fernuniversität 1992
Wei 93	Weihrauch, Klaus: Computabilitiy on computable metric spaces, Theoretical Computer Science 113, 191 - 210, 1993
Wei 94	Weihrauch, Klaus: Grundlagen der effektiven Analysis, correspondence course, FernUniversität, Hagen, 1994
WK 84	Weihrauch,K.; Kreitz, Ch.: A unified approach to constructive and recursive analysis. In: Computation and proof theory (M.M. Richter et al., eds.), Springer–Verlag, Berlin, Heidelberg, 1984
WK 87	Weihrauch,K.; Kreitz, Ch.: Representations of the real numbers and of the open subsets of the set of real numbers, Annals of Pure and Applied Logic 35, 247 – 260, 1987
WK 91	Weihrauch,K.; Kreitz, Ch.: Type 2 computational complexity of functions on Cantor's space , Theoretical Computer Science 82, 1 – 18 (1991)
Wie 80	Wiedmer, E.: Computing with infinite objects, Theoretical Computer Science 10, 133 - 155 (1980)
WW 86	Wagner, Klaus; Wechsung, Gerd: Computational complexity, D. Reidel, Dordrecht, 1986