Electronic Colloquium on Computational Complexity - Reports Series 1995 - available via:

ECCC TR95-025

FTP: ftp.eccc.uni-trier.de:/pub/eccc/

WWW: http://www.eccc.uni-trier.de/eccc/

Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

Analytic Machines

Günter Hotz, Gero Vierke and Björn Schieffer

Fachbereich 14 – Informatik
Universiät des Saarlandes
Postfach 15 11 50
66041 Saarbrücken

Abstract

In this paper the R-machines defined by Blum, Shub and Smale are generalized by allowing infinite convergent computations. The description of real numbers is infinite. Therefore, considering arithmetic operations on real numbers should also imply infinite computations on *analytic machines*. We prove that $\mathbb{R}$-computable functions are $\mathbb{Q}$-analytic. We show that R-machines extended by finite sets of *strong analytic* operations are still $\mathbb{Q}$-analytic. The halting problem of the analytic machines contains the stability problem of dynamic systems. It follows with well known methods that this problem is not analytical decidable. This is in a sense a stronger result as the *numerical undecidable* stability in the theory of Kolmogoroff, Arnold and Moser.

Keywords: $\mathbb{R}$-computability, stability, approximation

1 Introduction

In this paper we consider R-machines within the meaning of [B.S.S]. The memory holds elements of the ring R. The arithmetic of the machine is identical to the arithmetic in R. The memory size is assumed to be infinite. The machine may address the memory indirectly, i.e. a memory cell may be used to address others, if it holds a natural number. We assume only a finite number of memory cells to contain a nonzero value at the beginning of a computation. Only halting computations are accepted.

In this paper we restrict to machines with $R = \mathbb{Q}$ (the set of rational numbers) and $R = \mathbb{R}$ (the set of real numbers). We now extend the machine model sketched above:

- We allow infinite computations, i.e. computations without halting. Instead of halting we consider the convergence of the output stream. We call such machines analytic machines.
- With infinite computations, analytic $\mathbb{Q}$-machines are able to deal with real valued functions. To achieve that, we allow real numbers at the input tape of a $\mathbb{Q}$-machine. The machine has a rounding register containing a precision $\delta \in \mathbb{N}$, so that it can read a rational approximation $[x]_\delta \in \mathbb{Q}$ with $|x - [x]_\delta| \leq 2^{-\delta}$ instead of the real valued input x. A possible approximation function $[.] : \mathbb{R} \to \mathbb{Q}$ is defined by selecting the binary number with δ digits after the point of the infinite binary number notation.

A function that can be computed by an $\mathbb{R}$-analytic machine is called $\mathbb{R}$-analytic. A function is called $\mathbb{Q}$-analytic, if it can be computed by a $\mathbb{Q}$-analytic machine reading the input again and again with increasing precision (i.e. $2^{-\delta} \to 0$). With this machine model the following question rises: Is an $\mathbb{R}$-analytic function $\mathbb{Q}$-analytic as well, or are $\mathbb{R}$-analytic machines more powerful than $\mathbb{Q}$-analytic machines?

A continuous function $f : \mathbb{R} \to \mathbb{R}$ is completely determined by its rational restriction $f \,|\, \mathbb{Q}$. If $f \,|\, \mathbb{Q}$ is computable, then also f in a sense: Take a rational sequence $x_1, x_2, \ldots$ converging to $x \in \mathbb{R}$ and you receive an approximative computation of $f(x)$ with the resulting sequence $f(x_1), f(x_2), \ldots \to f(x)$. We will extend this theorem to all $\mathbb{R}$-computable functions by transforming the programs for the $\mathbb{R}$-machine into programs for the $\mathbb{Q}$-analytic machine. The difficulty in the proof is that $\mathbb{R}$-computable functions are not continuous in general. The representation theorem for an $\mathbb{R}$-computable function of [B.S.S] shows, that f may be given as a countable sum of continuous functions $f^\sigma : A_\sigma \to \mathbb{R}$ with semi-algebraic sets A_σ. The problem is that such a set A_σ does not necessarily contain any rational point. So we are not able to convert the procedure used for the approximation of continuous functions directly. The basic idea for the solution of this problem lies within a suitable enlargement of A_σ, in which we can find the rational points needed to approximate x. As an introduction, we will give a simple proof to a little more general version of the representation theorem.

In [B.S.S] the area of attraction for the fixed-points of the iteration of simple polynomials is considered and it is shown that the complement of these areas cannot be characterized by $\mathbb{R}$-machines. Therefore, the representation theorem together with the fact, that the number of disjoint components in the complementary set is of the power of the continuum, is used. We will show that two well known curves, namely Koch's curve and Hilbert's space filling curve, are not $\mathbb{R}$-computable but $\mathbb{Q}$-analytic. To do this, we use the representation theorem together with the Hausdorff dimension of the graph of a curve.

The set of $\mathbb{Q}$-analytic functions is shown to be a proper subset of the set of $\mathbb{R}$-analytic functions. While the set of $\mathbb{R}$-computable functions is closed under composition and primitive recursion, this is not true for $\mathbb{Q}$-analytic functions. If we require the $\mathbb{Q}$-analytic machine to give the precision of its approximation – we will call such functions *strong* $\mathbb{Q}$-*analytic* – we will get the well-known *computable real functions* within the meaning of Grzegorczyk which are closed under the named operations.

We extend the R-machines by procedure calls of strong $\mathbb{Q}$-analytic functions $f_1, f_2, \ldots f_m$ and call the resulting machine $\mathcal{M}_R(f_1, f_2, \ldots, f_m)$. We will show that functions computable by $\mathcal{M}_{\mathbb{R}}(f_1, f_2, \ldots, f_m)$ are $\mathbb{Q}$-analytic.

A different motivation of our examination is pointed out by the following consideration: computers are usually thought to be discrete machines, whereas they really change their states continuously over time. But they do this change from one stable state to another very fast. Such a change is a *call* of a continuous procedure with very high output convergence. The simulation of such a system on an R-machine contains the sequence of calls of strong $\mathbb{Q}$-analytic procedures. The very fast convergence of these procedures reduces the question of stability of the continuous system to the halting problem of R-machines. The success in using computers should give us the certainty that the conception of the $\mathcal{M}_R(f_1, f_2, \ldots, f_m)$-machines is reasonable.

For analytic machines the halting problem is equivalent to the question of convergence. Finally we will show that the question whether an analytic machine is convergent is not analytically decideable. The problem is related to the stability problem of dynamic systems. In many cases, the stability problem can be reduced to the question whether the iteration of a function S is contractile in the neighborhood of a given point. If such a function is included as a procedure into our machine, the halting problem of the analytic machine includes the stability problem of those dynamical systems for which S is $\mathbb{R}$-analytically or $\mathbb{Q}$-analytically computable. The theory of dynamic systems, associated with names like Kolmogoroff, Moser, Arnold [A.A] and Siegel [S], is able to prove stability of such systems in some special cases. But the theory shows that in general there is a very sensitive dependence on the starting point of the iteration. Therefore, the decision of the stability of such systems is thought to be uncomputable for numerical reasons. The well-known construction to prove the undecideability of the halting problem of discrete machines helps to show that even with exact $\mathbb{R}$-arithmetic the stability of the systems is not decideable in a universal way with $\mathbb{R}$-analytic or $\mathbb{Q}$-analytic machines.

2 Basic Definitions

A mathematical machine consists of a *configuration-set* K, the set of initial states $K_a \subset K$, the set of final states $K_e \subset K$ and a next state function

$$\Delta : K \to K \text{ with } \Delta | K_e = 1_{K_e}.$$

(1_M denotes the identity function on the set M.) A computation on such a machine is a sequence of states

$$b = (k_0, k_1, k_2, \ldots)$$

with

$$k_{i+1} = \Delta(k_i) \text{ for } i \geq 0.$$

We call the computation *regular*, if $k_0 \in K_a$ and there exists a n such that $k_n \in K_e$. Because of $\Delta(k_n) = k_n$ for $k_n \in K_e$ we shortly write

$$b = (K_a \ni k_0, k_1, k_2, \ldots, k_n \in K_e)$$

for a regular computation.

R^* denotes the set of all finite tuples of numbers from the ring R. C and D will be arbitrary subsets of R^*. To explain in which cases a function

$$f : D \to C$$

is called *computable* we need a simple input function $in : D \to K_a$ and a simple output function $out : K \to C$ that do not depend on the computed function f. It may be a little surprising that out is not only given for final states $k_e \in K_e$, but we will need this later on. We now define a machine

$$\mathcal{M} := (K, K_a, K_e, \Delta, in, out)$$

and call $f : D \to C$ *$\mathcal{M}$-computable*, iff the computation

$$b = (K_a \ni k_0, k_1, k_2, \ldots, k_n \in K_e)$$

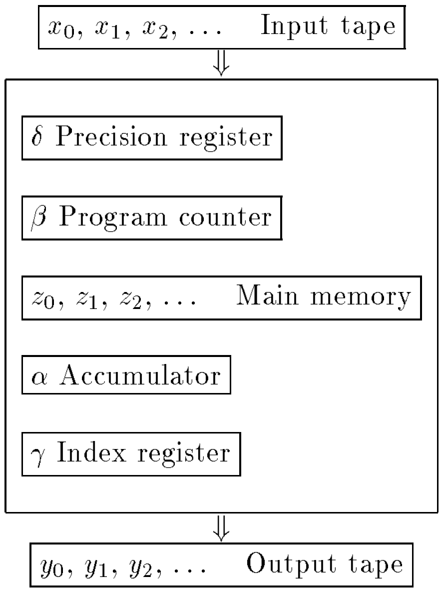


Figure 1: Graphic description of the R-machine

with the initial state $k_0 = in(x)$ is regular and $f(x) = out(k_n)$.

In section 6 we extend this idea of *computable* functions allowing infinite computations if the output function is convergent for some marked states. Therefore, we introduce the set K_t with

$$K_e \subset K_t \subset K.$$

Instead of demanding that a computation ends in a final state, we require that it reaches a state of K_t again and again.

The set K_t leads to a possibly helpful regularity in the computations. This concept is identical to the term of infinite accepting computations on finite automata, invented by Büchi.

We now turn to the aim of this paper. The configuration sets of the machines we observe are infinite products of $\mathbb{Q}$ and $\mathbb{R}$. The functions Δ are simple extensions of rational functions. We call the analytically computable functions $\mathbb{Q}$-analytic or $\mathbb{R}$-analytic, respectively. We will examine whether there is a universal method to approximate the $\mathbb{R}$-analytic functions by the $\mathbb{Q}$-analytic and we will look at the relation between the $\mathbb{Q}$-analytic and the $\mathbb{R}$-computable functions. Finally, we will examine whether it is possible to decide the halting problem of analytic machines by analytically computable functions.

3 The Concrete Machine-Model

We now define a machine-model as an abstraction of a simple computer. The R-machine is equivalent to the usual computer-models and especially to the model used in [B.S.S]. Figure 1 gives a graphic description of the machine.

The input tape is read-only memory, the output tape is write-only memory. The variables or memory cells can contain a number from R. γ and δ can contain a natural number and β a number from

$$[0:N] := \{j \in \mathbb{N} | 0 \leq j \leq N\},$$

where N is the length of the program. The δ-register is only used for analytic $\mathbb{Q}$-machines where it keeps the precision for reading the real numbers on the input tape. We will come back to this point when we introduce the set of instructions for the R-machines.

Hence we can describe a machine state by a map

$$\rho : (\alpha, \beta, \gamma, \delta, x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2, \ldots) \to R \times [0:N] \times \mathbb{N} \times \mathbb{N} \times R \times R \times \ldots$$

We require only a finite number of variables to contain a value $\neq 0$. By that the configuration set K is defined.

Let $k \in K$ be a machine state.

$$\alpha(k), \beta(k), \gamma(k), \delta(k), z_j(k), x_j(k), y_j(k).$$

denote the value of the variables $\alpha, \beta, \gamma, \delta \ldots$ belonging to the state k. If k is obvious from the context we omit the dependency on k. Now we define

$$K_a := \{k \in K \,|\, \alpha = \beta = \gamma = \delta = 0, x_0 \in \mathbb{N}, x_j = 0 \text{ for } j > x_0, z_j = 0, y_j = 0 \text{ for all } j\}$$

$$K_e := \{k \in K \,|\, \beta = N, y_0 \in \mathbb{N}, y_j = 0 \text{ for } j > y_0\}.$$

At this point we can define the functions $in : R^* \to K_a$ by

$$\begin{aligned} in((r_1, r_2, \ldots, r_n)) \;:=\; & (\alpha = \beta = \gamma = \delta = 0, x_0 = n, x_i = r_i \text{ for } 0 < i \leq n, \\ & x_j = 0 \text{ for } j > n, y_l = z_l = 0 \text{ for } l \geq 0) \end{aligned}$$

and $out : K \to R^*$ by

$$out(k) := (y_1(k), y_2(k), \ldots, y_{y_0}(k)).$$

We define the set Ω of instructions for our machine:

1. assignments:

   $\alpha := z_j,\ z_j := \alpha,\ \alpha := x_j,\ y_j := \alpha \qquad$ for $j \in \mathbb{N}$.

2. arithmetic operations:

   $\alpha := \alpha \star z_j,\ \alpha := \alpha \star r \qquad$ for $\star \in \{+, -, \times, /\}, j \in \mathbb{N}$ and $r \in R$.

3. conditionals:

   if $\alpha \sim 0$ then goto m else goto n, $\qquad m, n \in [0:N]$ and $\sim \in \{=, >\}$

4. halting:

   end

5. index operations:

   $\alpha := z(\gamma),\ z(\gamma) := \alpha,\ \alpha := x(\gamma),\ y(\gamma) := \alpha,\ \gamma := \alpha,\ \alpha := \gamma,\ \gamma := \gamma \pm 1.$

The effects of the instructions are quite self-evident. The operation *end* is only allowed to be the last instruction. We consider a program to be correct only if a division by zero is avoided for any input and if there are only natural values assigned to the index register. The programer is responsible for that; to make this easier we define an additional instruction

$$\text{if } \alpha \in \mathbb{N} \text{ then goto } m \text{ else goto } n, \qquad m, n \in [0:N]$$

which can also be interpreted as a macro, programed using only instructions from Ω. A program is given by a map

$$\pi : [0:N] \to \Omega.$$

Hence we are able to define $\Delta : K \to K$. We define for $\rho, \rho' \in K$

$$\Delta(\rho) = \rho'$$

iff the application of the instruction $\pi(\beta(\rho))$ on ρ is leading to ρ'. Every instruction changes only the value of the variables to which it assigns something, except β, the program counter. If $\pi(\beta(\rho))$ is no if-operation, every instruction causes $\beta(\rho') = \beta(\rho)+1$. Whereas the instruction

$$\text{if } \alpha \sim 0 \text{ then goto } m \text{ else goto } n, \qquad m, n \in [0:N] \text{ and } \sim \in \{=, >\},$$

causes

$$\beta(\rho') = m \text{ if } \alpha \sim 0 \text{ is satisfied,}$$

and

$$\beta(\rho') = n \text{ if } \alpha \sim 0 \text{ is not satisfied.}$$

The instruction $\alpha := z(\gamma)$ causes $\alpha := z_\gamma$. The other indexed instructions work analogously. We will define the operations to read from the input tape

$$\alpha := [x_j]_\delta \text{ and } \alpha := [x(\gamma)]_\delta$$

later on, when we introduce the analytic machines. They are used to read a rational rounding instead of a real number, where δ is the precision of the approximation.

We give an example of a program computing the absolute value $|x_1|$ of the input x_1.

1 $\alpha := x_1$;
2 if $\alpha > 0$ then goto 4 else goto 3;
3 $\alpha := \alpha \cdot (-1)$;
4 $y_1 := \alpha$;
5 $\alpha := 1$;
6 $y_0 = \alpha$;
7 end.

$\mathcal{M}_{|.|}$ denotes the R-machine with that program.

4 A Representation Theorem for R-Computable Functions

We interpret the positions of the program counter as nodes of a graph G. Let $p_1, p_2 \in G$ be nodes. There is an edge s from p_1 to p_2 iff there exists a $\rho \in K$ such that $\beta(\rho) = p_1$ and $\beta(\Delta(\rho)) = p_2$. We mark the edge s with the operation $\pi(p_1)$, which is applied on ρ and leads to $\Delta(\rho)$. If $\pi(p_1)$ is an if-operation we mark the two outgoing edges with the result of the test; e.g. with "$\alpha = 0$" or "$\alpha \neq 0$" in case of $\pi(p_1) =$"if $\alpha = 0$ then…".

We interpret the paths in this graph as branches of a tree and get the computation tree $\mathcal{L}(\mathcal{M})$ belonging to the machine $\mathcal{M}$. Hence for every regular computation we have a path from the root of the tree to a terminal node. This is basically Rabin's decision tree.

For the proof of the representation theorem we need a more sophisticated flow chart: The set of nodes is $[0:N] \times \mathbb{N}$ where $[0:N]$ is the set of possible positions of the program counter of the machine. There is an edge from the node (p_1, n_1) to (p_2, n_2) iff there exists a $\rho_1 \in K$ such that

$$\beta(\rho_1) = p_1, \quad \gamma(\rho_1) = n_1, \quad \rho_2 = \Delta(\rho_1)$$

and

$$\beta(\rho_2) = p_2, \quad \gamma(\rho_2) = n_2.$$

So we additionally use the value of the index register for the characterization of the machine state. The graph belonging to $\mathcal{M}$ is called $G_\gamma(\mathcal{M})$. As before we construct the decision tree from $G_\gamma(\mathcal{M})$ and call it $\mathcal{L}_\gamma(\mathcal{M})$. Figure 2 shows $G_\gamma(\mathcal{M}_{|.|})$ and $\mathcal{L}_\gamma(\mathcal{M}_{|.|})$.

Let $\rho \in K_a$ be an initial state of $\mathcal{M}$. $\sigma(\rho)$ denotes the computation path belonging to ρ in $\mathcal{L}_\gamma(\mathcal{M})$. If the computation belonging to ρ is regular, $\sigma(\rho)$ leads to a terminal node in $\mathcal{L}_\gamma(\mathcal{M})$ corresponding to the node N in G. Now we deduce

Lemma 1: Let $\rho_0, \bar\rho_0 \in K_a$. We define $\rho_i := \Delta^i(\rho_0)$ and $\bar\rho_i := \Delta^i(\bar\rho_0)$. If $\sigma(\rho_0) = \sigma(\bar\rho_0)$ then $\pi(\rho_i) = \pi(\bar\rho_i)$ for all i and the instructions are referring to the same memory cells and registers. Especially all results depend on the same input variables.

This can be proven by an induction over i.

V_σ denotes the set of variables (=memory cells) which are read by a regular computation following the path σ in $\mathcal{L}_\gamma(\mathcal{M})$. V_σ generally contains variables from $\alpha, \gamma, x_0, \ldots x_n, z_0, \ldots z_m$. V_σ does not contain any variables from $y_0, y_1, \ldots$, which are write-only memory cells. S_σ denotes the set of all memory cells to which a value is assigned during the computation.
We now define a rational function

$$f_s^\sigma : D \to R$$

for any regular path σ and any $s \in S_\sigma$. We define the f_s^σ inductively by the length of σ. For $|\sigma| = 0$ we define

$$f_s^\sigma(v) := 0 \text{ for all } v \in D \text{ and } s \in S_\sigma.$$

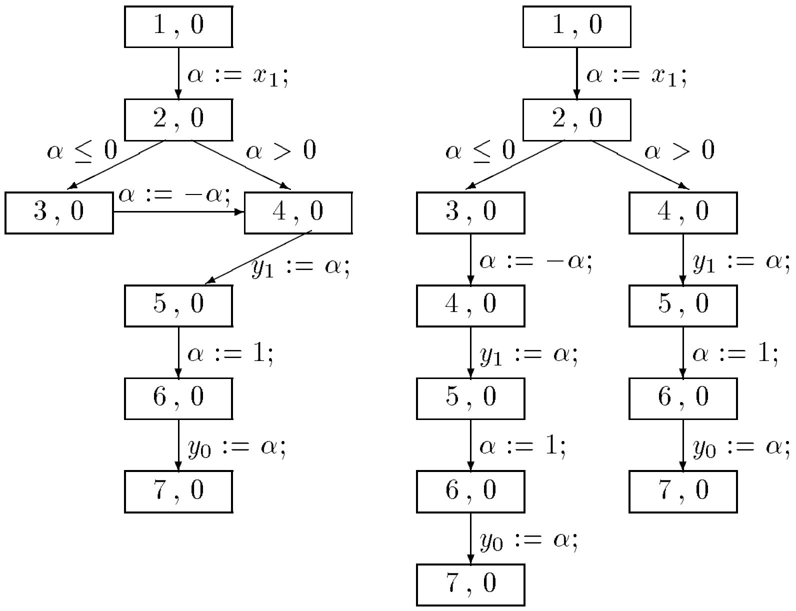


Figure 2: Computation graph and decision tree of $\mathcal{M}_{|.|}$

Let $|\sigma| = n$ and σ' be the prefix of σ such that $|\sigma'| = n-1$. If π_n is the instruction marked on the n-th edge of σ and $\star \in \{+, -, \times, /\}$, we define

$$f_s^\sigma := \begin{cases} f_s^{\sigma'} & \text{if } \pi_n \text{ does not assign a value to } s, \\ f_t^{\sigma'} & \text{if } \pi_n = (s := t), \\ f_\alpha^{\sigma'} \star f_t^{\sigma'} & \text{if } \pi_n = (s := \alpha \star t), \\ f_\alpha^{\sigma'} \star r & \text{if } \pi_n = (s := \alpha \star r), r \in R, \\ f_\gamma^{\sigma'} \pm 1 & \text{if } s = \gamma, \pi_n = (\gamma := \gamma \pm 1). \end{cases}$$

If π_n uses a indexed variable then s or t is the variable fixed by the current value of γ. It is found in the second component of the terminal node of σ'; e.g. in the case $\pi_n = (\alpha := x(\gamma))$ and $\sigma' = (\ldots, (n-1, 5))$ we have $t = x_5$ and $f_\alpha^\sigma := f_{x_5}^{\sigma'}$.

Let $\rho_0, \bar{\rho}_0 \in K_a$ such that $\sigma(\rho_0) = \sigma(\bar{\rho}_0)$. By Lemma 1 we can obtain the results of the computation produced by ρ_0 and $\bar{\rho}_0$ by the same rational function f_s^σ from the input values $(x_1, \ldots, x_m)$.

We will now characterize the input values leading to the same computation paths. Let σ be a computation path in $\mathcal{L}_\gamma(\mathcal{M})$. Σ^σ denotes the set of prefixes σ' of σ, leading to if-operations or to an operation "$\gamma := \ldots$".

We see that for all $\sigma' \in \Sigma^\sigma$ the (in)equations

$$f_\alpha^{\sigma'}(x_1, \ldots, x_m) = 0, \quad \neq 0, \quad > 0, \quad \leq 0$$

are fulfilled by the rational functions $f_\alpha^{\sigma'}$ iff the corresponding condition in σ holds. Likewise $f_\gamma^{\sigma'}(x_1, \ldots, x_m) = k$, if $\sigma' = (\ldots, (n, k))$ holds. $F^{\sigma'}(x_1, x_2, \ldots, x_m)$ denotes these (in)equations. It determines the path σ. Hence

$$A_\sigma := \{(x_1, \ldots, x_m) \in D \mid F^{\sigma'}(x_1, \ldots, x_m) \text{ is satisfied for all } \sigma' \in \Sigma^\sigma\}$$

is the set of the solutions of the algebraic requirements belonging to σ.

We demanded for our programs that the division by zero has to be avoided; therefore, the denominators of the rational functions have no zeros. By multiplying the rational equations with their denominators we get a system $C_\sigma(x_1, \ldots, x_m)$ of polynomial (in)equations which determines A_σ. Hence A_σ is a semi-algebraic set. $a_\sigma : R^* \to \{0, 1\}$ shall be the characteristic function of $A_\sigma \subset R^*$. Let $S_\mathcal{M}$ be the set of terminating computation paths in $\mathcal{L}_\gamma(\mathcal{M})$. Concludingly, we sum up our result:

Representation Theorem: For an R-computable function $f : D \to R^*$ holds

$$\begin{array}{ll} (1) & D = \bigcup\limits_{\sigma \in S_\mathcal{M}} A_\sigma \\ (2) & f = \sum\limits_{\sigma \in S_\mathcal{M}} a_\sigma \cdot (f^\sigma_{y_1}, f^\sigma_{y_2}, \ldots, f^\sigma_{y_{f^\sigma_{y_0}}}) \end{array}$$

Part (1) of the representation theorem is shown in [B.S.S] for $D \subset \mathbb{R}^k$. This theorem is a generalization of a representation theorem Rabin proved for computations of limited length. It is the basis for the development of lower bounds for the decision $x \in D$ depending of the betti-number $\beta_0(D)$ [BenOr].
The representation theorem implies that certain functions are not $\mathbb{R}$-computable.

5 Examples of Non-$\mathbb{R}$-Computable Functions

In [B.S.S] the representation theorem has been used to prove that the characteristic functions of certain Julia-sets are not $\mathbb{R}$-computable. In fact functions $\rho(z) = z^2 + c$ were treated with $c \in \mathbb{C}$ and z varying over $\mathbb{C}$. Looking at the case $|c| > 4$ and the attractor ∞ we realize that the Julia-set of (ρ, ∞), i.e. the set of points which are not attracted by ∞ has a nondenumerable number of connected components. If the characteristic function of that set was $\mathbb{R}$-computable it would contain only a countable number of connected components.
In [Ho] the representation theorem is used to show that fractal curves

$$\mu[0:1] \to \mathbb{R}^2$$

are not $\mathbb{R}$-computable. The Hausdorff-dimension of μ is at most 1 if μ is a rational function or consists of not more than a countable number of rational functions. The well-known Koch curve has the Hausdorff-dimension

$$\frac{1}{(1 - \log_2(\sqrt{5} - 1))} \approx 1,2\,.$$

Hence it is not $\mathbb{R}$-computable. The Hilbert curve is space filling, so its Hausdorff-dimension is 2. Hence it is not $\mathbb{R}$-computable either.

6 Analytic Machines

It is easy to see that the set of the $\mathbb{R}$-computable functions is closed under composition and the cartesian product: To construct a machine computing the function $g \circ f$ we use the output of a machine $\mathcal{M}_f$ (computing the function f) as the input for another machine $\mathcal{M}_g$ (computing g).
Let $f : D \to D$ be an R-computable function. We define

$$f^{(1)} := f \text{ and } f^{(n+1)} := f \circ f^{(n)} \text{ for } n = 1, 2, \ldots.$$

We get "iterations" of f which are also R-computable. The fact that

$$g(n, x) = f^{(n)}(x)$$

is also computable, can be shown by well-known methods from the theory of computing. It is an interesting question, how to continue this function in a natural way from $\mathbb{N} \times D$ to $\mathbb{R} \times D$ or a subset $D' \subset \mathbb{R} \times D$. In general it is not clear how to continue a machine computing $g(n, x)$ in a natural way. Let us have a look at the question what kind of function we get, if we allow an infinite number of iterations. That means we are interested in the function

$$\lim_{n \to \infty} f^{(n)}(x)$$

where f is R-computable. This question is a generalization of a question raised by Julia [J]. We are interested in the following two problems:

- Is it possible to approximate the $\mathbb{R}$-analytic functions by the $\mathbb{Q}$-analytic functions? Our interest in that question is based on the fact that in contrast to $\mathbb{R}$-machines we are able to construct $\mathbb{Q}$-machines. So the question means, can we approximate the ideal world given with $\mathbb{R}$-machines with our real world.

- Is there a general way to decide the convergence? As we mentioned before this problem is connected to the problem af stability of dynamic systems.

We will now define "$\mathbb{Q}$-analytic" and "$\mathbb{R}$-analytic". As we mentioned before the definitions are not quite analogous.

Definition 1: Let $\mathcal{M}$ be an R-machine. We call a computation

$$\rho_0, \rho_1, \rho_2, \ldots$$

analytic iff it satisfies (1) and (2).

(1) $\rho_i \in K_t$ for an infinite number of i

(2) $out(\rho_i)$ converges for $i \to \infty$.

Definition 2: We call $\mathcal{M}$ a δ-$\mathbb{Q}$-machine iff it satisfies (1),(2) and (3).

(1) $\mathcal{M}$ has the same structure as a R-machine and has the same registers. The input variables x_j may contain real numbers; for all the other variables $(\alpha, z_0, z_1, \ldots)$

only rational numbers are allowed. δ and γ contain positive integers and β contains a number from $[0:N]$.

(2) The operations which are defined for the R-machine are also defined for $\mathcal{M}$. $\mathcal{M}$ does not terminate with the operation *end* like an R-machine. Instead of the end operation it has an init operation to restart the computation with the address $\beta = 1$. The instruction at address 1 is

$$\delta := \delta + 1;$$

which is not used elsewhere. The instruction at address N is

$$init;$$

which assigns 0 to α, γ and all z_i and 1 to β. We define the additional instructions

$$\alpha := \delta$$

and

$$\alpha := [x_i]_\delta \text{ and } \alpha := [x(\gamma)]_\delta.$$

The effects of $\delta := \delta + 1$ and $\alpha := \delta$ are obvious. $[\,]_\delta$ is an $\mathbb{R}$-computable function from $\mathbb{R}$ to $\mathbb{Q}$, such that

$$[x]_\delta \in \mathbb{Q} \text{ and } |x - [x]_\delta| < 2^{-\delta}.$$

(3) The definition of K can be deduced from (1). We define K_a as before. K_t is defined like K_e, but K_t are no final states. In addition, we distinguish the set K_b. This is motivated by the special status of the precision register δ. We define

$$K_b := \{k \mid \alpha = \gamma = 0,\ \beta = 1,\ \delta \in \mathbb{N},\ x_0 \in \mathbb{N},\ x_j = 0 \text{ for } j > x_0,\ z_j = 0\ \ \forall j\}.$$

The difference between K_b and K_a is that δ can contain a number greater then 0 in K_b.

Definition 3: Let $\mathcal{M}$ be a δ-$\mathbb{Q}$-machine, then we call the computation

$$\rho_0, \rho_1, \rho_2, \ldots$$

analytic iff it satisfies (1) and (2) from definition 1 and additionally

(3) When the machine leaves K_t it goes to a state of K_b.

Explanation: We want to make sure that the machine only uses one rounding of an input value at a time. Therefore we require (3), namely that all memory cells except the output tape are cleared after the computation for one precision δ.

Definition 4: A function $f : D \to C$ is called $\mathbb{R}$-analytic iff there exists an analytic computation on an $\mathbb{R}$-machine such that

for all $x \in D$ holds: $in(x) = \rho_0, \rho_1, \rho_2, \ldots$ is analytic, and

$$\lim_{i\to\infty}(out(\rho_i)) = f(x).$$

A function $f : D \to C$ is called δ-$\mathbb{Q}$-analytic iff there exists an analytic computation on an δ-$\mathbb{Q}$-machine such that

$$\text{for all } x \in D \text{ holds: } in(x) = \rho_0, \rho_1, \rho_2, \ldots \text{ is analytic, and}$$

$$\lim_{i\to\infty}(out(\rho_i)) = f(x).$$

We will see that the choice of the function $[]_\delta$ has a great influence on the set of the δ-$\mathbb{Q}$-analytic functions.

Definition 5: A rounding is called *normal* iff it satisfies (1) and (2).

(1) For all $x \in \mathbb{R}$ and all $\delta \in \mathbb{N}$ there exists a $\tilde{x} \in \mathbb{R} \setminus \mathbb{Q}$ such that $[x]_\delta = [\tilde{x}]_\delta$.

(2) $[[x]_\delta]_{\delta'} = [x]_{\delta'}$ for all δ, δ' with $\delta' \leq \delta$

It is easy to see that the function that takes δ digits after the point from the infinite binary number notation is an example for a normal rounding.

Definition 6: A program for a δ-$\mathbb{Q}$-machine is called *robust*, iff it computes the same function for every rounding.

A function $f : D \to C$ with $D, C \in \mathbb{R}^*$ is called $\mathbb{Q}$-*analytic*, iff there exists a robust program that computes f analytically.

It is called *normal analytic*, iff it can be computed analytically by a δ-$\mathbb{Q}$-machine using a normal rounding.

Theorem 1: Every $\mathbb{R}$-computable function is $\mathbb{Q}$-analytic.

Proof: Let $f : D \to \mathbb{R}^*$ be $\mathbb{R}$-computable and let $\mathcal{M}$ be an $\mathbb{R}$-machine computing it. In this proof we will regard f as an infinite tuple of single-valued functions $(f_1, f_2, \ldots)$ with $f_i : D \to \mathbb{R}$. We will call $x = (x_1, x_2, \ldots, x_n) \in D$ the input and define $[x]_\delta := ([x_1]_\delta, [x_2]_\delta, \ldots, [x_n]_\delta)$.

The representation theorem implies

$$f = (f_1, f_2, \ldots) = \sum_{\sigma \in S_{\mathcal{M}}} a_\sigma \cdot (f^\sigma_{y_1}, f^\sigma_{y_2}, \ldots, f^\sigma_{y_{f^\sigma_{y_0}}}, 0, 0, \ldots),$$

with $f^\sigma_{y_i}$ being the rational functions belonging to the signature σ and a_σ being the characteristic function of the semi-algebraic set A_σ belonging to σ.

First we examine the case that $x \in A_\sigma$ and A_σ is an open set. In this case

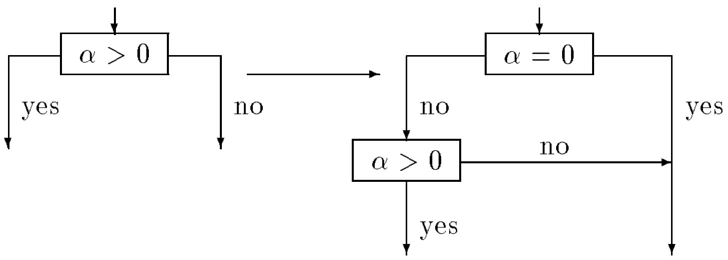
$$\lim_{\delta\to\infty} f^\sigma_{y_i}([x]_\delta) = f^\sigma_{y_i}(x) \text{ holds for all } i.$$

Hence we can compute $f(x)$ analytically with a δ-$\mathbb{Q}$-machine using the program defined like this:

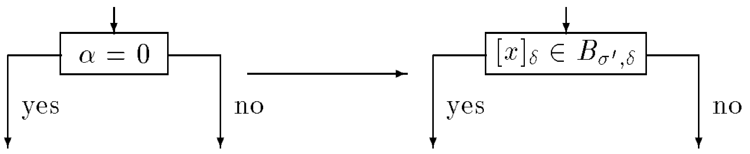
$\pi_{\mathcal{M}}$ of $\mathcal{M}$	$\pi_{\mathcal{M}'}$ of $\mathcal{M}'$
$\pi_{\mathcal{M}}(1)$	$\pi_{\mathcal{M}'}(1) :=$ "$\delta := \delta + 1$;"
$\pi_{\mathcal{M}}(2)$	$\pi_{\mathcal{M}'}(2) := \pi_{\mathcal{M}}(1)$
$\vdots$	$\vdots$
$\pi_{\mathcal{M}}(N-1)$	$\pi_{\mathcal{M}'}(N) := \pi_{\mathcal{M}}(N-1)$
$\pi_{\mathcal{M}}(N) =$ "*end;*"	$\pi_{\mathcal{M}'}(N+1) :=$ "*init;*"

The "end" in π is replaced by "init", so that after every run of the original program the precision used to read x is increased. A_σ is open, so there exists an $M \in \mathbb{N}$ with $[x]_\delta \in A_\sigma$ for $\delta > M$, such that the computation with input $[x]_\delta$ for $\delta > M$ takes the same computation path as for input x. Since the $f^\sigma_{y_i}$ are continuous the output of the computation converges to $f(x)$ in this case. In general $\mathcal{M}'$ will fail to compute correctly if A_σ is not open and $x \in A_\sigma$ is located at the border of A_σ: Let $\sigma' \in \Sigma^\sigma$ be a prefix of σ leading to a conditional $\alpha > 0$ or $\alpha = 0$ as before. Then it is possible that $f^{\sigma'}_\alpha(x) = 0$ but $f^{\sigma'}_\alpha([x]_\delta) \neq 0$ for all δ. In this case the simulation takes the wrong path for every precision. This problem can occur in two cases: At the condition "if $\alpha > 0$" with σ taking the "no"-path or at the condition "if $\alpha = 0$" with σ taking the "yes"-path.

We replace all conditions "$\alpha > 0$" in π like this:



So we only have to consider the second case. We solve our problem by replacing every condition "$\alpha > 0$" in π' by a procedure deciding whether $[x]_\delta \in B_{\sigma,\delta}$ or not. We will show later on that this procedure is computable. We substitute



and obtain a program $\tilde{\pi}$. $B_{\sigma',\delta}$ is an open neighborhood of the set $\{x \mid f^{\sigma'}_\alpha(x) = 0\}$ and is defined such that it contracts itself to this set for $\delta \to \infty$. In addition

$$f^{\sigma'}_\alpha(x) = 0 \Rightarrow [x]_\delta \in B_{\sigma',\delta}.$$

must hold for all δ. We define:

$$B_{\sigma',\delta} := \{\tilde{x} \mid \exists x \in \mathbb{R}^n, (\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2 < n \cdot 2^{-\delta},\ f^{\sigma'}_\alpha(x) = 0)\}$$

By substituting the rational function $f_\alpha^{\sigma'}$ by the polynomial $P_\alpha^{\sigma'}$ that we get by multiplying $f_\alpha^{\sigma'}$ with its numerator we have

$$B_{\sigma',\delta} = \{\tilde{x} \mid \exists x \in \mathbb{R}^n, (\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2 < n \cdot 2^{-\delta},\, P_\alpha^{\sigma'}(x) = 0)\}$$

We apply the construction we used to change π into an analytic program on $\tilde{\pi}$ and obtain a machine $\tilde{\mathcal{M}}$.

We now observe the computations of $\mathcal{M}$ and $\tilde{\mathcal{M}}$ and realize the following:

If the condition "$\alpha = 0$" is satisfied in $\mathcal{M}$ then the computation on $\tilde{\mathcal{M}}$ is taking the same path as the computation for x on $\mathcal{M}$ because $B_{\sigma',\delta}$ has been defined such that $[x]_\delta \in B_{\sigma',\delta}$ if $f_\alpha^{\sigma'}(x) = 0$. On the other hand, if "$\alpha = 0$" is not satisfied in $\mathcal{M}$ then there exists an $M \in \mathbb{N}$ such that $[x]_\delta \notin B_{\sigma',\delta}$ for all $\delta > M$.

If $\delta > M$ we see that the computation for $[x]_\delta$ on $\tilde{\mathcal{M}}$ takes the same path as the computation for x on $\mathcal{M}$. Inductively this holds for every condition in the path σ and so it is obvious that $\tilde{\mathcal{M}}$ is analytically computing f.

There are two more problems to mention: It is possible that $f^\sigma([x]_\delta)$ is infinite for $[x]_\delta \in B_{\sigma',\delta} \setminus A_\sigma$. It could happen that $[x_1]_\delta = 0 \neq x_1$ and the instruction "$\alpha := \alpha / x_1$" should be carried out. In this case we go to the instruction "init" and restart the computation with higher precision. As f^σ has no infinite discontinuity on A_σ, f^σ on $B_{\sigma',\delta}$ has no infinite discontinuity for sufficient large δ.

The second problem is that it is possible that $\tilde{\mathcal{M}}$ never terminates because the rounding $[x]_\delta$ may leave the definition area D of the computed function $f : D \to C$ and so $\mathcal{M}$ may hang up in an endless loop. We solve this problem by restricting the number of branchings, e.g. to δ. This means we abort the computation and go to the instruction "init" after the δ-th branching. There exists a sufficient large δ such that $\tilde{\mathcal{M}}$ will take the right path and does not abort.

Now we have to prove that the test $[x]_\delta \in B_{\sigma',\delta}$ can be decided by a computable procedure. We see that

$$B_{\sigma',\delta} = \{\tilde{x} \mid \exists x \in \mathbb{R}^n, y \in \mathbb{R}, (y^2 + \sum_{i=1}^{n}(x_i - \tilde{x}_i)^2 - n \cdot 2^{-\delta} = 0,\, P_\alpha^{\sigma'}(x) = 0)\}.$$

So we get $B_{\sigma',\delta}$ as the projection $\tilde{x}$ of an algebraic manifold over $x, \tilde{x}$ and y. The test $\tilde{x} \in B_{\sigma',\delta}$ is equivalent to

$$B_{\sigma,\delta}(\tilde{x}) = \{(x,y) \mid y^2 + \sum_{i=1}^{n}(x_i - \tilde{x}_i)^2 - n \cdot 2^{-\delta} = 0,\, P_\alpha^{\sigma'}(x) = 0\} \neq \emptyset.$$

$P_\alpha^{\sigma'}(x)$ is computable, hence $B_{\sigma',\delta}(\tilde{x}) = \emptyset$ is decidable ([T], [Col]). So we can replace the test $\tilde{x} \in B_{\sigma',\delta}$ by a procedure deciding this question. Thus, the proof is completed. The simulation is robust because we have not used any quality of the rounding except $|x - [x]_\delta| < 2^{-\delta}$.

(theorem 1) □

Theorem 2: The set of the $\mathbb{R}$-computable functions is a proper subset of the set of the δ-$\mathbb{Q}$-analytic functions.

Proof: As we mentioned before the Koch curve and Hilbert's space filling curve are not $\mathbb{R}$-computable. They are $\mathbb{Q}$-analytic because these functions are defined by an infinite sequence of $\mathbb{Q}$-computable functions, converging to the Koch curve resp. Hilbert's curve.

(theorem 2) $\square$

We will now look at two functions which will be useful later on. We define $d_i : [0,1] \to \mathbb{R}$ for $i = 1, 2$ by

$$d_1(x) := \begin{cases} \frac{1}{q} & \text{for } x = \frac{p}{q},\ (p,q) = 1,\ p,q \in \mathbb{N} \\ 0 & \text{for } x \in [0,1] \setminus \mathbb{Q} \end{cases}$$

$$d_2(x) := \begin{cases} 1 & \text{for } x \in \mathbb{Q} \cap [0,1] \\ 0 & \text{for } x \in [0,1] \setminus \mathbb{Q}. \end{cases}$$

Lemma 2: d_1 is $\mathbb{Q}$-analytic.

Proof: The machine reads x with the precision δ, i.e. the rational number $[x]_\delta$. The machine searches the smallest rational number $\frac{p}{q} \in \left[[x]_\delta - 2^{-\delta}, [x]_\delta + 2^{-\delta}\right]$ with minimal q[1]. If x is irrational then $\frac{1}{q} \to 0$; else there exists a δ', such that for all $\delta > \delta'$ the input $x = \frac{p}{q}$ is the number with the minimal denominator in the interval $\left[[x]_\delta - 2^{-\delta}, [x]_\delta + 2^{-\delta}\right]$, thus the machine will constantly write $\frac{1}{q}$ on the output.

(lemma 2) $\square$

Lemma 3: d_2 is $\mathbb{R}$-analytic.

Proof: The machine enumerates all rational numbers. Then it compares every q_i to the input x. If $x = q_i$ then the machine writes 1 to the output and *terminates*[2].

(lemma 3) $\square$

We will now examine the question in which cases d_2 is $\mathbb{Q}$-analytic as well and in which cases d_2 is not $\mathbb{Q}$-analytic. For that, $y_i(x, \delta)$ shall denote the output y_i of a δ-$\mathbb{Q}$-machine for the input x with the precision δ.

Lemma 4: There exist roundings $[]_\delta$, such that d_2 is δ-$\mathbb{Q}$-analytic.

[1] This is easy to do by enumerating the positive integers $q \in \mathbb{N}$ and testing whether there is a multiple of $\frac{1}{q}$ in the input interval.

[2] As it is an analytic machine it does not actually terminate but keeps on running without changing the output.

Proof: If there is an approximation $[]_\delta$ for which

$$[x]_\delta = \frac{p}{q} \text{ and } q \le \delta \quad \Rightarrow \quad \frac{p}{q} = x$$

holds then d_2 is δ-$\mathbb{Q}$-analytic. If this condition holds the machine only has to calculate p and q such that $[x]_\delta = \frac{p}{q}$. Then it checks whether $q \le \delta$. If this is the case it writes 1 to the output y_1 else it writes 0. Hence for all $x \in \mathbb{Q}$ there exists a $M \in \mathbb{N}$, such that $y([x]_\delta, \delta) = 1$ for all $\delta > M$.

We still have to show that such an $\mathbb{R}$-computable rounding exists: If $x = \frac{p}{q}$ and $q \le \delta$ we define $[x]_\delta := x$ else we search any number $\frac{s}{t}$ with $t > \delta$ and $|x - \frac{s}{t}| < 2^{-\delta}$ and define $[x]_\delta := \frac{s}{t}$. This rounding is not normal because it does not satisfy the requirement (2) for normal roundings.

(lemma 4) □

Lemma 5: d_2 is not δ-$\mathbb{Q}$-analytic for any normal roundings.

Proof: We assume that there exists a δ-$\mathbb{Q}$-machine computing d_2 analytically with $[]_\delta$ being normal. We now show that this assumption is contradictory.

Let $\frac{1}{2} > \varepsilon > 0$. The assumption implies: If $x \in \mathbb{Q}$ there exists a δ_x such that $y_1(x, \delta) > 1 - \varepsilon$ for $\delta > \delta_x$. If $x \in \mathbb{R} \backslash \mathbb{Q}$ then there exists a δ_x such that $y_1(x, \delta) < \varepsilon$ for $\delta > \delta_x$.

We will construct a real number v using a convergent sequence such that $y_1(x, \delta)$ is oscillating between values greater than $1 - \varepsilon$ and less than ε with increasing δ.

We start with any number $v_1 \in \mathbb{Q}$. Then there exists a δ_1, such that $y_1(v_1, \delta_1) > 1 - \varepsilon$.

The requirement (1) for normal roundings implies that there exists an irrational v_2 such that $[v_2]_{\delta_1} = [v_1]_{\delta_1}$ (with $\delta = \delta' = \delta_1, v_2 = \tilde{x}$ and $v_1 = x$). The requirement (2) for normal roundings implies $[v_2]_\delta = [[v_2]_{\delta_1}]_\delta = [[v_1]_{\delta_1}]_\delta = [v_1]_\delta$ for all $\delta \le \delta_1$. As $v_2 \in \mathbb{R} \setminus \mathbb{Q}$ there exists a $\delta_2 > \delta_1$, such that $y_1(v_2, \delta_2) < \varepsilon$. Then we define $v_3 := [v_2]_{\delta_2} \in \mathbb{Q}$, and find a $\delta_3 > \delta_2$ with $y_1(v_3, \delta_3) > 1 - \varepsilon$.

In the same way we find for all even i an irrational v_i with $[v_i]_\delta = [v_{i-1}]_\delta$ for all $\delta \le \delta_{i-1}$ and a $\delta_i > \delta_{i-1}$, such that $y_1(v_i, \delta_i) < \varepsilon$. For all odd i there exists a rational v_i with $[v_i]_\delta = [v_{i-1}]_\delta$ for all $\delta \le \delta_{i-1}$ and a $\delta_i > \delta_{i-1}$, such that $y_1(v_i, \delta_i) > 1 - \varepsilon$.

As $2^{-\delta_i} \to 0$ for $i \to \infty$ and $v_{i+1} \in [v_i - 2^{-\delta_i}, v_i + 2^{-\delta_i}]$ there exists a limit $\lim\limits_{i \to \infty} v_i$. We define $v := \lim\limits_{i \to \infty} v_i$ and hence $[v]_\delta = [v_j]_\delta$ holds for all j and all $\delta \le \delta_j$. Hence if i is odd $y_1(v, \delta_i) > 1 - \varepsilon$ else $y_1(v, \delta_i) < \varepsilon$. Hence the output of the machine diverges.

(lemma 5) □

From Lemma 3 and Lemma 5 we conclude:

Theorem 3: The set of the normal $\mathbb{Q}$-analytic functions is a proper subset of the set of the $\mathbb{R}$-analytic functions.

We now deduce:

Theorem 4: The set of the δ-$\mathbb{Q}$-analytic functions is not closed under composition.

Proof: By first computing $d_1(x)$ and then comparing it to 0 we can compute d_2:

$$d_2(x) = \begin{cases} 1 & \text{if } d_1(x) \neq 0 \\ 0 & \text{if } d_1(x) = 0 \end{cases}$$

(theorem 4) □

This is not satisfactory at all because the composition of computable functions should be computable as well. Therefore we will define "strong analytic" by increasing the requirements for "analytic".

7 Strong $\mathbb{Q}$-analytic Machines

Definition 7: We call a computation

$$\rho_0, \rho_1, \rho_2, \ldots$$

strong analytic iff (1), (2) and (3) are satisfied.

(1) The computation is analytic.
(2) $|\lim_{j\to\infty} y_l(\rho_j) - y_l(\rho_i)| \leq y_1(\rho_i)$, for all y_l and all $\rho_i \in K_t$
(3) $y_1(\rho_j) \to 0$ for $j \to \infty$.

That means if a computation is strong analytic we know the precision of the output.

Lemma 6: The strong $\mathbb{Q}$-analytic functions are continuous.

Proof: We assume a strong $\mathbb{Q}$-analytic function f with a discontinuity $\tilde{x}$. Without loss of generality let $f(x) > f(\tilde{x}) + \varepsilon$ for a $\varepsilon > 0$ and $x > \tilde{x}$. Then there exists a M such that for all $\delta \geq M$

$$y_1(\tilde{x}, \delta) < \frac{\varepsilon}{3}$$

holds. There exists a rounding $[]_\delta$ and $x > \tilde{x}$, with $[\tilde{x}]_\delta = [x]_\delta$ for all $\delta \leq M$. Hence $y_2(\tilde{x}, M) = y_2(x, M)$ and $y_1(\tilde{x}, M) = y_1(x, M) < \frac{\varepsilon}{3}$. By $|f(x) - f(\tilde{x})| > \varepsilon$ we deduce a contradiction to definition 7.

(lemma 6) □

In the following we show that the strong $\mathbb{Q}$-analytic functions are closed under composition and primitive recursion.

Lemma 7: Let $g : \mathbb{R}^r \to \mathbb{R}$ and $h : \mathbb{R}^{r+2} \to \mathbb{R}$ be two strong $\mathbb{Q}$-analytic functions. We define $f : \mathbb{N} \times \mathbb{R}^r \to \mathbb{R}$ by

$$f(0, x) := g(x)$$

and

$$f(n+1, x) := h(n, f(n, x), x)$$

for $n \in \mathbb{N}$ and $x \in \mathbb{R}^r$. Then f is strong $\mathbb{Q}$-analytic.

Proof: We can compute $f(n, x)$ using $n+1$ δ-$\mathbb{Q}$-machines, using the output of the i-th machine as input for the $i+1$-th.

The computation is analytic so the machines don't terminate. Therefore, all machines have to work simultaneously and have to use the current output of the previous machines.

We have to prove:

1. The output of a δ-$\mathbb{Q}$-machine computing a strong analytic function can be used as input for another δ-$\mathbb{Q}$-machine.
2. It is possible to simulate all reading operations and writing operations in the main memory. This point is obvious.
3. It is possible to simulate several δ-$\mathbb{Q}$-machines by a single one.

ad 1: Let $\mathcal{M}$ be a machine computing a strong analytic function. Let $\bar{y}$ be the real number the output of $\mathcal{M}$ converges to. That means: $y_1(\rho_j) \to 0$ and $y_2(\rho_j) \to \bar{y}$ for $j \to \infty$, with $|\bar{y} - y_2(\rho_j)| \leq y_1(\rho_j)$ for all $\rho_j \in K_t$.

Now we want to use $\bar{y}$ as input for another δ-$\mathbb{Q}$-machine $\mathcal{M}'$. The special characteristics of the rounding are not important. Hence it is sufficient if $|[\bar{y}]_\delta - \bar{y}| < 2^{-\delta}$. If we wait long enough the machine $\mathcal{M}$ is computing so exactly that $y_1 < 2^{-\delta}$ holds. Hence $|\bar{y} - y_2| < 2^{-\delta}$. So we can take y_2 for $[\bar{y}]_\delta$ and use it as input for $\mathcal{M}'$ as soon as $y_1 < 2^{-\delta}$.

ad 3: We prove that a single machine $\mathcal{M}$ can simulate two machines $\mathcal{M}_1$ and $\mathcal{M}_2$. From this we conclude inductively that n machines can be simulated by one.

We divide the memory of $\mathcal{M}$ in 4 sets of memory cells:

$$\begin{array}{lllll} Z^0: & z_1, & z_2, & z_3, & z_4, \quad z_5 \\ Z^1: & z_6, & z_9, & z_{12}, & \ldots \\ Z^2: & z_7, & z_{10}, & z_{13}, & \ldots \\ Z^3: & z_8, & z_{11}, & z_{14}, & \ldots \end{array}$$

Now we identify Z^1 with the memory of $\mathcal{M}_1$, Z^2 with the output of $\mathcal{M}_1$ resp. the input of $\mathcal{M}_2$ and Z^3 with the memory of $\mathcal{M}_2$. The input tape of $\mathcal{M}_1$ is the input tape of $\mathcal{M}$, the output tape of $\mathcal{M}_2$ is the output tape of $\mathcal{M}$.

We use Z^0 to save the registers. Let $z_{\alpha,1} := z_1$, $z_{\gamma,1} := z_2$ be the α- and γ-registers of $\mathcal{M}_1$ and $z_{\alpha,2} := z_3$, $z_{\gamma,2} := z_4$, $z_{\delta,2} := z_5$ the α-, γ- and δ-registers of $\mathcal{M}_2$; The δ-register of $\mathcal{M}$ is the δ-register of $\mathcal{M}_1$.

Now we give an informal description of the program running on $\mathcal{M}$.

- Load α- and γ-register of $\mathcal{M}_1$ from $z_{\alpha,1}$ and $z_{\gamma,1}$.
- Simulate $\mathcal{M}_1$ until δ is increased. It is easy to see that it is no problem to work with Z^1 and Z^2 instead of the whole memory and the output tape. This requires only few modifications in the program of $\mathcal{M}_1$. Write an approximate output into Z^2.
- Save the α- and γ-register.
- Test if the approximate output is precise enough to be used as input for $\mathcal{M}_2$.
- If not, go back to $\mathcal{M}_1$.
- Otherwise load the α- and γ-register of $\mathcal{M}_2$ from $z_{\alpha,2}$ and $z_{\gamma,2}$.
- Simulate $\mathcal{M}_2$ until δ is increased. Write the output of $\mathcal{M}_2$ onto the output tape of $\mathcal{M}$.
- Save the α- and γ-register, increase δ and go on simulating $\mathcal{M}_1$.

The output of $\mathcal{M}_1$ converges and with convergent input the output of $\mathcal{M}_2$ converges as well. Hence by induction we conclude that the strong $\mathbb{Q}$-analytic functions are closed under primitive recursion.

(lemma 7) $\square$

In general we are not able to say whether the minimal precision for which the machine takes the right path σ is reached or not: If there is at least one branching on which we assume $x = 0$ we do not know whether this decision is stable for all computations with higher precision. Therefore, this approximation is only analytic and not strong analytic.

We will now observe the relation between the strong analytic functions and the computable real functions defined by Grzegorczyk. We are not looking at the original definition but at an equivalent one ([Ko], page 51). This definition uses a turing machine with an oracle. The oracle serves the same purpose as our rounding $[]_\delta$. The main difference to our definition of *strong analytic* is the following: The turing machine has to compute an approximation $\tilde{y}$ for a function $f(x)$ with

$$|\tilde{y} - f(x)| \leq 2^{-n}.$$

From this it computes the input precision 2^{-m} needed to compute $\tilde{y}$. Then the machine queries the oracle, i.e. it claims an approximate value $\tilde{x}$ for x with $|x - \tilde{x}| \leq 2^{-m}$. Using $\tilde{x}$ as input the turing machine computes the output $\tilde{y}$. f is called *real computable* iff there exists a machine that can compute a $\tilde{y}$ with $|\tilde{y} - f(x)| \leq 2^{-n}$ for any input $\tilde{x}$ with $|x - \tilde{x}| \leq 2^{-m}$.

It is easy to see that our machine can simulate the turing machine and therefore the real computable functions are strong analytic. We also realize that any strong analytic function is real computable. Here we need that the strong analytic functions are robust.

There exists a precision δ' such that for all $\delta > \delta'$ we know whether $x = [x]_\delta$ holds for every rational x if we use a rounding with

$$[x]_\delta = \frac{p}{q} \text{ and } q \leq \delta \quad \Rightarrow \quad \frac{p}{q} = x$$

With this rounding we can compute strong δ-$\mathbb{Q}$-analytic functions with rational discontinuities. As the real computable functions are continuous the definitions are not equivalent in this case. It is not satisfactory that the definition of strong analytic depends on the rounding for real numbers. On the other hand it is not satisfactory that the Theta function $\Theta(x)$ with $\Theta(x) := 0$ for $x \leq 0$ and $\Theta(x) := 1$ for $x > 1$ is not real computable.

8 Procedures

We now extend our machine model by allowing the use of procedures $\{f_1, \ldots, f_m\}$ which are robust and strong $\mathbb{Q}$-analytic.

Let $\mathcal{M}_{f_i}$ be a $\mathbb{Q}$-machine which computes the strong $\mathbb{Q}$-analytic function f_i analytically.

We define a new instruction for an $\mathbb{R}$-machine by

$$\alpha := f_i(p_1, p_2, \ldots, p_n)$$

which assigns the result of $\mathcal{M}_{f_i}$ for the input $p_1, p_2, \ldots, p_n$ to α. Here p_j is an element of $\{\alpha, x_1, x_2, \ldots, z_1, z_2, \ldots\}$ for $j \in \{1, 2, \ldots, n\}$. Note that this is an extension to the well known term *procedure* because $\mathcal{M}_{f_i}$ is analytical and never has to *return* a value as a result.

$\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$ denotes an $\mathbb{R}$-machine using programs including the procedures $\{f_1, \ldots, f_m\}$.

If we want to include strong $\mathbb{Q}$-analytic functions as procedures into $\mathbb{Q}$-machines we can only use approximations because we are not able to assign real numbers to the variables of the $\mathbb{Q}$-machine. So we define

$$\alpha := [f(p_1, p_2, \ldots, p_n)]_\delta$$

as a new instruction for a $\mathbb{Q}$-machine. Thereby is $[f(p_1, p_2, \ldots, p_n)]_\delta \in \mathbb{Q}$ and $|[f(p_1, p_2, \ldots, p_n)]_\delta - f(p_1, p_2, \ldots, p_n)| < 2^{-\delta}$. $\mathcal{M}_{\mathbb{Q}}(f_1, \ldots, f_m)$ denotes a $\mathbb{Q}$-machine using the procedures $\{f_1, \ldots, f_m\}$.

Can a machine using these procedures be simulated by an ordinary $\mathbb{Q}$-analytic machine? We can compute the approximation $[f(p_1, p_2, \ldots, p_n)]_\delta$ for a strong $\mathbb{Q}$-analytic function f in finite time. Therefore, we have no problems in simulating $\mathcal{M}_{\mathbb{Q}}(f_1, \ldots, f_m)$.

We now examine the machines $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$, for which the exact result is available.

Theorem 5: All functions which can be computed by $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$ are $\mathbb{Q}$-analytic.

Proof: The proof of that theorem is related to the proof of theorem 1. But instead of the question whether $[x]_\delta \in B_{\sigma',\delta}$ we now use the knowledge of the precision of the approximated value x to decide a condition $\alpha = 0$ or $\alpha > 0$. So we can define an interval

$$[x]_\delta - 2^{-\delta} =: x_{min}(\delta) \leq x \leq x_{max}(\delta) := [x]_\delta + 2^{-\delta}.$$

Where $x_{min}(\delta) \rightarrow x$ and $x_{max}(\delta) \rightarrow x$ for $\delta \rightarrow \infty$.

If $x = 0$ than $x_{min}(\delta) \leq 0 \leq x_{max}(\delta)$ holds always.

If $x \neq 0$ then there exists a M, such that for $\delta > M$: $0 < x_{min}(\delta) \leq x_{max}(\delta)$ or $x_{min}(\delta) \leq x_{max}(\delta) < 0$ holds.

If $x_{min}(\delta) \leq 0 \leq x_{max}(\delta)$ we assume $x = 0$ and chose the corresponding branch. If $x = 0$ we are always right; If $x \neq 0$ then there is a M so that we avoid this mistake for all $\delta > M$.

To keep the knowledge about the precision when a procedure call is made, we use the fact that these functions are strong $\mathbb{Q}$-analytic and therefore can tell us the precision of their current approximation.

Every $\mathbb{R}$-machine terminates after a finite number of branchings. Therefore there exists a M such that for all $\delta \geq M$ the δ-$\mathbb{Q}$-machine takes the right computation path.

As in the proof of theorem 1 difficulties with infinite discontinuities or non-terminating computations can occur. They are treated analogously.

This simulation is robust as we have not used any quality of the rounding except $|x - [x]_\delta| < 2^{-\delta}$.

(theorem 5) $\square$

Remark: In general, functions analytically computed by $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$ are not $\mathbb{Q}$-analytic. This is easy to see because they include the $\mathbb{R}$-analytic functions which are not $\mathbb{Q}$-analytic.

9 The Halting Problem

We examine whether the Halting Problem of R-analytic machines is R-analytic itself. Without loss of generality we do without the instructions

$$\alpha := \alpha \star r \text{ with } \star \in \{+, -, \cdot, /\}, r \in \mathbb{R}$$

and interpret the real numbers r as additional input values. Therefore, the correct programs for R-machines are countable. Let Π be the set of correct programs and let $\mathcal{M}_\pi$ be the machine with the program $\pi \in \Pi$. There exists a computable injective function $\phi : \Pi \to \mathbb{Q}$, for which the inverse ϕ^{-1} is also computable. We define the function $erg : \Pi \times \mathbb{R}^* \to \mathbb{R} \cup \{u\}$, such that for all $x \in \mathbb{R}^*$ and $\pi \in \Pi$ holds: $erg(\pi, x) := y$ if $\mathcal{M}_\pi$ converges to $y \in \mathbb{R}$ for the input x and $erg(\pi, x) := u$ if $\mathcal{M}_\pi$ diverges for the input x.

Now we assume the existence of a machine $\mathcal{M}_{\pi'}$ which can decide whether the output of an arbitrary machine $\mathcal{M}_\pi$ converges or diverges for any given input $x \in \mathbb{R}^*$. Its result is 1 in the case of divergence and 2 in the case of convergence. If $\mathcal{M}_{\pi'}$ can do this it can do it for the special input $x = \phi(\pi)$ as well. So for all $\pi \in \Pi$

$$erg(\pi', \phi(\pi)) = \begin{cases} 1 & \text{if } erg(\pi, \phi(\pi)) = u \\ 2 & \text{otherwise} \end{cases}$$

Then we define π'' by programing an infinite loop that produces an output oscillating between 0 and 1 if $\mathcal{M}_{\pi'}$ produces an output close to 2. So we get

$$erg(\pi'', \phi(\pi)) = \begin{cases} 1 & \text{if } erg(\pi, \phi(\pi)) = u \\ u & \text{otherwise.} \end{cases}$$

Hence

$$erg(\pi'', \phi(\pi'')) = \begin{cases} 1 & \text{if } erg(\pi'', \phi(\pi'')) = u \\ u & \text{if } erg(\pi'', \phi(\pi'')) = 1 \end{cases}$$

This contradiction proves the incorrectness of the assumption. Therefore the question of the convergence of an analytic R-machine is not analytically decidable by an R-machine in general.

It follows from results in ergodic theory [A.A], that in general, the halting problem can not be decided by numerical methods. Our result is stronger in so far as it states that there is no universal decision algorithm with finite description, even if we allow real arithmetic and infinite computations.

Literature

[A.A] Arnold V.I., Avez A.
Ergodic Problems of Classical Mechanics
Addison-Wesley, Advanced Book Classics (1989)

[BenOr] Ben-Or, M.
Lower Bounds For Algebraic Computation Trees
Proceedings 15th ACM STOC, 1983, 80-86

[B.S.S] Blum L., Shub M., Smale St.
On a theory of computation and complexity over the real numbers:
NP completeness, recursive functions and universal machines
Bull AMS Vol 21 (1989)

[Col] Collins, G. E.
Quantifier Elimination for Real Closed Fields
by Cylindrical Algebraic Decomposition
Proceedings of EUROSAM 74, SIGSAM Bulletin, Vol. 8, No. 3 (1974)

[Grz] Grzegorczyk, A.
On the definitions of computable real continuous functions
Fund. Math. 44, 61-71

[Ho] Hotz, G.
Über Berechenbarkeit fraktaler Strukturen,
Abhandlungen der Akademie der Wissenschaften und Literatur, Mainz
Math.-Nat. Klasse. (1994, Nr. 1)

[J] Julia, G.
Sur l'iteration des fonctions rationelles
Jour. de Math. Pure et Appl. 8 (1918)

[Ko] Ko, Ker-I
Complexity Theory of Real Functions
Birkhäuser Boston (1991)

[S] Siegel, C.L.
Vorlesungen über Himmelsmechanik (Kap. 3)
Springer Verlag (1956)

[T] Tarski, A.
A Decision Method for Elementary Algebra and Geometrie
second ed., rev., Univ. of California Press, Berkeley. (1951)