ELSEVIER

Theoretical Computer Science 219 (1999) 151–167

Theoretical Computer Science

www.elsevier.com/locate/tcs

Analytic machines

Thomas Chadzelek *, Günter Hotz

Fachbereich Informatik, Universität des Saarlandes, Postfach 15 11 50, 66041 Saarbrücken, Germany

Abstract

In this paper we present some results about *analytic machines* regarding the power of computations over $\mathbb{Q}$ or $\mathbb{R}$, solutions of differential equations and the stability problem of dynamical systems.

We first explain the machine model, which is a kind of Blum–Shub–Smale machine enhanced by infinite convergent computations. Next, we compare the computational power of such machines over the fields $\mathbb{Q}$ and $\mathbb{R}$ showing that finite computations with real numbers can be simulated by infinite converging computations on rational numbers, but the precision of the approximation is not known during the process. Our attention is then shifted to *ordinary differential equations* (ODEs) where we establish sufficient criteria for the computability of their solutions within our model. We investigate dynamical systems described by ODEs and show the undecidability of a class of stability problems for dynamical systems.

Keywords: Real Turing machine; Ordinary differential equations; Dynamical systems; Stability problem

1. Introduction

Why should one consider machines computing with real numbers if only rational numbers appear in actual computations? First of all one can object that also nearly all rational numbers will never appear in a computation and even the successor function is not actually computable. The introduction of the infinite and of the real numbers greatly simplified analysis and in so far real numbers have proven to be very practical. They also help to investigate into the inherent computational complexity of a problem without having to address numerical issues at the same time; this greatly facilitates understanding.

We are interested here in computations taking infinitely long in order to examine how far computations on machines over the real numbers can be approximated by

* Corresponding author. E-mail: chadzele@cs.uni-sb.de.

0304-3975/99/$ - see front matter
PII: S0304-3975(98)00287-4

computations on finite machines. We regard all functions that can be approximated in this sense as interesting objects for the theory of machines. These include closure properties of such functions under composition and solutions of differential equations computable in this sense.

But our formulation of the question is also motivated by entirely concrete problems. Computers control vehicles, airplanes, power plants, and chemical factories. These processes or at least part of them are continuous and can only be described by differential equations. The computer obtains information about the current state of the process via sensors. This information consists of measurements of limited precision which are available to the computer as inputs. If one wants to ensure the "correctness" of the whole system – computer plus controlled process – then the theory must contain both the computer and the continuous process. Our δ–$\mathbb{Q}$-analytic machines take this notion into account by receiving values as input which are obtained by a rounding with "precision δ". Systems of discrete and continuous components are called hybrid on the proposal of Nerode [4]. Here we are not interested in proving the correctness of hybrid systems but in simulating them approximately and in the question of their stability.

We establish criteria for the ability to approximate real functions by computations on analytic $\mathbb{Q}$-machines and show that even simple questions about the stability of such systems are not generally decidable. By showing that these systems can be conceived as dynamical systems we also make a contribution to a classical problem of computer science [8, Ch. 3, 1]. A particular challenge for the theory is represented by the question of diagnosing systems which obviously work erroneously [7].

2. Machine model

We first present an abstract notion of *mathematical machines* and *analytic computations* which we use later to define a more concrete model of register machines over the rational or real numbers. A mathematical machine in our sense is a tuple

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

where K is the set of *configurations* or *states* of $\mathcal{M}$ and $K_a, K_e, K_z \subset K$ are the *initial, final* and *target configurations*. $\Delta : K \to K$ with $\Delta|_{K_e} = id_{K_e}$ is the *next state function* of $\mathcal{M}$; $in : A^* \to K_a$ and $out : K \to A^*$ are called *input* and *output functions* over the *alphabet*[1] A with $A^* := \bigcup_{i=0}^{\infty} A^i$.

We call a sequence $b = (k_i)_{i=0}^{\infty}$ of states $k_i := \Delta^i(k_0)$ a *computation* of $\mathcal{M}$ applied to k_0. It is called *finite* iff $\exists n$: $k_n \in K_e$; the sequence then becomes stationary at the nth term and the smallest such n is called the *length* and $out(k_n)$ is called the *result* of the computation. If b is finite with $k_0 \in K_a$ we call it *regular*.

For any given metric on A^* we can extend the above definition to infinite convergent computations. Let b be a computation with $k_0 \in K_a$ such that $k_{i_j} \in K_z$ for infinitely

[1] This notion is not confined to a *finite* set here.

many i_j and let $(k_{i_j})_{j=0}^{\infty}$ be the partial sequence of all these target configurations. The computation is now called *analytic* if

$$\lim_{j\to\infty} out(k_{i_j})$$

exists; this limes is the result of b and $out(k_{i_n})$ is called nth *approximation* of the result.

This machine $\mathcal{M}$ now defines a partial function $\Phi_{\mathcal{M}} : A^* \rightsquigarrow A^*$ in the following way. If for any given $x \in A^*$ the computation of $\mathcal{M}$ applied to $in(x)$ is regular or analytic with result $y \in A^*$ we take $\Phi_{\mathcal{M}}(x) := y$ and undefined else. Furthermore, the nth approximation $\Phi_{\mathcal{M}}^{(n)}$ of this function is defined on the same domain by $\Phi_{\mathcal{M}}^{(n)}(x) := out(k_{i_n})$; if a (regular) computation contains less than n target configurations we take $\Phi_{\mathcal{M}}^{(n)}(x) := \Phi_{\mathcal{M}}(x)$.

We denote the *domain* of $\Phi_{\mathcal{M}}$ by $\mathbb{D}_{\mathcal{M}}$; the *halting set* $\mathbb{D}_{\mathcal{M}}^{\mathrm{H}} \subset \mathbb{D}_{\mathcal{M}}$ contains exactly those inputs for which the computation of $\mathcal{M}$ is regular.

2.1. Register machines

Now, we introduce a special kind of register machines over a ring $\mathcal{R}$ which will only be used for $\mathcal{R} \in \{\mathbb{Q}, \mathbb{R}\}$ although the definition can easily be extended to arbitrary rings with unity containing the integers $\mathbb{Z}$. In the case of analytic machines $\mathcal{R}$ has to be a metric space. These conditions are implicitly assumed if we speak of a *ring* later. The construction is similar to [9] and – concerning (finite) computability – equivalent to the model of Blum et al. [2].

These $\mathcal{R}$-machines (cf. Fig. 1) are equipped with a finite program π and a control unit with an accumulator α, program counter β, index or address register γ, and precision register δ. Furthermore, there is an infinite input tape[2] X which may only be read, an infinite output tape Y which may only be written to, and an infinite memory Z. The precision register is only used for extended $\mathbb{Q}$-machines and explained later.

A configuration of such a machine is given by the contents $\pi : [1 : N] \to \Omega$ of the program and $\alpha \in R$, $\beta \in [1 : N]$, $\gamma, \delta \in \mathbb{N}$ of the registers as well as $x, y, z : \mathbb{N} \to R$ of

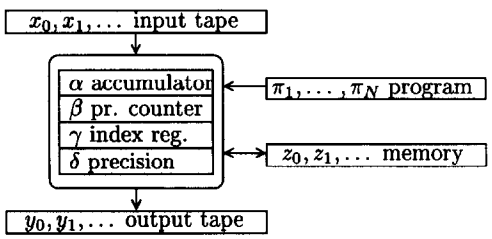

Fig. 1. Structure of our register machine.

[2] The word *tape* here is merely a conventional phrase which does not indicate the presence of a read/write head. In fact, we assume random access to each cell.

Table 1
Instruction set

1.	assignments ($i \in \mathbb{N} \cup \{\gamma\}$)
	(a) $\alpha := x_i$, $\alpha := z_i$, $y_i := \alpha$, $z_i := \alpha$
	(b) $\alpha := r$ for $r \in \mathscr{R}$
	(c) $\alpha := \delta$
	(d) $\gamma := 0$
2.	arithmetical operations ($i \in \mathbb{N} \cup \{\gamma\}$)
	(a) $\alpha := \alpha + z_i$, $\alpha := \alpha \cdot z_i$
	(b) $\alpha := -\alpha$, $\alpha := \alpha^{-1}$
	(c) $\gamma := \gamma + 1$, $\gamma := \gamma - 1$
3.	conditional branching ($m, n \in [1 : N]$)
	`if` $\alpha > 0$ `then goto` m `else goto` n
4.	special instructions
	`end`, `next` δ, `print`

the tapes and memory. Here Ω denotes the set of machine instructions to be specified shortly. For the sake of simplicity we do not distinguish between the names of registers and their contents and abbreviate $x(i)$ by x_i, etc.

$$K := \{k = (\alpha, \beta, \gamma, \delta, \pi, x, y, z) \text{ as above}\},$$

$$K_a := \{k \in K \mid \alpha = \gamma = \delta = 0,\ \beta = 1,\ \forall j\colon y_j = z_j = 0\},$$

$$K_e := \{k \in K \mid \pi_\beta = \texttt{end}\},$$

$$K_z := \{k \in K \mid \pi_\beta = \texttt{print}\}.$$

The input and output functions interpret x_0 and y_0 as the length of a sequence following in the next cells. The set $\Omega = \Omega_N^{\mathscr{R}}$ contains the instructions in Table 1; it depends on the size N of the program and the ring $\mathscr{R}$ but this is usually not denoted explicitly.

The semantics of these instructions should be quite obvious and define in a natural way the next state function Δ. $\dot{-}$ denotes the *non-negative difference*, `print` only marks target configurations, and '`next` δ' is reserved for extended $\mathbb{Q}$-machines. A program is only deemed correct if $\alpha := \alpha^{-1}$ is only applied to invertible elements regardless of the input.

Definition 1. Given a ring $\mathscr{R}$, a natural number N, and a program $\pi : [1 : N] \to \Omega_N^{\mathscr{R}}$, we call the abstract machine $\mathscr{M}_\pi^{\mathscr{R}} = (K, K_a, K_e, K_z, \Delta, \mathscr{R}, in, out)$ uniquely defined by the above construction the *$\mathscr{R}$-machine with program π*.

2.2. Extended $\mathbb{Q}$-machines

An infinite computation of a $\mathbb{Q}$-machine could produce an output sequence (of rational numbers) that converges to an irrational real number. In this way – which is not covered by our definition – a function $f : \mathbb{Q}^* \rightsquigarrow \mathbb{R}^*$ could be computed. We shall now

extend our model of $\mathbb{Q}$-machines in a suitable way to allow real inputs and outputs and thus compute functions $f:\mathbb{R}^* \rightsquigarrow \mathbb{R}^*$. The simple idea is to *round* real inputs to a certain precision, compute a rational approximation of the result, and then increase precision so that the output converges to the real function value.

This means that instructions $\alpha := x_i$ read a rational approximation $x_\delta \in \mathbb{Q}$ with $|x_i - x_\delta| < 2^{-\delta}$ of the real-valued input x_i. We proceed analogously for assignments $\alpha := r$ of *irrational* constants. The precision is increased with each '`next` δ' which also restarts the machine. Formally, this means that for a configuration $k = (\alpha, \beta, \gamma, \delta, \pi, x, y, z)$ with $\pi_\beta =$ '`next` δ' we have $\Delta(k) := (0, 1, 0, \delta + 1, \pi, x, y, z)$. Furthermore, we allow real numbers on the input tape and as program constants ($in:\mathbb{R}^* \to K_a, x:\mathbb{N} \to \mathbb{R}$ and $\pi:[1:N] \to \Omega_N^{\mathbb{R}}$) and the limes $\lim_{j\to\infty} out(k_{i_j})$ in analytical computations need only exist in $\mathbb{R}$.

Extended $\mathbb{Q}$-machines are not determined by the program alone but we also have to specify how to round. As this should be done effectively[3] we restrict ourselves to $\mathbb{R}$-computable functions which will be introduced in Definition 4 (please note that their definition does not depend on rounding functions).

Definition 2. An $\mathbb{R}$-computable function $\rho:\mathbb{R}\times\mathbb{N}\to\mathbb{Q}$, $(x,n)\mapsto x_n$ is called *rounding function*, if always $|x - x_n| < 2^{-n}$.

Given a rounding function ρ the assignment $\alpha := x_i$ is interpreted as $\alpha := \rho(x_i, \delta)$ and the machine remains deterministic.

Definition 3. Given a program $\pi:[1:N]\to\Omega_N^{\mathbb{R}}$ and a rounding function ρ, we call the abstract machine $\mathcal{M}_{\pi,\rho}^{\delta\text{-}\mathbb{Q}} = (K, K_a, K_e, K_z, \Delta, \mathcal{R}, in, out)$ uniquely defined by the above construction the *δ-$\mathbb{Q}$-machine with program π and rounding function ρ.*

The dependency on the rounding function is disturbing, thus we are especially interested in programs π which compute the same function regardless of which rounding is used. Such programs will be called *robust* and any one of the equivalent δ-$\mathbb{Q}$-machines with program π is called $\mathcal{M}_\pi^{\delta\text{-}\mathbb{Q}}$. Of course, one might then consider non-computable rounding functions, but it would make no difference.

2.3. Computable functions

Now, we are in a position to formalize our notion of computable functions over a ring, of which there are many variants. Fig. 2 attempts to give an overview of the hierarchy of classes of computable functions together with a hint[4] to why each inclusion is strict. All classes below the line shown are closed under composition, but none above the line are.

[3] It seems natural not to introduce non-computability at this point because it would affect the power of δ-$\mathbb{Q}$-analytic machines.

[4] Not all of these will be explained in this article.

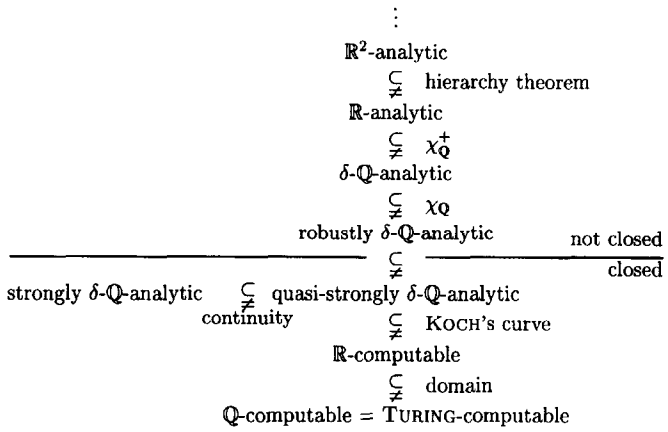

Fig. 2. Hierarchy of classes of computable functions.

Definition 4. A function $f:\mathscr{R}^* \supset D \to \mathscr{R}^*$ is called *analytically $\mathscr{R}$-computable* or short *$\mathscr{R}$-analytic* if there exists an $\mathscr{R}$-machine $\mathscr{M}$ such that $f = \Phi_{\mathscr{M}}|_D$ (and $D \subset \mathbb{D}_{\mathscr{M}}$). If $D \subset \mathbb{D}_{\mathscr{M}}^{\mathrm{H}}$ is a subset of the halting set of $\mathscr{M}$ then f is called *$\mathscr{R}$-computable.*

Analogously, $f:\mathbb{R}^* \supset D \to \mathbb{R}^*$ is called *δ-$\mathbb{Q}$-computable* or *δ-$\mathbb{Q}$-analytic* resp. if there exists a corresponding δ-$\mathbb{Q}$-machine $\mathscr{M} := \mathscr{M}_{\pi,\rho}^{\delta\text{-}\mathbb{Q}}$. Note that the program π as well as the rounding function ρ may be chosen in a suitable way. If we restrict ourselves to robust programs we speak of *robustly δ-$\mathbb{Q}$-computable* or *-analytic* resp.

We call a set $M \subset \mathscr{R}^*$ (*analytically*) *$\mathscr{R}$-decidable* if its characteristic function χ_M is (analytically) $\mathscr{R}$-computable.

A fundamental result (cf. [2, 6]) about $\mathbb{R}$-computable functions is the following *representation theorem.*

Theorem 1. *An $\mathbb{R}$-computable function decomposes its domain into a countable union of semi-algebraic sets; on each semi-algebraic set the function is rational.*

2.4. Quasi-strongly analytic functions

A naïve simulation of the composition $\mathscr{M}_2 \circ \mathscr{M}_1$ of two robust analytic δ-$\mathbb{Q}$-machines by a single machine $\mathscr{M}$ fails when $\mathscr{M}_2$ wants to read its input. The latter is the limit of the first machine's output and must be rationally approximated with a given precision by an arbitrary rounding function. The problem is that we never know this limit itself but only (rational) approximations to it with an unknown precision. It is solved if we turn our attention to programs which also compute a bound on the precision of these approximations in y_1.

Definition 5. Let $(k_{i_j})_{j=0}^{\infty}$ be the subsequence of target configurations of a δ-$\mathbb{Q}$-analytic computation. By $out(k_{i_j}) = (y_1^{(j)}, \ldots, y_{n_j}^{(j)}) \in \mathbb{Q}^*$ we denote the outputs and use the

abbreviations $y_i := \lim_{j\to\infty} y_i^{(j)}$ and $n := \lim_{j\to\infty} n_j$. The computation is then called *quasi-strongly* δ-$\mathbb{Q}$-*analytic* iff

1. $y_1 = 0$;
2. $\exists J\ \forall 2 \leqslant i \leqslant n\ \forall j > J$: $|y_i - y_i^{(j)}| \leqslant y_1^{(j)}$.

We regard the limes $(y_2, \ldots, y_n)$ of outputs without the precision bound as result of this computation. A function $f : \mathbb{R}^* \supset D \to \mathbb{R}^*$ is called *quasi-strongly* δ-$\mathbb{Q}$-*analytic* if there exists a robust program π such that $D \subset \mathbb{D}_{\mathscr{M}}$ and for each $x \in D$ the computation of $\mathscr{M} := \mathscr{M}_\pi^{\delta\text{-}\mathbb{Q}}$ starting with $in(x)$ is quasi-strongly δ-$\mathbb{Q}$-analytic with result $f(x)$.

Note that if we requested the precision bound to hold *always* (i.e. $J = 0$) then the computed function would become *continuous*; we call such functions *strongly* δ-$\mathbb{Q}$-*analytic*. If we restrict ourselves to rational constants in the program this notion coincides with Grzegorczyk's [5] and Weihrauch's [10, 11] definition of *computable real (continuous) functions*. Our weaker requirement suffices, nevertheless, to achieve closure under composition.

Lemma 2. *Let* $D \subset \mathbb{R}^*$ *and* $f : \mathbb{R}^* \rightsquigarrow D$ *as well as* $g : D \to \mathbb{R}^*$ *be quasi-strongly* δ-$\mathbb{Q}$-*analytic, then* $g \circ f$ *is also quasi-strongly* δ-$\mathbb{Q}$-*analytic.*

Proof. We denote by $\mathscr{M}_f$ and $\mathscr{M}_g$ two quasi-strongly analytic δ-$\mathbb{Q}$-machines for f and g which w.l.o.g. execute '`next` δ' infinitely often during each computation, thus dividing them into *phases*. Now, a single machine $\mathscr{M}$ alternately simulates one phase of $\mathscr{M}_f$ on the original input x and one phase of $\mathscr{M}_g$ on the approximation of $f(x)$ computed so far if the precision bound as computed by $\mathscr{M}_f$ seems suitable else $\mathscr{M}_g$ waits. One observes that $\mathscr{M}_g$ is provided with a wrong – i.e. not precise enough – input finitely many times, but this does not matter for the limes of its output. The precision bound of $\mathscr{M}$ itself becomes wrong only finitely often and the whole computation is quasi-strongly δ-$\mathbb{Q}$-analytic. □

In contrast to Weihrauch's class of computable real continuous functions, the quasi-strongly δ-$\mathbb{Q}$-analytic functions form a much larger class containing the $\mathbb{R}$-computable ones. One advantage of the model of extended $\mathbb{Q}$-machines is that it provides a means to compare the computational power of machines with rational or real arithmetic on the same (real) inputs. What we see now is that finite computations on (infinite) reals can be simulated by infinite (but convergent) computations on (finite) rationals. A weaker form of the following statement with a more complicated proof can be found in [6, 9].

Theorem 3 (Simulation theorem). *Every* $\mathbb{R}$-*computable function is quasi-strongly* δ-$\mathbb{Q}$-*analytic.*

Proof. Let $\mathscr{M}$ be an $\mathbb{R}$-machine computing an extension of the given function. The δ-$\mathbb{Q}$-machine $\mathscr{M}'$ simulates $\mathscr{M}$ by interval arithmetic and with increasing precision δ – over and over again in so-called *phases*. In doing so all cells of the memory and

output tape as well as the accumulator of $\mathscr{M}$ are recreated by lower and upper bounds in the memory of $\mathscr{M}'$ which is correctly initialized to zero. Most instructions can be emulated in a self-evident way except the following.

Assignments $\alpha := x_i$ (and analogously $\alpha := r$ for $r \notin \mathbb{Q}$) for which a δ-$\mathbb{Q}$-machine executes $\alpha := \rho(x_i, \delta)$ assign to the simulated accumulator the interval $[\alpha - 2^{-\delta}, \alpha + 2^{-\delta}]$ which safely contains x_i. The branching condition `if` $\alpha > 0$ `then` ... is interpreted in such a way that an interval is positive iff its lower bound is; in this sense it is "equal to zero" as long as it contains zero. The `print`-instruction is now used to write the output, i.e. the interval centers of the simulated output tape together with the maximal interval length as precision bound. Instead of `end` we do a '`next` δ' to start a new phase of the simulation.

If $\mathscr{M}$'s content of α is non-zero at a branching then the interval computed by $\mathscr{M}'$ with sufficient precision reflects the right sign. By approaching the undecidable case of $\alpha = 0$ carefully and from the secure side it is always correctly handled by $\mathscr{M}'$ – we call this approach *conservative branching*. Thus, it is clear that every finite computational path of $\mathscr{M}$ will be simulated by $\mathscr{M}'$ after a finite time and then the precision bound is correct and the output converges as desired.

There may appear some problems while $\mathscr{M}'$ is still simulating the wrong path but they can be easily solved. If the simulation encounters an instruction $\alpha := \alpha^{-1}$ while the interval for α contains zero we simply start a new phase. By branching at most δ times in each phase we can avoid endless loops which might occur on a wrong path. □

2.5. Halting problems

The analytic equivalent of the classical halting problem for Turing-machines is a *convergence problem* – namely the question whether the output[5] of an analytic $\mathscr{R}$-machine converges for a given input. As can be expected, a problem of this kind is undecidable and thus its characteristic function – with which we often identify the problem – is not computable. If we call the composition of i analytic $\mathscr{R}$-machines an $\mathscr{R}^i$-analytic machine and speak of $\mathscr{R}^i$-analytic functions, etc., we can summarize the following results which give rise to the hierarchy Theorem 5 of [6, 9].

Lemma 4. *The convergence problem of $\mathbb{R}^i$-analytic machines is not $\mathbb{R}^i$-analytic; the same holds for (robust) δ-$\mathbb{Q}^i$-analytic machines.*

Proof. A program consists of a series of instructions from a finite alphabet and optional indices, addresses, or (real) constants. Thus we can easily find a prefix-free encoding of $\Omega^{\mathbb{R}}$ in $\mathbb{R}^*$ which we extend to Π, the set of all correct programs, and finally to Π^i. This makes clear that programs can be used as inputs to computations. A tuple $\pi = (\pi_1, \ldots, \pi_i)$ of programs determines a partial function $\Phi_\pi : \mathbb{R}^* \rightsquigarrow \mathbb{R}^*$ which is computed by the composition $\mathscr{M}_{\pi_i} \circ \cdots \circ \mathscr{M}_{\pi_1}$ of $\mathbb{R}$-machines.

[5] I.e. content of output tape at `print` instructions.

Now assume that a tuple $\pi' \in \Pi^i$ exists which solves the convergence problem, i.e. $\mathbb{R}^i$-analytically computes the function $\Phi_{\pi'} : \Pi^i \times \mathbb{R}^* \to \mathbb{B}$ with

$$(\pi, x) \mapsto \begin{cases} 1 & \text{if } \Phi_\pi(x) \text{ defined,} \\ 0 & \text{else.} \end{cases}$$

Then one could easily change[6] π' into π'' in order to $\mathbb{R}^i$-analytically compute the partial function $\Phi_{\pi''} : \Pi^i \rightsquigarrow \mathbb{B}$ with

$$\pi \mapsto \begin{cases} 0 & \text{if } \Phi_\pi(\pi) \text{ undefined,} \\ \text{undefined} & \text{else.} \end{cases}$$

This argument is also valid for robust δ-$\mathbb{Q}$-analytic machines and leads to the classical contradiction

$$\Phi_{\pi''}(\pi'') \text{ defined} \Leftrightarrow \Phi_{\pi''}(\pi'') \text{ undefined.}$$

One can replace Π by $\Pi \times \mathrm{P}$, where $\mathrm{P} \subset \Pi$ is the set of all programs for $\mathbb{R}$-computable rounding functions, in order to deal with the non-robust case. □

Theorem 5 (Hierarchy theorem).

$$\forall i \in \mathbb{N}: \{\mathbb{R}^i\text{-analytic}\} \subsetneq \{\mathbb{R}^{i+1}\text{-analytic}\}.$$

Now, we show how to decide convergence of an $\mathbb{R}$-analytic machine by means of two $\mathbb{R}$-analytic machines; similar constructions can be done for (robust) δ-$\mathbb{Q}$-analytic machines. The following proof makes use of infinite intermediate results which can easily be integrated into a model of infinite computations; they can also be avoided at the cost of some coding and decoding and one extra machine for intermediate computations. We omit further technical details because this is not the focus of the current article.

Lemma 6. *Let $\mathcal{M}$ be an analytic $\mathbb{R}$-machine with $\Phi_\mathcal{M} : \mathbb{R}^* \supset \mathbb{D}_\mathcal{M} \to \mathbb{R}$. Then the set $\mathbb{D}_\mathcal{M}$ is $\mathbb{R}^2$-analytically decidable.*

Proof. For a given $x \in \mathbb{R}^*$ let u_n be the output of the nth target configuration of the computation of $\mathcal{M}$ applied to $in(x)$. We have to decide whether $\lim_{n\to\infty} u_n$ exists, i.e. whether the sequence is bounded and has exactly one accumulation point. The main idea now is that a monotonic sequence is unbounded iff the reciprocal values form an null sequence; for the sake of simplicity we ignore the possibility of division by zero. Thus let

$$M := \lim_{n\to\infty} \left(\max_{i \leqslant n} |u_i| \right)^{-1}.$$

[6] Double the input and replace all outputs in $[\frac{1}{2}, \frac{3}{2}]$ by 0 and 1 alternately.

Now consider intervals $I_n \supset \{u_0, \ldots, u_n\}$ of the form $[-2^l, 2^l]$ with minimal l and their equidistant refinements $J_{n,k}$ consisting of p_n subintervals where p_n is the nth prime. In this way no interval boundary of $J_{n,k}$ occurs twice. A subinterval $J_{n,k}$ has an accumulation point iff

$$\chi_{n,k} := \lim_{m\to\infty} \left(\sum_{i=0}^{m} \chi_{i,n,k} \right)^{-1} = 0,$$

where

$$\chi_{i,n,k} = \begin{cases} 1 & \text{if } u_i \in J_{n,k}, \\ 0 & \text{else.} \end{cases}$$

M is $\mathbb{R}$-analytic because all u_n are $\mathbb{R}$-computable; the $\chi_{n,k}$ are $\mathbb{R}$-analytic because all partial sums are $\mathbb{R}$-computable. The first machine $\mathscr{M}_1$ thus computes M and all $\chi_{n,k}$ giving an infinite output word.

The number of accumulation points in I_n is

$$\alpha_n := \sum_{k=1}^{p_n} (1 - \operatorname{sign}(\chi_{n,k})),$$

if none lies on an interval boundary. The sequence $(u_n)_{n=0}^{\infty}$ diverges iff $\operatorname{sign}(M) = 0$ or $\alpha_n \geqslant 2$ for more than one[7] n. Thus, the second machine first computes $\operatorname{sign}(M)$ and then – one after the other – all α_n, assuming convergence until the contrary is proven. □

3. Ordinary differential equations and stability

Many natural or technological processes can be described by differential equations, either by *ordinary* (ODE) or partial ones. They typically express a local understanding of how something happens while their solutions give a global view of the system. We now want to demonstrate that analytic functions form a large class containing the solutions to (certain) differential equations and then give an undecidability result for a stability problem of dynamic systems modeled by ODEs.

To this end we restrict ourselves to *initial value problems* for *systems of explicit first order ODEs* with a "right-hand side" which is computable by an $\mathbb{R}$-machine *without division*. Let $N \in \mathbb{N}$ be the dimension of the system, $\mathbf{f} : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$ with $(t, \xi) \mapsto \mathbf{f}(t, \xi)$ a computable function over the ring $\mathbb{R}$, and $(t_0, \mathbf{x}_0) \in \mathbb{R} \times \mathbb{R}^N$. We then consider initial value problems of the form

$$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0. \tag{1}$$

[7] A single slip may occur if the limit coincides with an interval boundary, but this cannot happen twice.

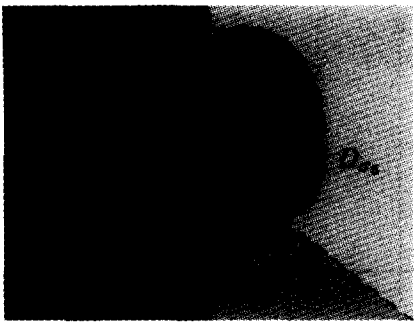

Fig. 3. Partitioning of domain $\mathbb{R}^2$ into regions.

3.1. Solving ODEs

Now, we shall clarify the definition, existence, uniqueness, and computability of solutions to Eq. (1). Because the right-hand side as an $\mathbb{R}$-computable function is defined by case distinction we first have to put the definition of a solution to such an ODE more precisely. Very important for this is an understanding of the structure of the function $\mathbf{f}$ as described by the representation Theorem 1.

As the $\mathbb{R}$-machine for $\mathbf{f}$ executes its program the flow of control follows a certain computational path σ. The branching conditions along all such paths decompose the domain of $\mathbf{f}$, i.e. $\mathbb{R}^{N+1}$, into disjoint basic semi-algebraic sets D_σ which form the "basins of attraction" of the paths σ and shall be called *regions*. This is illustrated in Fig. 3 for the simple two-dimensional case.

The ith component $f_{\sigma,i}$ of the function $\mathbf{f}_\sigma := f_{\sigma,1} \times \cdots \times f_{\sigma,N}$ computed along σ can be described as a polynomial with real coefficients, thus it is a C^∞-function, i.e. infinitely often continuously differentiable. It is well known from analysis that for such a (local) problem $\mathbf{x}'(t) = \mathbf{f}_\sigma(t, x(t))$ with arbitrary initial value $(\tau, \xi) \in \mathbb{R}^{N+1}$ there exists a unique solution on a maximal open interval $I_\sigma \ni \tau$.

We now imagine a global solution analogous to the representation of $\mathbf{f}$ to be piece by piece composed of solutions to the local problems. The solution should be differentiable and satisfy the ODE inside a region D_σ with continuous transitions between regions.

Definition 6. We call a function $\mathbf{x} : I \to \mathbb{R}^N$ which is continuous on an interval $I \subset \mathbb{R}$ and satisfies $\mathbf{x}(t_0) = \mathbf{x}_0$ a *(global) solution*, if for all paths σ and all interior times $t \in int\, T_\sigma$ it is differentiable with derivative $\mathbf{x}'(t) = \mathbf{f}_\sigma(t, \mathbf{x}(t))$. Here $T_\sigma := \{t \mid (t, \mathbf{x}(t)) \in D_\sigma\}$. It is called *maximal* if there is no extension to an enlarged interval.

Lemma 7. *For every initial value problem according to Eq.* (1) *there exists a maximal global solution.*

Proof. Let σ_0 be the path with $(t_0, \mathbf{x}_0) \in D_{\sigma_0}$. The solution to the local problem $\mathbf{x}' = \mathbf{f}_{\sigma_0}(t, \mathbf{x})$ gives a global solution if we choose the interval I with $t_0 \in I \subset T_{\sigma_0}$ suitably. Surely there exists a (not necessarily unique) extension to a maximal interval. □

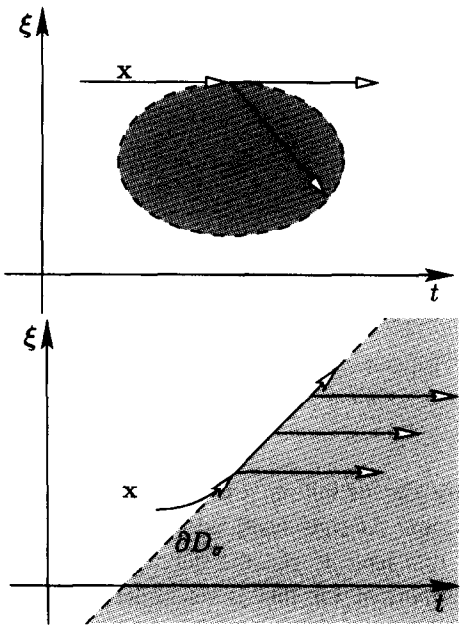

Fig. 4. Branching of tangential solution.

Of particular interest now are the conditions for the uniqueness of a maximal global solution. Under the following two conditions for the right-hand side[8] and the initial value we can prove that no branching of the solution occurs; the latter is necessary for computing the solution. Because of symmetry reasons we only consider the behavior of a solution *after* the initial time t_0.

1. The function $\mathbf{f}_\sigma$ defines for each point (t,ξ) the direction of a local solution through this point – we call this the *local vector field* of the region D_σ. If at the border of a region this direction is tangential to this border then the graph of a solution might touch this border and the solution might branch (cf. Fig. 4). Thus we only call those functions $\mathbf{f}$ *admissible* whose local vector fields are never tangent to a border.

   Given an admissible function $\mathbf{f}$ we can assume w.l.o.g. that every region is contained in the closure of its open interior: $D_\sigma \subset \overline{int\, D_\sigma}$; this is a non-degeneracy condition that does not influence the behavior of solutions in our sense.

   Finally, there is one more degenerate case we would like to avoid; it is illustrated in Fig. 5. The program that computes $\mathbf{f}(t,\xi)$ first checks for $t \leqslant 0$ and than for $t \geqslant 1/k$ with $k = 1,2,\ldots$ until it succeeds. Thus the borders of regions are dense in the neighborhood of the ξ-axis and a solution can cross infinitely many regions

[8] Strictly speaking, these are conditions for the *program*, not the function. But we will neglect this subtle difference to simplify notation.

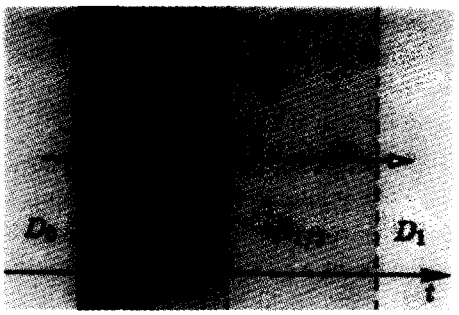

Fig. 5. Dense accumulation of regions.

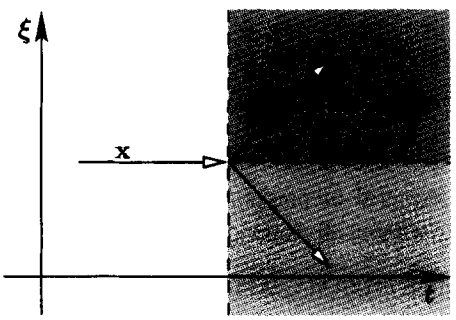

Fig. 6. Branching at a "multiple corner".

in finite time.[9] We therefore additionally require that every compact subset of an admissible function's domain intersects only finitely many regions.

2. If the initial value $(t_0, \mathbf{x}_0)$ is chosen in such a way that the unique local solution leaves its region in a "multiple corner", i.e. a point belonging to more than two borders, the extension need not be unique (cf. Fig. 6). Thus, we call initial values *suitable* for which no (maximal) global solution hits a multiple corner, not even in a far-away region.

Lemma 8. *For every initial value problem according to Eq.* (1) *with admissible* $\mathbf{f}$ *and suitable* $(t_0, \mathbf{x}_0)$ *there exists a unique maximal global solution which we denote by* $\mathbf{x}_{\mathbf{f},t_0,\mathbf{x}_0} : I_{\mathbf{f},t_0,\mathbf{x}_0} \to \mathbb{R}^{N+1}$.

Proof. Assume that two solutions $\mathbf{x}_1 : [t_0, t_1[\to \mathbb{R}^N$ and $\mathbf{x}_2 : [t_0, t_2[\to \mathbb{R}^N$ branch at time

$$t^* := \inf\{t \geqslant t_0 \mid \mathbf{x}_1(t) \neq \mathbf{x}_2(t)\} < \min\{t_1, t_2\}$$

and call this point $P := (t^*, \mathbf{x}_1(t^*)) = (t^*, \mathbf{x}_2(t^*))$ and, its unique region D_{σ_0}. Because local solutions are unique P cannot lie in the interior of a region but only on its

[9] One is reminded of Zeno's paradox.

boundary. On the other hand, it must not be a multiple corner and hence there is exactly one other region D_{σ_1} with $P \in \partial D_{\sigma_1}$ (remember that regions must not accumulate densely).

Now, consider for each curve $\mathbf{x}_i$ the regions $D_{\pi_i^-}$ and $D_{\pi_i^+}$ through which it passes immediately before and after t^* resp. The uniqueness of local solutions tells us that $\pi_1^- = \pi_2^-$ and $\pi_1^+ \neq \pi_2^+$ as both solutions started in the same way and then branched. So w.l.o.g. we can assume that $\mathbf{x}_1$ stays inside the region $D_{\pi_1^-}$ while $\mathbf{x}_2$ crosses from there to $D_{\pi_2^+}$. But this behavior of $\mathbf{x}_1$ is a contradiction to our precondition that $\mathbf{f}$ be admissible, i.e. that no local vector field is tangent to the border of a region. □

Theorem 9. *With the above preconditions and notations,* $\mathbf{x}_{\mathbf{f},t_0,\mathbf{x}_0}$ *is analytically* $\mathbb{R}$*-computable without division.*

Proof. Let $\mathcal{M}_\mathbf{f}$ be an $\mathbb{R}$-machine for the computation of $\mathbf{f}$, $\mathbf{x} := \mathbf{x}_{\mathbf{f},t_0,\mathbf{x}_0}$, and $I := I_{\mathbf{f},t_0,\mathbf{x}_0}$. We will construct an $\mathbb{R}$-machine $\mathcal{M}$ which computes, for a given time $t \in I$, the value $\mathbf{x}(t)$ of the solution.

We employ the explicit Euler-algorithm with step size $h := (t - t_0)(\frac{1}{2})^n$. It approximates the solution on the grid of points $\tau_k := t_0 + kh$, $0 \leqslant k \leqslant 2^n$, by $\mathbf{y}(\tau_k, h) \approx \mathbf{x}(\tau_k)$ which can be computed without division: $\mathbf{y}(\tau_0, h) := \mathbf{x}_0$ and

$$\mathbf{y}(\tau_{k+1}, h) := \mathbf{y}(\tau_k, h) + h\mathbf{f}(\tau_k, \mathbf{y}(\tau_k, h)). \tag{2}$$

The machine $\mathcal{M}$ outputs $\mathbf{y}(t,h)$ for $n \to \infty$ and thus $h \to 0$.

For symmetry reasons we only consider $t > t_0$ and have to show that $\lim_{h\to 0} \mathbf{y}(t,h) = \mathbf{x}(t)$. Let σ_0 be the path with $(t_0, \mathbf{x}_0) \in D_{\sigma_0}$, and we first assume that everything happens within that region. Furthermore let $B \subset \mathbb{R}^{N+1}$ denote the ball around $(t_0, \mathbf{x}_0)$ containing all $\mathbf{y}(\tau_k, h)$, and let $L > 0$ be a Lipschitz constant of $\mathbf{f}_{\sigma_0}$ on B with respect to ξ, i.e.

$$\forall (\tau, \xi_1), (\tau, \xi_2) \in B\colon |\mathbf{f}_{\sigma_0}(\tau, \xi_1) - \mathbf{f}_{\sigma_0}(\tau, \xi_2)| \leqslant L|\xi_1 - \xi_2|.$$

Then the global error of Euler's algorithm is $|\mathbf{y}(t,h) - \mathbf{x}(t)| = \mathrm{O}((h/L)\mathrm{e}^{(t-t_0)L})$ and the approximation converges for $h \to 0$ to the correct value.

We now consider the general case of the solution crossing a region's border. Assume that in $(t_1, \mathbf{x}_1)$ the local solution crosses the border from D_{σ_0} to D_{σ_1}. The approximation of this point be $(t_{1,h}, \mathbf{x}_{1,h})$, the first point generated by Euler's algorithm outside of D_{σ_0}. By the above considerations we know that $\lim_{h\to 0}(t_{1,h}, \mathbf{x}_{1,h}) = (t_1, \mathbf{x}_1)$.

For a sufficiently small step size h, $(t_{1,h}, \mathbf{x}_{1,h}) \in D_{\sigma_1}$ will hold. Because of the continuous dependency of the solution on the initial value the approximate solution starting in $(t_{1,h}, \mathbf{x}_{1,h})$ also converges to $\mathbf{x}$ as $h \to 0$. Thus the procedure can be extended correctly across the borders of regions. □

Main Theorem 1. *For every initial value problem according to Eq.* (1) *with admissible* $\mathbf{f}$ *and suitable* $(t_0, \mathbf{x}_0)$ *the unique maximal global solution is robustly* δ*-*$\mathbb{Q}$*-analytic without division.*

Proof. We proceed as in the proof of Theorem 9 while simulating the $\mathbb{R}$-machine $\mathscr{M}_{\mathbf{f}}$ as in Theorem 3. The main problem is to evaluate $\mathbf{f}$ with sufficient accuracy by means of a δ-$\mathbb{Q}$-machine and in finite time. Again, we first assume everything happens inside $D_{\sigma_0} \ni (t_0, \mathbf{x}_0)$ and then generalize.

From Eq. (2) we can deduce that $\mathbf{y}(t,h) = p_n(t_0, \mathbf{x}_0, t)$ where p_n is a polynomial with rational coefficients because $\mathbf{f}_{\sigma_0}$ is such a polynomial. The evaluation of $p_n(t_0, \mathbf{x}_0, t)$ with precision δ gives an approximation $p_n^{(\delta)}(t_0, \mathbf{x}_0, t)$ of $\mathbf{x}(t)$ of unknown quality as well as an error bound $\varepsilon_n^{(\delta)}(t_0, \mathbf{x}_0, t)$ *for the evaluation*, i.e.

$$|p_n^{(\delta)}(t_0, \mathbf{x}_0, t) - p_n(t_0, \mathbf{x}_0, t)| < \varepsilon_n^{(\delta)}(t_0, \mathbf{x}_0, t).$$

It is clear that $\lim_{\delta \to \infty} \varepsilon_n^{(\delta)}(t_0, \mathbf{x}_0, t) = 0$ and we would like to compute

$$\lim_{n \to \infty} \lim_{\delta \to \infty} p_n^{(\delta)}(t_0, \mathbf{x}_0, t) = \lim_{n \to \infty} p_n(t_0, \mathbf{x}_0, t) = \lim_{h \to 0} \mathbf{y}(t,h) = \mathbf{x}(t).$$

Unfortunately, a δ-$\mathbb{Q}$-machine may only compute a single limit for $\delta \to \infty$ and we need a little trick. We consider the inequality $\varepsilon_n^{(\delta)}(t_0, \mathbf{x}_0, t) < 2^{-n}$ which holds for every n and every $\delta \geqslant \delta_n$. The other way round, if we are given some δ we can search the maximal $n \leqslant \delta$ such that the inequality still holds and call this value n_δ. Note that the sequence of these n_δ monotonically increases with δ and tends to infinity. Then we construct a δ-$\mathbb{Q}$-machine $\mathscr{M}$ which in every phase outputs $p_{n_\delta}^{(\delta)}(t_0, \mathbf{x}_0, t)$ – from the above it should be clear that this converges as desired.

We now consider the general case of the solution crossing a region's border. Assume that in $(t_1, \mathbf{x}_1)$ the local solution crosses the border from D_{σ_0} to D_{σ_1}. The approximation of this point be $(t_{1,h,\delta}, \mathbf{x}_{1,h,\delta})$, the first point generated by Euler's algorithm which is – according to an evaluation of $\mathbf{f}$ with precision δ – outside of D_{σ_0}. For a sufficiently small step size h and large precision δ, $(t_{1,h,\delta}, \mathbf{x}_{1,h,\delta}) \in D_{\sigma_1}$ will hold. As above this point is given by a polynomial in the inputs $t_0, \mathbf{x}_0$, and t; again, we choose n such that the polynomial can be evaluated with accuracy 2^{-n}. Because of the continuous dependency of a solution on its initial value the approximate solution starting in $(t_{1,h,\delta}, \mathbf{x}_{1,h,\delta})$ also converges correctly as $\delta \to \infty$ and thus $n \to \infty$ and $h \to 0$. Thus the procedure can be extended correctly across the borders of regions. □

3.2. *Stability of dynamic systems*

Many natural or technological processes – e.g. the motion of the planets in our solar system – fundamentally depend on time. The aim of the general theory of dynamic systems according to [3] is to mathematically model such time-dependent processes, to describe their essential *qualitative* properties, and to predict these. Dynamic systems in a narrower sense are *homogeneous* in time, i.e. their development depends not on the initial time, but only on the initial state. One can model continuous dynamic systems by *autonomous* ODEs whose right-hand side $\mathbf{f}: \mathbb{R}^N \to \mathbb{R}^N$ does not depend on time and choose w.l.o.g. $t_0 = 0$.

An important aspect in the study of such systems is the question about the qualitative long-term behavior of a solution with respect to convergence or stability. Our focus is on the computability of such features of a solution if we are given an initial value and a program for an $\mathbb{R}$-machine that computes $\mathbf{f}$. By constructing ODEs from computations we can use our results about the undecidability of the "halting problem" for analytic $\mathbb{R}$-machines in this context. We show that the fundamental and apparently simple problem "Does the solution $\mathbf{x}(t)$ of an initial value problem converge for $t \to \infty$?" is undecidable for analytic $\mathbb{R}$-machines.

It is undecidable whether the one-dimensional output of an $\mathbb{R}$-machine $\mathcal{M} := \mathcal{M}_\pi^{\mathbb{R}}$ with empty input converges or not. The idea now is to constructively describe the analytic computation of $\mathcal{M}$ by an ODE whose solution at integral times corresponds to the output of $\mathcal{M}$ and in between interpolates it linearly. We choose $\xi_1(0) = 0$ and $\xi_1' = y_1^{(\lfloor t \rfloor + 1)} - y_1^{(\lfloor t \rfloor)}$ where $y_1^{(i)}$ is the output of $\mathcal{M}$ in the ith step and obtained by simulation. This defines a *non-autonomous* initial value problem (the right-hand side depends on time) with the desired property.

In the case of dynamic systems, however, we are restricted to autonomous right-hand sides and thus emulate time by an additional ODE $\xi_N' = -\ln 2 \cdot \xi_N$ with $\xi_N(0) = 1$. Then $\xi_N(t) = 2^{-t}$, $\lim_{t \to \infty} \xi_N = 0$, and $\xi_N \mapsto \lfloor t \rfloor = \lfloor \log_2 1/\xi_N \rfloor$ is $\mathbb{R}$-computable. If we substitute $\lfloor \log_2 1/\xi_N \rfloor$ for $\lfloor t \rfloor$ in the equation for ξ_1 we have a system of ODEs with $\mathbb{R}$-computable autonomous right-hand side that simulates the output of an analytic $\mathbb{R}$-machine in that it converges iff the latter does. We can pad the system to arbitrary size with equations $\xi_i' = 0$.

Theorem 10. *The problem of deciding whether the solution of an initial value problem*

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(0) = \mathbf{x}_0$$

with autonomous $\mathbb{R}$-computable right-hand side $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$ and $\mathbf{x}_0 \in \mathbb{R}^N$ converges as time approaches infinity, is not decidable for analytic $\mathbb{R}$-machines if $N > 1$.

Proof. It is clear from the above discussion that the convergence problem can be reduced to the mentioned problem. □

In a similar way, we can also show that it is undecidable for regular $\mathbb{R}$-machines whether the solution of an initial value problem is sensitive to small changes in the initial value or converges in the case $N = 1$.

Acknowledgements

The authors whish to thank the anonymous referees for their useful suggestions and Björn Schieffer for careful proof-reading and helpful discussions.

References

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

[2] L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, Bull. (New Series) Amer. Math. Soc. 21(1) (1989) 1–46 ICSI Technical Report TR-88-012.

[3] G. Grosche, E. Zeidler, D. Ziegler, V. Ziegler (Eds.), Teubner-Taschenbuch der Mathematik, vol. 2, B.G. Teubner Leipzig, 1995.

[4] R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (Eds.), Hybrid Systems, Lecture Notes in Computer Science, vol. 736, Springer, Berlin, 1993.

[5] A. Grzegorczyk, On the definition of computable real continuous functions, Fund. Math. 44 (1957) 61–71.

[6] G. Hotz, B. Schieffer, G. Vierke, Analytic Machines, Technical Report TR95-025, Electronic Colloquium on Computational Complexity, 1995. `http://www.eccc.uni-trier.de/eccc`.

[7] B. Schieffer, Diagnose komplexer Systeme am Beispiel eines Tank-Ballast-Systems, Dissertation, Universität des Saarlandes, Saarbrücken, Germany, 1996.

[8] C.L. Siegel, Vorlesungen über Himmelsmechanik, Number 85 in Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen, Springer, Berlin, 1956.

[9] G. Vierke, Berechenbarkeit reellwertiger Funktionen und analytische Berechnungen, Diplomarbeit, Universität des Saarlandes, Saarbrücken, Germany, July 1995.

[10] K. Weihrauch, Computability. EATCS Monographs on Theoretical Computer Science, vol. 9, Springer, Berlin, 1987.

[11] K. Weihrauch, A Foundation of computable analysis, in: K.-I Ko, K. Weihrauch (Eds.), Workshop on Computability and Complexity in Analysis, number 190-9/1995 in Informatikberichte, D-58084 Hagen, 1995. FernUniversität.