Analytic Machines

Thomas Chadzelek **Günter Hotz**

Technical Report 12/97 November 1997

e-mail: chadzelek@cs.uni-sb.de, hotz@cs.uni-sb.de
WWW: http://www-hotz.cs.uni-sb.de

Fachbereich Informatik

Fachbereich 14 Informatik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Analytic Machines

Günter Hotz* Thomas Chadzelek*

27th November 1997

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), dynamical systems described by ODEs and the undecidability of a class of stability problems for dynamical systems.

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 insofar real numbers have proven to be very practical.

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

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

use later to define a more concrete model of register machines over a ring or field. 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* 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.

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 $k_0 \in K_a$ holds additionally we call b *regular.*

For any given topology 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 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 $\Phi_{\mathcal{M}}^{(n)}(x) := y$.

We denote the *domain* of $\Phi_{\mathcal{M}}$ by $\mathbb{D}_{\mathcal{M}}$; the *halting set* $\mathbb{D}_{\mathcal{M}}^H \subset \mathbb{D}_{\mathcal{M}}$ contains exactly those inputs for which the computation of $\mathcal{M}$ is regular. Two machines $\mathcal{M}$ and $\mathcal{M}'$ are called *equivalent* if their halting sets and domains agree and $\Phi_{\mathcal{M}} = \Phi_{\mathcal{M}'}$.

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 $\mathbb{Z}$. The construction is similar to [9] and—concerning (finite) computability—equivalent to the model of Blum, Shub, and Smale [2].

These $\mathcal{R}$-machines (cf. figure 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 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.

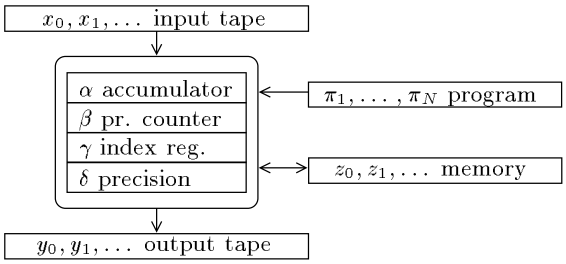

Figure 1: Structure of our register machine

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 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.

$$\begin{aligned}
K &:= \{k = (\alpha, \beta, \gamma, \delta, \pi, x, y, z) \text{ as above}\},\\
K_a &:= \{k \in K \mid \alpha = \gamma = \delta = 0,\ \beta = 1,\\
&\qquad \forall\, j:\ y_j = z_j = 0\},\\
K_e &:= \{k \in K \mid \pi_\beta = \texttt{end}\},\\
K_z &:= \{k \in K \mid \pi_\beta = \texttt{print}\}.
\end{aligned}$$

The input and output functions interpret x_0 and y_0 as the length of a sequence following in the next cells; in this way the machine operates on words from $\mathcal{R}^*$ rather than elements of the infinite direct sum $\mathcal{R}^\infty$. The set $\Omega = \Omega_N^{\mathcal{R}}$ contains the instructions in table 1; it depends on the size N of the program and the ring $\mathcal{R}$ but this is usually not denoted explicitly.

[1] The notion of alphabet is not confined to a *finite* set here.

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 $r \in \mathcal{R}$
   (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) $\gamma := \gamma + 1$, $\gamma := \gamma \dot{-} 1$
3. conditional branching ($m, n \in [1:N]$)
   `if` $\alpha > 0$ `then goto` m `else goto` n
4. special instructions
   `end`, `next` δ, `print`

Table 1: Instruction set

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 $\mathcal{R}$, a natural number N, and a program $\pi : [1:N] \to \Omega_N^{\mathcal{R}}$, we call the abstract machine $\mathcal{M}_\pi^{\mathcal{R}} = (K, K_a, K_e, K_z, \Delta, \mathcal{R}, in, out)$ uniquely defined by the above construction the $\mathcal{R}$-*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 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 $\Delta(k) := (0, 1, 0, \delta+1, \pi, x, y', z')$ with $\forall\, i :\ y_i' = z_i' = 0$. 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.

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 $\mathcal{M}_\pi^{\delta\text{-}\mathbb{Q}}$.

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. The figure 2 gives an overview of the hierarchy of classes of computable functions together with a hint to why the inclusion is strict. All classes below the line shown are closed under composition, but none above the line are.

$$
\begin{array}{ccc}
 & \vdots & \\
 & \mathbb{R}^2\text{-analytic} & \\
 & \subsetneq \quad \scriptstyle\text{hierarchy theorem} & \\
 & \mathbb{R}\text{-analytic} & \\
 & \subsetneq \quad \chi_{\mathbb{Q}}^{+} & \\
 & \delta\text{-}\mathbb{Q}\text{-analytic} & \\
 & \subsetneq \quad \chi_{\mathbb{Q}} & \\
 & \text{robustly } \delta\text{-}\mathbb{Q}\text{-analytic} & \scriptstyle\text{not closed} \\
\hline
 & \subsetneq & \scriptstyle\text{closed} \\
\text{strongly } \delta\text{-}\mathbb{Q}\text{-analytic} & \subsetneq & \text{quasi-strongly } \delta\text{-}\mathbb{Q}\text{-analytic} \\
= \text{`comput. real cont.'} & \scriptstyle\text{continuity} & \subsetneq \quad \scriptstyle\text{Koch's curve} \\
 & & \mathbb{R}\text{-computable} \\
 & & \subsetneq \quad \scriptstyle\text{domain} \\
 & & \mathbb{Q}\text{-computable} = \text{Turing-computable}
\end{array}
$$

Figure 2: Hierarchy of classes of computable functions

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

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 $\mathcal{M}_2 \circ \mathcal{M}_1$ of two robust analytic δ-$\mathbb{Q}$-machines by a single machine $\mathcal{M}$ fails when $\mathcal{M}_2$ 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 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.

Definition 5. Let $(k_{i_j})_{j=0}^{\infty}$ be the subsequence of target configurations of a δ-$\mathbb{Q}$-analytic computation. By $out(k_{i_j}) = (y_0^{(j)}, \ldots, y_{n_j}^{(j)}) \in \mathbb{Q}^*$ we denote the outputs and by $y_i := \lim_{j\to\infty} y_i^{(j)}$, the limes of the ith position. The computation is then called *quasi-strongly δ-$\mathbb{Q}$-analytic* iff

1. $y_0 = 0$;
2. $|y_i - y_i^{(j)}| \leq y_0^{(j)}$ for *almost all i, j.*

We regard the limes $(y_1, \ldots, y_n)$ 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 $D = \mathbb{D}_{\mathcal{M}}$ and for each $x \in D$ the computation of $\mathcal{M} := \mathcal{M}_{\pi}^{\delta\text{-}\mathbb{Q}}$ starting with $in(x)$ is quasi-strongly δ-$\mathbb{Q}$-analytic with result $f(x)$. $\square$

Note that if we requested the precision bound to hold *always* then the computed function would become *continuous*; we call such functions *strongly δ-$\mathbb{Q}$-analytic* and this 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 $\mathcal{M}_f$ and $\mathcal{M}_g$ the 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 $\mathcal{M}$ alternately simulates one phase of $\mathcal{M}_f$ on the original input x and one phase of $\mathcal{M}_g$ on the approximation of $f(x)$ computed so far if the precision bound is suitable else $\mathcal{M}_g$ waits. One observes that $\mathcal{M}_g$ 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 $\mathcal{M}$ 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 completely different proof can be found in [6, 9].

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

Proof. The δ-$\mathbb{Q}$-machine $\mathcal{M}'$ simulates the $\mathbb{R}$-machine $\mathcal{M}$ for the given function by interval arithmetic and with increasing precision δ. In doing so all cells of the memory and output tape as well as the accumulator of $\mathcal{M}$ are recreated by lower and upper bounds in the memory of $\mathcal{M}'$; they are 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 $[\alpha - 2^{-\delta},\ \alpha + 2^{-\delta}]$. 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.

Please note that in this way the program remains correct with regards to illegal $\alpha := \alpha^{-1}$. We can avoid endless loops by branching at most δ times in each phase. The precision bound is wrong at most until $\mathcal{M}'$ starts simulating the right computational path of $\mathcal{M}$. If the content of α is non-zero at a branching then the interval computed by $\mathcal{M}'$ 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 $\mathcal{M}'$—we call this approach *conservative branching*. Thus it is clear that every finite computational path of $\mathcal{M}$ will be simulated by $\mathcal{M}'$ after a finite time and then the output converges as desired. □

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 of an $\mathcal{R}$-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 $\mathcal{R}$-machines an $\mathcal{R}^i$-analytic machine and speak of $\mathcal{R}^i$-analytic functions etc., we can summarize the following results. We do not give proofs here but merely cite these for later use.

Theorem 4. *The convergence problem of $\mathbb{R}^i$-analytic machines is not $\mathbb{R}^i$-analytic but $\mathbb{R}^{i+1}$-analytic; the same holds for δ-$\mathbb{Q}^i$-analytic machines.*

Proof. The undecidability of the problem by the same type of machine follows from a more or less simple diagonalization argument. We have a constructive proof of how to simulate i machines on $i+1$ while deciding convergence of the output. □

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 $(t, \boldsymbol{\xi}) \mapsto \mathbf{f}(t, \boldsymbol{\xi})$ 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

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

3.1 Solving ODEs

Now we shall clarify the definition, existence, uniqueness, and computability of solutions to equation 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 figure 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 $(\tau, \boldsymbol{\xi}) \in \mathbb{R}^{N+1}$ there exists a unique solution on a maximal open interval $I_\sigma \ni \tau$.

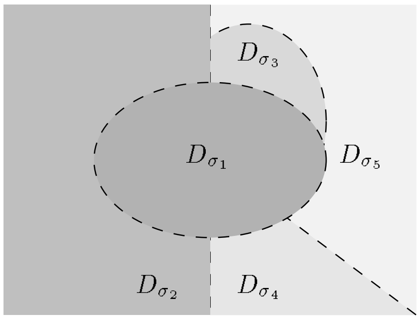

Figure 3: Partitioning of domain $\mathbb{R}^2$ into regions

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 (solution). 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 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 5 (existence). *For every initial value problem according to equation 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 $t_0 \in I \subset T_{\sigma_0}$ suitably. Surely there exists a (not necessarily unique) extension to a maximal interval. □

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

1. The functions $\mathbf{f}_\sigma$ each define a vector field in $\mathbb{R}^{N+1}$ by assigning to each point $(t, \boldsymbol{\xi})$ the direction $(1, \mathbf{f}_\sigma(t, \boldsymbol{\xi}))$ of a tangent to a local solution through this point. If at the border of a region the local vector field is tangential to

this border then the graph of a solution might touch this border or even follow it for a while and the solution might branch (cf. figure 4). Thus we only call those functions $\mathbf{f}$ *admissible* whose local vector fields are never tangent to a border.

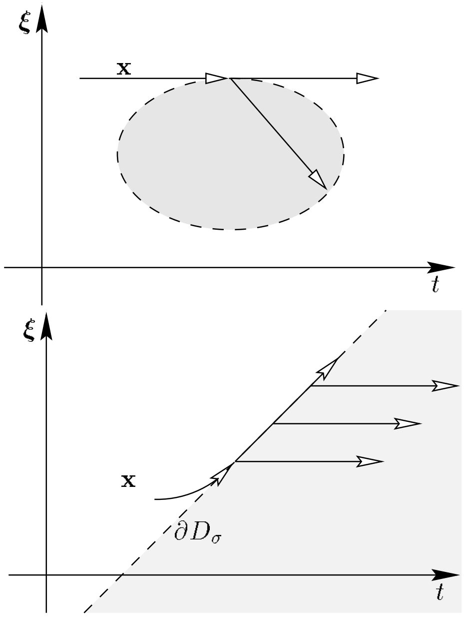

Figure 4: Branching of tangential solution

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 conditions that does not influence the behavior of solutions in our sense.

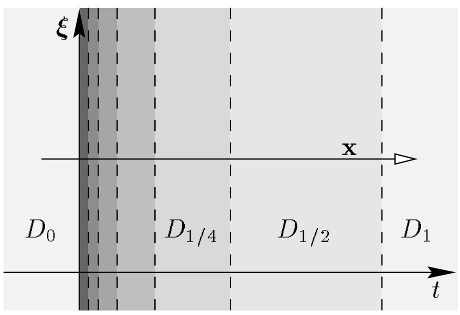

Figure 5: Dense accumulation of regions

Finally there is one more degenerate case we would like to avoid; it is illustrated in figure 5. The program that computes $\mathbf{f}(t, \boldsymbol{\xi})$ first checks for $t \le 0$ and than for $t \ge 1/k$ with $k = 1, 2, \ldots$ until it succeeds. Thus the borders of regions are dense in the neighborhood of the $\boldsymbol{\xi}$-axis and a solution can cross infinitely many regions in finite time[2]. 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. figure 6). Thus we call such initial values *suitable* for which no (maximal) global solution hits a multiple corner, not even in a far-away region.

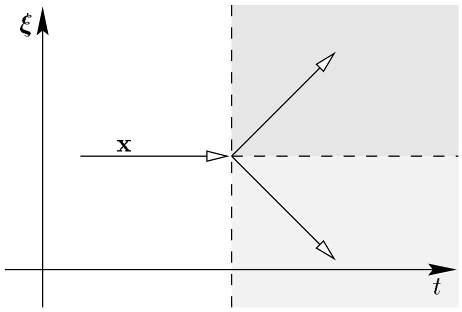

Figure 6: Branching at a "multiple corner"

Theorem 6 (uniqueness). *For every initial value problem according to equation 1 with admissible* $\mathbf{f}$ *and suitable* $(t_0, \mathbf{x}_0)$ *there exists a unique maximal global solution.*

Proof. By contradiction and case distinction. □

We denote such a unique solution by $\mathbf{x}_{\mathbf{f},t_0,\mathbf{x}_0} : I_{\mathbf{f},t_0,\mathbf{x}_0} \to \mathbb{R}^{N+1}$ where $I_{\mathbf{f},t_0,\mathbf{x}_0} \subset \mathbb{R}$ is an interval.

Theorem 7 (computability). *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 $I := I_{\mathbf{f},t_0,\mathbf{x}_0}$.

[2] One is reminded of ZENO's paradox.

We employ the explicit EULER-algorithm for the construction of an $\mathbb{R}$-machine $\mathcal{M}$ which computes, for a given time $t \in I$, the value $\mathbf{x}(t)$ of the solution. Using a step size of $h := (t - t_0)(\frac{1}{2})^n \to 0$ with $n \to \infty$ we can construct the polygonal chains $\mathbf{x}_h$ without division. For symmetry reasons we only consider $t > t_0$ and have to show that

$$\forall t \in I: \lim_{h \to 0} \mathbf{x}_h(t) = \mathbf{x}(t).$$

Let σ_0 be the path with $(t_0, \mathbf{x}_0) \in D_{\sigma_0}$, and we first assume the graph of $\mathbf{x}|_{[t_0,t]}$ to be completely contained in D_{σ_0}. Furthermore let $B \subset \mathbb{R}^{N+1}$ denote the ball around $(t_0, \mathbf{x}_0)$ containing the graphs of all $\mathbf{x}_h$, and $L > 0$ be a LIPSCHITZ constant of $\mathbf{f}_{\sigma_0}$ on B with respect to $\boldsymbol{\xi}$, i.e.

$$\forall (\tau, \boldsymbol{\xi}_1), (\tau, \boldsymbol{\xi}_2) \in B: \quad |\mathbf{f}_{\sigma_0}(\tau, \boldsymbol{\xi}_1) - \mathbf{f}_{\sigma_0}(\tau, \boldsymbol{\xi}_2)| \leq L|\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2|.$$

Then the global error of EULER'S algorithm is $|\mathbf{x}_h(t) - \mathbf{x}(t)| = O(\frac{h}{L} e^{(t-t_0)L})$ 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. In $(t_1, \mathbf{x}_1)$, the local solution crosses the border from D_{σ_0} to D_{σ_1}; this is by our assumptions no multiple corner and the solution is not tangential to the border. 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}. 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 polygonal chains starting in $(t_{1,h}, \mathbf{x}_{1,h})$ also converge to $\mathbf{x}$. Thus the procedure can be extended correctly across the borders of regions. □

We can improve this theorem by observing that the above construction together with the simulation theorem show the solution to be robustly δ-$\mathbb{Q}$-computable without division.

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 $\xi_N' = -\log 2 \cdot \xi_N$ with $\xi_N(0) = 1$. Then $\xi_N(t) = 2^{-t}$, $\lim_{t\to\infty} \xi_N = 0$, and $\xi_N \mapsto \lfloor t \rfloor = \lfloor \log_{1/2} 1/\xi_N \rfloor$ is $\mathbb{R}$-computable. If we substitute $\lfloor \log_{1/2} 1/\xi_N \rfloor$ 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 8. *The problem of deciding whether the solution of an initial value problem*

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

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. $\square$

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 $N = 1$.

References

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

[2] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, 21(1):1–46, July 1989. ICSI technical report TR-88-012.

[3] G. Grosche, E. Zeidler, D. Ziegler, and V. Ziegler, editors. *Teubner-Taschenbuch der Mathematik*, volume 2. B.G. Teubner Leipzig, 1995.

[4] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems.* Number 736 in Lecture Notes in Computer Science. Springer-Verlag, 1993.

[5] A. Grzegorczyk. On the definition of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.

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

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

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

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

[10] K. Weihrauch. *Computability.* Number 9 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1987.

[11] K. Weihrauch. A foundation of computable analysis. In K.-I. Ko and K. Weihrauch, editors, *Workshop on Computability and Complexity in Analysis*, number 190-9/1995 in Informatikberichte, D-58084 Hagen, 1995. Fern-Universität.