Annales UMCS Informatica AI 7 (2007) 37-47

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

The Extended Analog Computer and Turing machine

Monika Piekarz*

Department of Computer Science, Institute of Mathematics, Maria Curie-Skłodowska University, pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland

Abstract

In this paper we compare computational power of two models of analog and classical computers. As a model of analog computer we use the model proposed by Rubel in 1993 called the Extended Analog Computer (EAC) while as a model of classical computer, the Turing machines. Showing that the Extended Analog Computer can robustly generate result of any Turing machine we use the method of simulation proposed by D.S. Graça, M.L. Campagnolo and J. Buescu [1] in 2005.

1. Introduction

In the theory of analog computation two different families of models of analog computation that come from different behaviors of computation processes in time have been considered. The first family includes the models of computation on real numbers but in discrete time. These are, for example, the Analog Recurrent Neural Network [2] and the machines of Blum-Shub-Smale [3]. The second case are the models of computation on real numbers and in continuous time. In this scope the important model is the General Purpose Analog Computer proposed by Shannon in 1941. This model is able to generate all the differentially algebraic (DA) functions, and it is built from some units connected together. There also exists an extension of this model called the Extended Analog Computer. This model was defined by Rubel in 1993 [4] and was proposed as a model of a human brain. In this work we are interested in comparison of the computational power of the EAC and the Turing machine.

P. Koiran and C. Moore [5] showed that finite dimensional analytic maps are capable of simulating the transition function of an arbitrary Turing machine. These maps can simulate one step of the transition function in this manner that some computational hybrid system is used, which is continuous with respect to

* *E-mail*: mpiekarz@hektor.umcs.lublin.pl

the state space, but evolves discretely in time. Another approach has been given by simulation of the evolution of the Turing machine with continuous flows in $\mathbb{R}^n$ [6,7]. The whole nonanalytic form of an iteration of the map that simulates the transition function of the Turing machine was proposed in [8]. Moreover, it is known that the analytic differentially algebraic functions, which include most of usual mathematical analytic functions, are not closed under iteration [9]. It suggests that continuous-time computational models, which are closed under iteration, must contain some non-analytic functions. However, when we think about the iteration functions in the vicinity of integrals we can use the system of differential equations $y' = g_M(y,t)$ proposed by D.S. Graça, M.L. Campagnolo, J. Buescu in [10]. Here g_M is analytic and t represents the time steps of the Turing machine, in this case this system is able to simulate robustly the Turing machine.

2. Preliminaries

In the classical theory of computability several models of computation on natural numbers have been considered [11]. One of the most important ones is the Turing machine [11]. We will use it here in the standard way. The Turing machine is defined as a following tuple: $(\Sigma, Q, \delta, q_0, F)$, where Σ is a set of type symbols and Q is a set of states, $\delta: \Sigma \times Q \to \Sigma \times Q \times \{-1,0,+1\}$ is a partial function called a transition function, and -1 and $+1$ are the symbols which indicate the left and the right side movement of the head and 0 indicates no move of the head, q_0 and F denote the initial and the final states, respectively.

Next we recall some part of the theory of analog computation. We will start from the basic continuous time model of analog computation called the General Purpose Analog Computer [12]. In this model we have five based units connected together into various shapes of circuits. The units used in the GPAC are the following: *Constant multiplier*, *Adder*, *Multiplier*, *Constant function* and *Integrator* – a unit with a setting for the initial conditions – two constants a and t_0; two inputs – unary functions u, v; one output – the Riemann-Stieljes integral $\lambda t.\int_{t_0}^{t} u(x)dv(x) + a$. Pour-El shows, although with some corrections made by Lipshitz and Rubel in [13], the following:

Theorem 1. *If y is generable on I by a GPAC, then there is a closed subinterval $I' \subseteq I$ with non-empty interior such that, on I', y is differentially algebraic.*

Notice that not every function from the class of the recursive functions defined over the natural number can be computed by the GPAC, for example $\lambda xn.\exp^{[n]}(x)$ [14].

The main model of this paper is proposed by L.A. Rubel in 1993 and called Extended Analog Computer (EAC) [4]. The EAC works on a hierarchy of levels. It has no inputs from the outside, but it has got a finite number of "settings", which may be arbitrary real numbers. The outputs of the machine at level $(n-1)$ can be used as inputs at level n or higher where by the following operations: *addition*, *multiplication*, *composition*, *inversion*, *differences*, *analytic continuation*, *solving a differential equation with boundary-value requirements* and *taking of limits*, next results are generated. At the lowest level, level 0, it produces real polynomials of any finite number of real variables. However at level 1, it produces differentially algebraic real-analytic functions (C^{ω}) of any finite number of real variables.

Below there is the definition of the EAC in terms of the real functions. Let us start with some useful notation of analytical analysis. By $y \in C^{\omega}(\Omega)$ we mean that y has an extension of y^* to an open supersubset Ω^* of Ω, and y^* is real-analytic on Ω^*. These are the functions that are locally the sum of convergent power series. This definition is the mathematical description of the definition of the EAC proposed by Rubel in [4].

Definition 2. *The function* $y \in C^{\omega}(\Omega)$, $\Omega \subset \mathbb{R}^k$ *is generated by an EAC on some level n,* ($y \in EAC_n$) *if y is a function, which holds the following conditions*:

1. *For* $n = 0$

$$y(\overline{x}) = \sum_{(\alpha_1,\alpha_2,\ldots,\alpha_k)\in A} c_{\alpha_1\alpha_2\ldots\alpha_k} \prod_{i=1}^{k} x_i^{\alpha_i}$$

where $A \subset N_0^k$, $c_{\alpha_1\alpha_2\ldots\alpha_k}$ *are fixed real constants and* $\overline{x} = x_1, x_2 \ldots, x_k$.

2. *For* $n > 0$, *y is defined by one of the following methods:*
   - $y(\overline{x}) = u_1(\overline{x}) + u_2(\overline{x})$, *where* $u_1, u_2 \in EAC_{n-1}$;
   - $y(\overline{x}) = u_1(\overline{x})u_2(\overline{x})$, *where* $u_1, u_2 \in EAC_{n-1}$;
   - $y(\overline{x}) = v(u_1(\overline{x}), u_2(\overline{x}), \ldots, u_l(\overline{x}))$, *where* $v, u_1, u_{2,\ldots,} u_l \in EAC_{n-1}$;
   - $y(\overline{x}) = y_i(\overline{x})$ *for* $i = 1,2,\ldots, l$, *where* $y_1(\overline{x}), y_2(\overline{x}),\ldots,y_l(\overline{x})$ *is* C^{ω} – *functions solution of*

$$\begin{cases} f_1(\overline{x}, y_1, y_2, \ldots, y_l) = 0 \\ f_2(\overline{x}, y_1, y_2, \ldots, y_l) = 0 \\ \vdots \\ f_l(\overline{x}, y_1, y_2, \ldots, y_l) = 0 \end{cases}$$

where $f_1, f_2, \ldots, f_l \in EAC_{n-1}$;[1]

- $y(\overline{x}) = Df(\overline{x})$, *where* $Df = \dfrac{\partial^{\alpha_1+\alpha_2+\ldots+\alpha_k} f(\overline{x})}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_k^{\alpha_k}}$ *are partial derivatives of* f *and* $f \in EAC_{n-1}$;
- $y = y^*_{|\Omega}$, *where* $\Omega^* \subset \Omega$, $\Omega^* \in EAC_{n-½}$ *and* $y^* \in EAC_n$;
- $y(\overline{x}) = v^*(\overline{x})$ *if for* $v^* \in EAC_n$ *is an analytic continuation of* v *from* $\Omega \cap \Omega^*$ *to all of* Ω, *where* $v \in EAC_n$, $\Omega \in EAC_{n-½}$ *and* $\Omega \cap \Omega^* \neq \phi$;
- $y(\overline{x})$ *is a solution of equations* $F_i(\overline{x} : y, y_1, y_2, \ldots, y_l) = 0$, *for* $i = 1, \ldots, k$ *on a set* Ω *which are subject to certain boundary values requirements*[2], *where* $\Omega \in EAC_{n-½}$, $F_i \in EAC_{n-1}$ *and* $y_1, y_2, \ldots, y_l$ *are the partial derivatives of* y;
- *for all* $\overline{x}_0 \in \Omega$,

$$y(\overline{x}_0) = \lim_{\overline{x} \to \overline{x}_0, \overline{x} \in \Lambda} f(\overline{x}),$$

$$\frac{\partial^{\alpha_1+\alpha_2+\ldots+\alpha_k} y(\overline{x})}{\partial x_1^{\alpha_1}, \partial x_2^{\alpha_2}, \ldots, \partial x_k^{\alpha_k}} = \lim_{\overline{x} \to \overline{x}_0, \overline{x} \in \Lambda} \frac{\partial^{\alpha_1+\alpha_2+\ldots\alpha_k} f(\overline{x})}{\partial x_1^{\alpha_1}, \partial x_2^{\alpha_2}, \ldots, \partial x_k^{\alpha_k}}$$

where $\Omega \in EAC_{n-½}$ *is a subset of* $\partial\Lambda$ (*is the edge of* Λ), $\Omega \in EAC_{n-½}$ *an* $f \in EAC_{n-1}$ *defined on* Λ.

The Extend Analog Computer defined above has some interesting properties. The EAC is that one which expands the scope of the General Purpose Analog Computer (GPAC) [12]. This machine can compute all the functions which can be computed by the GPAC and all the differential algebraic functions. In particular the EAC can compute some base analytic functions like the exponential function or the trigonometric functions [4].

Example 3. *The pair of trigonometric functions* (*sin, cos*) *can be obtained by an EAC by solving the following differential equations with boundary-value requirements:*

$$sin(0) = 0,\ cos(0) = 1,\ cos(0) - \partial_x sin(x) = 0,\ sin(0) + \partial_x cos(x) = 0.$$

From Theorem 1 it follows that all functions generated by the GPAC are the differentially algebraic functions which can be obtained, by the EAC on level 1.

On the higher levels of the EAC one can obtain such functions as Γ-Eulers function or ξ-Riemmans function [4], which cannot be obtained by the GPAC.

---

[1] We require these functions to be well-defined C^ω function on Ω. For example the equation $xy - 1 = 0$ has the solution $y = 1/x$ which is not well-defined on R (because it is not defined for $x = 0$) but it is well-defined on the intervals $(-\infty, 0)$ and $(0, \infty)$. So $y = 1/x$ is not EAC computable on R but is EAC computable on $(-\infty, 0)$ or on $(0, \infty)$.

[2] For example: $y = y_0$ on a piece γ_0 of the boundary of Ω, where we use only functions $y_0 \in EAC_{n-1}$.

Moreover, it is shown in [4] that the solution of the Dirichlet's problem for the Laplace's equation in the unit disc can be computed by the EAC.

Moreover, the EAC is required to produce unique outputs that are close, on a compact set, to the original unique outputs, when the inputs are slightly varied [4].

In the following Lemmas some functions of [1], which will be useful for a simulation of the Turing machine by EAC, are presented.

Lemma 4. *Let* $n \in \mathbb{N}$ *and let* $\varepsilon \in (0, 1/2)$. *Then there is a factor* $\varsigma_\varepsilon > 0$ *such that, for* $\forall x \in [n-\varsigma_\varepsilon, n+\varsigma_\varepsilon] \Rightarrow |\nu(x) - n \bmod 10| \le \varepsilon$ *and* ν *is uniformly continuous in R, the EAC-computable function given by* $\nu(x) = a_0 + a_5\cos(\pi x) + \left(\sum_{j=1}^{4} a_j \cos\left(\frac{j\pi x}{5}\right) + b_j \sin\left(\frac{j\pi x}{5}\right)\right)$, *where* a_0, ..., a_5,b_1, ..., b_5 *are computable coefficients that can be obtained by solving a system of linear equations.*

The function from the above lemma is an analytic extension ν: $\mathbb{R} \to \mathbb{R}$ of the function g: $\mathbb{N} \to \mathbb{N}$ defined by $g(n) = n \bmod 10$. From the form of ν we can see easily that it is EAC-computable as a composition of the sum function and the trigonometric functions.

Lemma 5. *Let* $n \in \mathbb{Z}$ *and let* $\varepsilon \in [0, 1/2)$. *Then there is some contracting factor* $\lambda_\varepsilon > 0$ *such that, for* $\forall \delta \in [-\varepsilon, \varepsilon], |\sigma(n+\delta) - n| \le \lambda_\varepsilon \delta^3$ [3] *and* σ *is the EAC-computable function given by* $\sigma(x) = x - 0.2\sin(2\pi x)$.

The function σ is a contraction on the vicinity of integers needed to keep the error of the simulation under control. Of course σ is EAC-computable as a composition of the difference function and the sinus function.

Lemma 6. *Let* $a \in \{0,1\}$ *and* $\overline{a}$, $y \in \mathbb{R}$ *satisfying* $|a - \overline{a}| \le 1/4$ *and* $y > 0$. *Then there is a function* l_2: $\mathbb{R}^2 \to \mathbb{R}$ *such that* $|a - l_2(\overline{a}, y)| < 1/y$ *and* l_2 *is the EAC-computable function given by* $l_2(x,y) = \frac{1}{\pi}\arctan(4y(x - 1/2)) + \frac{1}{2}$.

[3]Through the remain part of this paper we suppose that $\varepsilon \in [0, 1/2)$ is fixed and that λ_ε is the corresponding contracting factor given by Lemma 5.

Lemma 7. *Let* $a \in \{0,1,2\}$ *and* $\overline{a}$, $y \in \mathbb{R}$ *satisfying* $|a-\overline{a}| \le \varepsilon$ *and* $y \ge 2$. *Let* $d = \begin{cases} 0 & \varepsilon \le 1/4 \\ \lceil -\log(4\varepsilon)/\log \lambda_\varepsilon \rceil & \varepsilon > 1/4 \end{cases}$. *Then there is a function* $l_3 : \mathbb{R}^2 \to \mathbb{R}$ *such that* $|a - l_3(\overline{a}, y)| < 1/y$ *and* l_3 *is the EAC-computable function given by* $l_3(x,y) = l_2\left(\left(\sigma^{[d+1]}(x)-1\right)^2, 3y\right)\left(2l_2\left(\sigma^{[d]}(x)/2, 3y\right)-1\right)+1$.

The functions from the above two lemmas are other needed error contracting functions that control the error for unbounded quantities. The l_2 function is EAC-computable as a composition of some base analytic functions (sum, difference, product) and the arctan function, which is computable by the EAC as the DA function [4] The l_3 function is the EAC computable too as a composition of the EAC-computable functions.

Lemma 8. *If* $|\alpha_i|, |\overline{\alpha}_i| \le K$ *for* $i = 1, ..., n$ *then*

$$|\alpha_1 \ldots \alpha_n - \overline{\alpha}_1 \ldots \overline{\alpha}_n| \le \left(|\alpha_1 - \overline{\alpha}_1| + \ldots + |\alpha_n - \overline{\alpha}_n|\right) K^{n-1}.$$

All above presented lemmas will be used to keep the error under control during the simulation of the Turing machine.

3. The Turing machine simulation by the EAC

The ideas and concepts related to the simulation due to Graça, Campagnolo and Buescu of [10] are recalled here. They code configuration of the Turing machine into a triple $(x, y, z) \in \mathbb{N}^3$ without loss of generality. They used only 10 symbols, the blank symbol $B = 0$, and symbols 1,2,...,9. The string

$$\ldots BBBa_{-k}a_{-k+1}\ldots a_{-1}a_0a_1\ldots a_nBBB\ldots$$

represents the tape content for a given Turing machine M. The head is reading symbols $a_i \in \{0,1,\ldots,9\}$ for all i. Moreover, M has m states, represented by numbers 1 to m. In all transitions, the head either moves to the left, moves to the right, or does not move. For convenience, they consider that if the machine reaches a halting state it moves to the same configuration. Take

$$y_1 = a_0 + a_1 10 + \ldots + a_n 10^n, y_2 = a_{-1} + a_{-2}10 + \ldots + a_{-k}10^{k-1}$$

and let q be the state associated to the current configuration. The triple $(x, y, z) \in \mathbb{N}^3$ gives the current configuration of M.

Our intention in this paragraph is to establish the fact that the EAC can use such simulations of the Turing machine. We want to make this in two steps. Firstly we need an EAC simulation of one step of the Turing machine. Secondly

we need an EAC-computable function which iterates the simulation of one step of Turing machine. Let us cite Theorem 1 from work [10]: Let θ: $\mathbb{N}^3 \to \mathbb{N}$ be the transition function of the Turing machine *M*, under the encoding described above and let $0 < \delta < \varepsilon < 1/2$. Then θ admits an analytic extension f_M: $\mathbb{R}^3 \to \mathbb{R}^3$, robust to perturbations in the following sense: for all f such that $\|f - f_M\|_\infty \le \delta$ and for all $\overline{x}_0 \in \mathbb{R}^3$ satisfying $\|\overline{x}_0 - x_0\|_\infty \le \varepsilon$, where $x_0 \in \mathbb{N}^3$ represents an initial configuration,

$$\left\| f^{[j]}(\overline{x}_0) - \theta^{[j]}(x_0) \right\|_\infty \le \varepsilon .$$

Theorem 9. *The function f_M defined as above is EAC-computable.*

Proof. We will show how to construct f_M with EAC-computable functions.

Suppose that (y_1,y_2,q) describes the present configuration and that $(\overline{y}_1,\overline{y}_2,\overline{q})$ is an approximation of this configuration. First we will want $\overline{f}_M$ to satisfy the following condition:

$$\left\|(y_1,y_2,q) - (\overline{y}_1,\overline{y}_2,\overline{q})\right\|_\infty \le \varepsilon \Rightarrow \left\|\theta(y_1,y_2,q) - \overline{f}_M(\overline{y}_1,\overline{y}_2,\overline{q})\right\|_\infty \le \varepsilon .$$

Let a_0 be the symbol being actually read by the Turing machine M. Then $\nu(y_1) = a_0$. Since $|y_1 - \overline{y}_1| \le \varepsilon$ then $\left|a_0 - \nu \circ \sigma^{[l]}(\overline{y}_1)\right| \le \varepsilon$, with $l = \left\lceil \left| \frac{\log(\zeta_\varepsilon / \varepsilon)}{\log \lambda_\varepsilon} \right| \right\rceil$[4], where ν and σ are the functions from Lemma 4 and Lemma 5. So $\overline{y} = \nu \circ \sigma^{[l]}(\overline{y}_1)$ gives us an approximation of the symbol being currently read. Similarly, $\nu \circ \sigma^{[l]}(\overline{y}_2)$ gives an approximation of a_{-1}, with the error bounded by ε. These approximations can be computed for constant l by certain EAC.

Now we determine the next state. Recall that m denotes the number of states and $k = 10$ is the number of symbols. This can be done as follows:

$$\overline{q}_{next} = \sum_{i=0}^{9} \sum_{j=1}^{m} \left(\prod_{r=0 r \ne i}^{9} \frac{\left(\sigma^{[n]}(\overline{y}) - r\right)}{(i-r)} \right) \left(\prod_{s=1 s \ne j}^{m} \frac{\left(\sigma^{[n]}(\overline{y}) - s\right)}{(j-s)} \right) q_{i,j}$$

with $n = \left\lceil \frac{\log\left(10m^2K^{m+7}(m+8)\right)}{-\log \lambda_\varepsilon} \right\rceil$, $K = \max\left\{9.5, m + \frac{1}{2}\right\}$, where $q_{i,j}$ is the state that follows symbol i and state j. With this choice for n we have $9\left|y - \sigma^{[n]}(\overline{y})\right| + (m-1)\left|q - \sigma^{[n]}(\overline{q})\right| \le \varepsilon / 10m^2K^{m+7}$. From this and Lemma 8 we conclude that $|\overline{q}_{next} - q_{next}| \le \varepsilon$.

[4] $\lceil x \rceil = \min\{n \in \mathbb{Z}: x \le n\}$

The map that returns the next state is defined by the polynomial interpolation with the function σ composed of n times, so can be computed by the EAC.

To determine the symbol to be written on the tape and the direction of the move of the head we use similar constructions. We mark then as $\overline{s}_{next}$ and $\overline{h}_{next}$ (where $h = 0$ denotes a move to the left, $h = 1$ denotes a "no move", and $h = 2$ denotes a move to the right).

To update the tape contents we define the following functions P_1, P_2, P_3, which are intended to approximate the tape contents after the head moves left, does not move, or moves right, respectively. Let H be an additional approximation of $\overline{h}_{next}$ determined later. Then, the next value of y_1, can be approximated by

$$\overline{y}_1^{next} = P_1 \frac{1}{2}(1-H)(2-H) + P_2 H(2-H) + P_3\left(-\frac{1}{2}\right)H(1-H),$$

with

$$P_1 = 10\left(\sigma^{[p]}(\overline{y}_1) + \sigma^{[p]}(\overline{s}_{next}) - \sigma^{[p]}(\overline{y})\right) + \sigma^{[p]} \circ \nu \circ \sigma^{[l]}(\overline{y}_2),$$

$$P_2 = \sigma^{[p]}(\overline{y}_1) + \sigma^{[p]}(\overline{s}_{next}) - \sigma^{[p]} \circ \nu \circ \sigma^{[l]}(\overline{y}_1),$$

$$P_3 = \frac{\sigma^{[p]}(\overline{y}_1) - \sigma^{[p]}(\overline{y})}{10},$$

where $p \in \mathrm{N}$ is sufficiently large. We have problem because P_1 depends on $\overline{y}_1$, which is not a bounded value. Thus, if we simply take $H = \overline{h}_{next}$, the error of the term $(1 - H)(2 - H)/2$ is arbitrary amplified when this term is multiplied by P_1. To mark sufficiently good H, proportional to y_1 we use the functions l_2 and l_3 from Lemma 6 and Lemma 7 and put:

$$H = l_3\left(\overline{h}_{next}, 10000(\overline{y}_1 + 1/2) + 2\right)$$

Using the same argument for P_2 and P_3, we conclude that $\left|\overline{y}_1^{next} - y_1^{next}\right| < \varepsilon$. Moreover, P_1, P_2 and P_3 are defined as a composition of the EAC-computable functions so they are EAC-computable too. Similarly for the left side of the tape, we can define $\overline{y}_2^{next}$ such that $\left|\overline{y}_2^{next} - y_2^{next}\right| < \varepsilon$.

Finally, $\overline{f}_M : \mathbb{R}^3 \to \mathbb{R}^3$ is defined by $\overline{f}_M(\overline{y}_1, \overline{y}_2, \overline{q}) = \left(\overline{y}_1^{next}, \overline{y}_2^{next}, \overline{q}_{next}\right)$.

Let $0 \le \delta < \varepsilon$ and let $i \in \mathbb{N}$ satisfies $\sigma^{[i]}(\varepsilon) \le \varepsilon - \delta$. Define a map $f_M = \sigma^{[i]} \circ \overline{f}_M$. Then, if $x_0 \in \mathbb{N}^3$ is an initial configuration,

$$\left\|\overline{x}_0 - x_0\right\|_\infty \le \varepsilon \Rightarrow \left\|f_M(\overline{x}_0) - \theta(x_0)\right\|_\infty \le \varepsilon - \delta.$$

Thus, by the triangular inequality, if $\|\bar{x}_0 - x_0\|_\infty \le \varepsilon$, then for $\|f - f_M\|_\infty \le \delta$ we have:

$$\|f(\bar{x}_0) - \theta(x_0)\|_\infty \le \|f(\bar{x}_0) - f_M(\bar{x}_0)\|_\infty + \|f_M(\bar{x}_0) - \theta(x_0)\|_\infty \le \delta + (\varepsilon - \delta) = \varepsilon .$$

This proves the result for $j = 1$ and all the constructed elements are EAC-computable. For $j > 1$, we proceed by an induction.

□

The above theorem states that there exists the EAC-computable function which can simulate one step of the Turing machine.

And now we can write our main result.

Theorem 10. *For any Turing machine there exist some EAC which can robustly simulate it.*

Proof. This proof is based on the construction from [10]. Let $\theta: \mathbb{N}^3 \to \mathbb{N}$ be the transition function of the Turing machine M, under the encoding described above. To iterate the function θ we can use a pair of functions to control the evolution of two simulation variables z_1, z_2. Both simulation variables have values close to x_0 at $t = 0$.

The first variable is iterated during half of a unit period while the second remains approximately constant, its derivative is kept close to zero by a control function. Then, the first variable remains controlled during the following half unit period of time and the second variable is brought up close to it and at time $t = 1$ both variables have values close to $\theta(x_0)$.

This process can be repeated an arbitrary number of times, keeping the error under control because there exists an analytic function robust to error that simulates θ. We think of the following system of the differential equations:

$$z_1' = c_1 \left(z_1 - f_M \circ \sigma^{[m]}(z_2)\right)^3 \Phi_1(t, z_1, z_2),$$

$$z_2' = c_2 \left(z_2 - \sigma^{[n]}(z_1)\right)^3 \Phi_2(t, z_1, z_2),$$

with the initial conditions $z_1(0) = z_2(0) = \bar{x}_0$, where

$$\Phi_1(t, z_1, z_2) = l_2\left(s(t), \frac{c_1}{\gamma}\left(z_1 - f_M \circ \sigma^{[m]}(z_2)\right)^4 + \frac{c_1}{\gamma} + 10\right),$$

$$\Phi_2(t, z_1, z_2) = l_2\left(s(-t), \frac{c_2}{\gamma}\left(z_2 - \sigma^{[n]}(z_1)\right)^4 + \frac{c_2}{\gamma} + 10\right),$$

and c_1, c_2, γ, m and n are some constants; s is the EAC-computable control function of the form $s(t) = 1/2\left(\sin^2(2\pi t) + \sin(2\pi t)\right)$. The function f_M is the EAC-computable extension of θ from Theorem 9 where θ is the transition

function of the Turing machine M and σ, l_2 and l_3 are the functions from Lemmas from section 2. These functions keep the error under control. This construction guarantees that z_2' is sufficiently small on [0, 1/2] and, therefore,

$$\left\| z_2(1/2) - x_0 \right\|_\infty < 1/2,$$

and

$$\left\| z_2(1/2) - \theta(x_0) \right\|_\infty < \varepsilon .$$

For interval [1/2, 1] the roles of z_1 and z_2 are switched. This process can be repeated for z_1 and z_2 on subsequent intervals and we obtain for $j \in \mathbb{N}$, $t \in [j, j + 1/2]$ the following result: $\left\| z_2(t) - \theta^{[j]}(x_0) \right\|_\infty < \eta$, where $0 < \eta < 1/2$. Because all the above used functions are EAC-computable and because the EAC can generate the functions z_1 and z_2 as solutions of the differential equations[1] written above with the initial conditions $z_1(0) = z_2(0) = \overline{x}_0$, where $\left\| \overline{x}_0 - x_0 \right\|_\infty \le \varepsilon$ for $0 < \varepsilon < 1/4$.

Suppose that $z(t)$ is the function which has the following quality $\left\| z(t) - \theta^{[j]}(x_0) \right\|_\infty < \eta$ for $j \in \mathrm{N}$, if $t \in [j, j + 1/2]$. Since from the statement, that if the Turing machine reaches a halting state it moves to the same configuration, it is enough to take $\lim_{t \to 0} z(1/t)$ to obtain the EAC-computable function, which robustly simulates computations of the Turing machine with the transition function θ.

□

Conclusion

In this manner we obtain a positive answer to the question given by Rubel in [4] "whether the digital computer can be simulated by the EAC". There still remains the open question of an existence of some simulation of the EAC by digital computer and also the interesting question about closing the EAC with regard to the iteration.

Acknowledgemends

I would like to thank Jerzy Mycka for helpful discussion.

References

[1] Graça D.S., Campagnolo M.L., Buescu J., *Robust Simulations of Turing Machines with Analytic Maps and Flows*, In B. Cooper, B. Loewe, and L. Torenvliet, editors, Proceedings of

[1] This differential equations have an analytic solution z_1 and z_2.

CiE'05, New Computational Paradigms, Lecture Notes in Computer Science 3526. Springer-Verlag, (2005) 169.
[2] Siegelmann H.T., *Neural Networks and Analog Computation: beyond the Turing Limit*. Birkhäuser, (1999).
[3] Blum L., Shub M.,.Smale S., *On a theory of computational and complexity over the real numbers: NP-completeness, recursive functions and universal machines*. Bull. Amer. Math. Soc. (NS), 21 (1989) 1.
[4] Rubel L.A., *The Extended Analog Computer*. Advances in Applied Mathematics, 14 (1993) 39.
[5] Koiran P., Moore C., *Closed-form analytic maps in one and two dimentions can simulate universal Turing machine*. Theoretical Computer Science, 210 (1999) 217.
[6] Branicky M.S., *Universal computation and other capabilities of hybrid and continuous dynamical systems*. Theoret. Comput. Sci., 138 (1995) 67.
[7] Campagnolo M.L., Moore C., Costa J.F., *An analog characterization of the Grzegorczyk hierarchy*. J. Complexity, 18 (2002) 997.
[8] Moore C., *Finite-dimentional analog computers: Flows, maps, and recurrent neural natworks*. In C.Calude, J.Casti, M.Dinneen, eds.: 1st International Conference on Unconventional Models of Computation-UMC'98, Springer, (1998) 59.
[9] Campagnolo M.L., Moore C., Costa J.F., *Iteration, inequalities, and differentiability in analog computers*. J. Complexity, 16 (2000) 642.
[10] Graça D.S., Campagnolo M.L., Buescu J., *Robust Simulations of Turing Machines with Analytic Maps and Flows*. In B.Cooper, B.Loewe, and L.Torenvliet, editors, Proceedings of CiE'05, New Computational Paradigms, Lecture Notes in Computer Science 3526. Springer-Verlag, (2005) 169.
[11] Odifreddi P., *Classical recursion theory*. Elsevier, (1989).
[12] Rubel L.A., *Some Mathematical Limitations of the General-Purpose Analog Computer*. Advances in Applied Mathematics, 9 (1988) 22.
[13] Lipshitz L., Rubel L.A., *A differentialy algebraic replacement theorem, and analog computation*, Proceedings of the A.M.S., 99(2) (1987) 367.
[14] Campaniolo M.L., *Computational complexity of real valued recursive functions and analog circuits*, Universidade Técnica de Lisbona Instituto Superior Técnico, (2001).