Acta Cybernetica **19** (2010) 749–764.

The Extended Analog Computer and Functions Computable in a Digital Sense

Monika Piekarz*

Abstract

In this paper we compare the computational power of the Extended Analog Computer (EAC) with partial recursive functions. We first give a survey of some part of computational theory in discrete and in real space. In the last section we show that the EAC can generate any partial recursive function defined over $\mathbb{N}$. Moreover we conclude that the classical halting problem for partial recursive functions is an equivalent of testing by EAC if sets are empty or not.

Keywords: analog computation, Extended Analog Computer, recursion theory, recursive functions

1 Introduction

People have always sought some models which could help us to explain and model various aspects of Nature. Hence, the need for machines which could simulate some aspects of the physical world in order to understand and model it appeared in a natural way. Several computational models seemed to perform this task, but nowadays discrete models play the main role. Nevertheless, computers need not to be digital. In fact, the first computers were analog computers where the internal states are continuous rather than discrete which is the case in digital computers. Studies of these machines began long ago with the machines of V. Bush (see [2]) which suited well to solve ordinary differential equations and were effectively used to solve many military problems during World War II (see [10]) and work of C. Shannon (see [21]). Later, in the 60s and 70s, this subject lost its importance, but from the 80s onwards a renewed interest of the study on analog computation can be observed. This stems partly from the search for new models which could provide an adequate notion of computation and from the complexity of the dynamical systems that are currently used to model the physical world.

Notwithstanding, the present knowledge about analog computation still leaves many unsolved questions. Although the study of analog computation began long

*Institute of Mathematics, University of Maria Curie-Sklodowska, Lublin, Poland, E-mail: monika.piekarz@poczta.umcs.lublin.pl

ago many fundamental questions have been analyzed relatively recently and many of them still remain unsolved.

Two different families of models of analog computation, that come from different behavior of computation processes in time, appear in the theory of analog computation. The first family contains models of computation on real numbers but in discrete time. These are, for example, the machines of Blum-Shub-Smale (see [1]) and Analog Recurrent Neural Network (see [22]). The second one are the models of computation on real numbers and in continuous time. One of the important model seems to be General Purpose Analog Computer proposed by C. Shannon in 1941 (see [21]), and took over by M. B. Pour-El (see [15]), L. Lipshitz and L. A. Rubel (see [11]). This model is able to generate all differentially algebraic functions, and it is built from some analog units connected together. Recently, many authors discussed the GPAC (see [3], [4], [7], [9]). There exists also an extension of this model called Extended Analog Computer (EAC). This model was defined by L. A. Rubel (see [20]) in 1993 and was proposed as a model of the human brain.

This work focuses on the comparison of a computational power of the EAC and recursive functions over $\mathbb{N}$. This subject is related to the Rubel's question ([20]) whether the EAC can be simulated by a digital computer and whether the digital computer can be simulated by an EAC. Solution of similar problems in the context of GPAC can be found in [18] and [8].

2 Preliminaries

The fundamental notion of partial recursive functions plays a significant role in the classical theory of computability (main notation taken from [14]). These are the functions which map $\mathbb{N}$ into $\mathbb{N}$ and which can be thought of as an equivalent to the class of functions computable in an intuitive sense. In fact, in the theory of computability it is shown that the recursive functions are precisely the functions that can be computed by Turing machines. First let us establish some notation. Lower letters: k, n, m will denote natural numbers, x, y, z real numbers and $\bar{k}, \bar{n}, \bar{m}$ sequences of natural numbers ($\bar{x}, \bar{y}, \bar{z}$ respectively sequences of real numbers).

The class $\mathcal{PRF}$ of partial recursive functions can be defined as follows.

Definition 2.1. *The class of partial recursive functions ($\mathcal{PRF}$) defined over $\mathbb{N}$ is the smallest class of functions:*

- *contains $\mathcal{O}(n) = 0$, $\mathcal{S}(n) = n + 1$, $\mathcal{I}^i_k(n_1, \ldots, n_k) = n_i$ $(1 \leq i \leq k)$*
- *closed under composition, i. e. the schema that for given $g_1, \ldots, g_l : \mathbb{N}^k \to \mathbb{N}$, $f : \mathbb{N}^l \to \mathbb{N}$, $(k, l > 0)$ produces*

$$h(\bar{n}) = f(g_1(\bar{n}), \ldots, g_l(\bar{n})), \quad h : \mathbb{N}^k \to \mathbb{N},$$

where $\bar{n} = n_1, \ldots, n_k$, and the left side is undefined when at least one of the values of $g_1, \ldots, g_l, f$ for the given arguments is undefined,

– *closed under primitive recursion, i. e. the schema that for given* $f : \mathbb{N}^{n+2} \to \mathbb{N}, g : \mathbb{N}^n \to \mathbb{N}$ *produces* $h : \mathbb{N}^{n+1} \to \mathbb{N}, (n > 0)$

$$h(\bar{n}, 0) = g(\bar{n}),$$

$$h(\bar{n}, m + 1) = f(\bar{n}, m, h(\bar{n}, m)),$$

– *closed under unrestricted* μ*-recursion, i. e. the schema that for given* $f : \mathbb{N}^{n+1} \to \mathbb{N}$ *produces* $h : \mathbb{N}^n \to \mathbb{N}, (n > 0)$

$$h(\bar{n}) = \min_m((\forall j < m)(f(\bar{n}, j) \downarrow \wedge f(\bar{n}, j) \neq 0) \wedge f(\bar{n}, m) = 0),$$

i. e. $h(\bar{n})$ *is the smallest* m *such that* $f(\bar{n}, m) = 0$ *and* $h(\bar{n})$ *is undefined if there is no such* m*.*

The operation of unrestricted μ-recursion is undefined when $(\forall m)f(\bar{n}, m) \neq 0$ or $(\exists m)f(\bar{n}, m) = 0$ but for certain $j < m$, $f(\bar{n}, j)$ is undefined. If $h(\bar{n})$ is defined by unrestricted μ - recursion we will simply write $h(\bar{n}) = \mu_m f(\bar{n}, m)$. To receive only total functions, the above definition should be modified. Within the operation of μ-recursion it is required for the function f to be regular which means f is total and $(\forall \bar{n})(\exists m)f(\bar{n}, m) = 0$. Such definition gives the class of total recursive functions. For simplicity, we will write "recursive function" instead of "total recursive function".

In the rest of the paper the notions of recursive sets and recursively enumerable sets will be used. It is said that a set S of natural numbers is recursive if its characteristic function is recursive. The characteristic function is understood as the function defined by:

$$c_S(\bar{n}) = \begin{cases} 0 & \bar{n} \in S \\ 1 & \bar{n} \in \neg S \end{cases}$$

where $\neg S$ denotes the complement of S.

It is said that a set S of natural numbers is recursively enumerable if S is the range of a partial recursive function.

Let us recall that the graph G_f of f is a set (a relation) defined in the following way: $G_f(\bar{n}, m) \Leftrightarrow f(\bar{n}) = m$.

Let us take a few useful results from [14].

Proposition 2.1. *Let* f *and* g *be, respectively, a partial and a total function. Then:*

– f *is partial recursive if and only if its graph is recursively enumerable,*

– g *is recursive if and only if its graph is recursive.*

Proposition 2.2. *A set* S *is recursive iff it is a recursively enumerable set and its complement is a recursively enumerable set too.*

Let $\mathcal{P}[\bar{n}, \bar{m}, \mathbb{Z}]$ be the ring of polynomials in the (infinite denumerable) set of unknowns with coefficients in the set $\mathbb{Z}$ of integers[1].

Let us present here some facts about recursive enumerable sets (taken from [14]).

Theorem 2.1. *Let $p(\bar{n}, \bar{m}) \in \mathcal{P}$ be a polynomial with the $(k+l)$ unknowns, $k, l \in \mathbb{N}$, where $n \in \mathbb{N}^k, m \in \mathbb{N}^l$. The set D (called a Diophantine set) defined in the following way* [2]

$$\langle \bar{n} \rangle \in D \Leftrightarrow (\exists(\bar{m}))p(\bar{n}, \bar{m}) = 0$$

is recursively enumerable and conversely every recursively enumerable set S is a Diophantine set.

More information about the above result can be found in [12] or [6]. It is worth mentioning here that Theorem 2.1 negatively solves Hilbert's Tenth Problem to decide effectively whether a given Diophantine equation: $p(\bar{n}, \bar{m}) = 0$ has a solution or not.

3 The General Purpose Analog Computer (GPAC)

In the two following sections some part of the theory of analog computation will be recalled. We start with the basic continuous-time model of analog computation known as General Purpose Analog Computer (GPAC). The GPAC was introduced in 1941 by C. Shannon (see [21]) as a mathematical model of an analog device called the Differential Analyzer (see [2]). Many variants of the Differential Analyzer was used from the 1930s to the early 60s to solve numerical problems. The Differential Analyzer may be seen as a circuit build of interconnected analog units. The units used by GPAC can be listed as follows:

- *Integrator*: a unit with a setting for initial condition: two constants a and t_0; two inputs: unary functions f, g; one output: the Riemann-Stieltjes integral $\int_{t_0}^{t} f(x)dg(x) + a$.
- *Constant multiplier*: a unit associated with a real number c with one input: function f; one output: cf.
- *Adder*: a unit with two inputs: functions f, g; one output: $f + g$.
- *Multiplier*: a unit with two inputs: functions f, g; one output: fg.
- *Constant function*: a unit with no input; one output: always equals 1.

[1] Sometimes for brevity we will write $\mathcal{P}$.

[2] Equation of the form $p(\bar{n}, \bar{m}) = 0$ is called Diophantine equation.

The GPAC model proposed by Shannon in [21] has further been refined in [11], [15], [7], [9]. Shannon, in his original paper, claimed that a unary function can be generated by GPAC if and only if the function is differentially algebraic, i. e. if it satisfies the condition the following definition:

Definition 3.1. *A unary function $f(x)$ is differentially algebraic ($\mathcal{DA}$) on the interval I if there exist some natural number n (where $n > 0$) and some $(n+2)$-ary nonzero polynomial P with real coefficients such that*

$$P(x, f(x), f'(x), \ldots, f^{(n)}(x)) = 0,$$

for every $x \in I$.

Shannon's definition of the GPAC was refined by M. B. Pour-El (see [15]) and L. Lipshitz, L. A. Rubel (see [11]). It is worth to notice that there are some functions such as the Euler Γ-function, $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$, that cannot be generated by the GPAC because they are not differentially algebraic functions (compare [21]). However, it is a classical result in computable analysis that Γ is computable. So, it might seem that the GPAC is a less powerful model than computable analysis when "real time" computation is used for the GPAC. But in [9] D. S. Graça and J. F. Costa introduce some useful notion of the GPAC called a FF-GPAC. They refine the GPAC in terms of circuits to avoid problematic cases, showing that their model is equivalent to solutions of polynomial ODE's.

Definition 3.2. *Consider a GPAC U with n integrators $\overline{U}_1, \ldots, \overline{U}_n$. Suppose that to each integrator $\overline{U}_i$, $i = 1, \ldots, n$, we can associate two linear circuits [3], A_i and B_i, with the property that the integrand and the variable of integration inputs of $\overline{U}_i$ are connected to the outputs of A_i and B_i respectively. Suppose also that each input of the linear circuits A_i and B_i is connected to one of the following: the output of an integrator or to an input unit. U is said to be a feedforward GPAC (FF-GPAC) iff there exists an enumeration of the integrators of U, $U_1, \ldots, U_n$, such that the variable of integration of k-th integrator can be expressed as*

$$c_k + \sum_{i=1}^{m} c_{ki}x_i + \sum_{i=1}^{k-1} \overline{c}_{ki}y_i, \quad for\ all\ k = 1, \ldots, n,$$

where y_i is the output of U_i, for $i = 1, \ldots, n$, x_j is the input associated to the j-th input unit, and $c_k, c_{kj}, \overline{c}_{ki}$ are suitable constants, for all $k = 1, \ldots, n$, $j = 1, \ldots, m$ and $i = 1, \ldots, k-1$, where m is a number of input units used in U.

This definition describes the GPAC with some restrictions of the feedforward. So we get a model which is "well-behaved" when compared with the GPAC proposed by Pour-El.

[3] A limear circuit is an acyclic GPAC build only with adders, constant multipliers, and constant function units. If $x_1, \ldots, x_n$ are the inputs of a linear circuit, then the output of the circuit will be $y = c_0 + c_1x_1 + \cdots + c_nx_n$, where $c_0, c_1, \ldots, c_n$ are appropriate constants.

Other refinements of the notion of the GPAC and the GPAC-computability were made by D. S. Graça in [7]. He showed that if we do not compute in "real time" (which is usual way of computing for the GPAC) but in "limit time", then both the Euler Γ-function and Riemann ζ-function become computable. This result was generalized in [3] where it is shown that, on compact intervals, any function computable in the computable analysis sense can be computed by a GPAC using "limit time". In [7] Graça stated that, if f is generated by such GPAC, then f is computable by Rubel's Extended Analog Computer [20].

4 The Extended Analog Computer (EAC)

The topic of the section will be the mathematical model of analog computation proposed by L. A. Rubel in 1993 and called the Extended Analog Computer (EAC) (see [20]), which does not correspond to any existing device. Now, as a result of over a decade's of research we have some kind of implementation of Rubel's Extended Analog Computer model (see [13]) given by J. Mills. But this implementation is neither identical to the EAC model, nor is a complete implementation of the EAC model.

Rubel's EAC was introduced to expand the scope of the General Purpose Analog Computer (GPAC) (see [17]). The EAC works on a hierarchy of levels. It has no inputs from the outside, but it has a finite number of "settings", which are arbitrary real numbers (we don't put any restrictions of these real numbers, they have to be neither rational nor digitally computable i. e. approximable by Turing machine). At each level we have black boxes which produce real constants and independent variables $x_1, x_2, \ldots$. The outputs of the machine at level $(n-1)$ can be used as inputs at level n or any higher level. The inputs and outputs of levels are functions of a finite number of independent variables which are defined on some sets. Every function f produced by the EAC is associated with some set Λ on which it is defined, thus we have an ordered pair (f, Λ). At the lowest level 0, the EAC produces real polynomials of a finite number of real variables. However, at level 1 it produces all differentially algebraic functions of a finite number of real variables.

In general the EAC can generate a function on level n which is built of functions generated on level $(n-1)$ by the following operations: addition, multiplication, composition, inversion, differences, analytic continuation, solving a differential equation with boundary value conditions and limit taking.

The EAC can produce on a half of levels certain sets Ω in Euclidean space too. The sets can be obtained as non-negative or positive part of domain of some function f produced on the previous level. Moreover on the same half of levels unions, intersections or projections of such sets can be produced.

Before we give the definition of the EAC, let us recall some useful notation concerning analytic function. We say that f is in a class $C^\omega(\Lambda)$ and write $f \in C^\omega(\Lambda)$ if there is an extension $\tilde{f}$ of f to an open supersubset $\tilde{\Lambda}$ of Λ, and $\tilde{f}$ is real-analytic on $\tilde{\Lambda}$. By real-analytic functions we mean that these functions are locally sums of convergent power series.

Definition 4.1. [4] *The* EAC *machine can produce: function* $f \in C^\omega(\Lambda)$ *defined on* Λ *on the level* n, $(f,\Lambda) \in \mathrm{EAC}_n$ *where* $n \in \mathbb{N}_0$ *and* $\Lambda \in \mathrm{EAC}_{n-\frac{1}{2}}$; *or set* $\Omega \subseteq \mathbb{R}^k$ *on the level* $n+\frac{1}{2}, \Omega \in \mathrm{EAC}_{n+\frac{1}{2}}$ *where* $n \in \mathbb{N}_0$ *if the following conditions hold:*

1. *For* $n = 0, (f, \mathbb{R}^k) \in \mathrm{EAC}_0$

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

where finite $A \subset \mathbb{N}_0^k$ *and* $c_{\alpha_1\alpha_2\ldots\alpha_k}$ *are fixed real constants.*

2. *For finite* $n > 0$, $(f,\Lambda) \in EAC_n$ *is defined by one of the following methods:*

   - $f(\bar{x}) = g_1(\bar{x}) + g_2(\bar{x})$, *where* $(g_1,\Lambda),(g_2,\Lambda) \in \mathrm{EAC}_{n-1}$;
   - $f(\bar{x}) = g_1(\bar{x})g_2(\bar{x})$, *where* $(g_1,\Lambda),(g_2,\Lambda) \in \mathrm{EAC}_{n-1}$;
   - $f(\bar{x}) = h(g_1(\bar{x}),\ldots,g_l(\bar{x}))$, *where* $(h,\Omega),(g_1,\Lambda),\ldots,(g_l,\Lambda) \in \mathrm{EAC}_{n-1}$;
   - $f(\bar{x}) = f_i(\bar{x})$ *for some* $i = 1,2,\ldots,l$, *where* $f_1(\bar{x}), f_2(\bar{x}), \ldots, f_l(\bar{x})$ [5] *are the* $C^\omega(\Lambda)$*-functions which are the solutions of*

$$\begin{cases} g_1(\bar{x}, f_1, f_2, \ldots, f_l) = 0 \\ g_2(\bar{x}, f_1, f_2, \ldots, f_l) = 0 \\ \ldots \\ g_l(\bar{x}, f_1, f_2, \ldots, f_l) = 0 \end{cases}$$

     where $(g_1,\Gamma),(g_2,\Gamma),\ldots,(g_l,\Gamma) \in \mathrm{EAC}_{n-1}$, $\Gamma \in EAC_{n-\frac{3}{2}}$;
   - $f(\bar{x}) = Dg(\bar{x})$, *where* $Dg = \frac{\partial^{\alpha_1+\alpha_2+\cdots+\alpha_k}}{\partial x_1^{\alpha_1}\partial x_2^{\alpha_2}\ldots\partial x_k^{\alpha_k}} g(\bar{x})$ *are the partial derivatives and* $(g,\Lambda) \in \mathrm{EAC}_{n-1}$;
   - $f = \tilde{f}_{|\Lambda}$ [6] *where* $\Lambda \subset \tilde{\Lambda}$ *and* $(\tilde{f},\tilde{\Lambda}) \in \mathrm{EAC}_n$;
   - $f(\bar{x}) = h(\bar{x})$ *if* $(h,\Lambda) \in \mathrm{EAC}_n$ *is an analytic continuation* [7] *of* $\tilde{h}$ *from* $\Lambda \cap \tilde{\Lambda}$ *to all of* Λ, *where* $(\tilde{h},\tilde{\Lambda}) \in \mathrm{EAC}_n$ *is defined on* $\tilde{\Lambda}$ *and* $\Lambda \cap \tilde{\Lambda} \neq \emptyset$;

[4] This definition is the formalized version of the definition of the EAC proposed by L. A. Rubel in [20]

[5] It is required for these functions to be well-defined C^ω-functions on Λ. For example the equation $xy - 1 = 0$ has the solution $y = \frac{1}{x}$ which is not well-defined on $\mathbb{R}$ (because it is not defined for $x = 0$) but it is well-defined on the intervals $(-\infty, 0)$ and $(0, \infty)$. So $y = \frac{1}{x}$ is not EAC computable on $\mathbb{R}$ but is EAC computable on $(-\infty, 0)$ or on $(0, \infty)$.

[6] $f : \Lambda \to \mathbb{R}$ and $f(\bar{x}) = \tilde{f}(\bar{x})$ for all $\bar{x} \in \Lambda$.

[7] We understand the analytic continuation as in [23].

– *$f(\bar{x})$ is a solution of equations*

$$F_i(\bar{x} : f, f^{(\alpha_1)}, f^{(\alpha_2)}, \ldots, f^{(\alpha_l)}) = 0,$$

for $i = 1, \ldots, k$ on a set Λ which are subject to certain boundary values requirement [8] *where $F_i \in \mathrm{EAC}_{n-1}$ and $f^{(\alpha_1)}, f^{(\alpha_2)}, \ldots, f^{(\alpha_l)}$ denote some partial derivatives of f;*

– *for all $\bar{x}_0 \in \Lambda$,*

$$f(\bar{x}_0) = \lim_{\substack{\bar{x} \to \bar{x}_0 \\ \bar{x} \in \Gamma}} g(\bar{x})$$

and

$$\frac{\partial^{\alpha_1+\alpha_2+\cdots+\alpha_k}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_k^{\alpha_k}} f(\bar{x}_0) = \lim_{\substack{\bar{x} \to \bar{x}_0 \\ \bar{x} \in \Gamma}} \frac{\partial^{\alpha_1+\alpha_2+\cdots+\alpha_k}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_k^{\alpha_k}} g(\bar{x})$$

where $(g, \Gamma) \in \mathrm{EAC}_{n-1}$ and Λ is a subset of $\partial\Gamma$ (is the edge of Γ);

3. For $n + \frac{1}{2}, n \in \mathbb{N}_0, \Omega \in \mathrm{EAC}_{n+\frac{1}{2}}$

– *Ω is the set $\{\bar{x} \in \Lambda : f(\bar{x}) > 0\}$ or the set $\{\bar{x} \in \Lambda : f(\bar{x}) \geq 0\}$ where $(f, \Lambda) \in \mathrm{EAC}_n$;*

– *$\Omega = \Omega_1 \cap \Omega_2$ or $\Omega = \Omega_1 \cup \Omega_2$ where $\Omega_1, \Omega_2 \in \mathrm{EAC}_{n+\frac{1}{2}}$;*

– *$\Omega = \{\bar{x} : (\exists x \in \mathbb{R})(x, \bar{x}) \in \Omega_1\}$ where $\Omega_1 \in \mathrm{EAC}_{n+\frac{1}{2}}$.*

Figure 1 presents an example of how we can get an EAC on the level $n + 1$ where functions $g_1, \ldots, g_l$ are not necessarily different from $f_1, \ldots, f_m$. In this Figure we obtain functions $f_1, \ldots, f_m$ on the level n, then from these functions we can compute a set on EAC level $n + 1/2$ by some operations O_s. Farther these functions and the set can be used as inputs for the level $n+1$ and by the operation O_f we obtain some function as output of EAC level $n + 1$. Where O_s denotes operation on sets described in point 3 of Definition 4.1 and O_f denotes operation on functions described in point 2 of Definition 4.1.

Moreover, there is an additional requirement for the EAC. The machine is required to produce unique outputs that are close on a compact set to the original

[8] For example: $f = f_0$ on a piece γ_0 of the boundary of Λ, and only functions $f_0 \in \mathrm{EAC}_{n-1}$ is used.

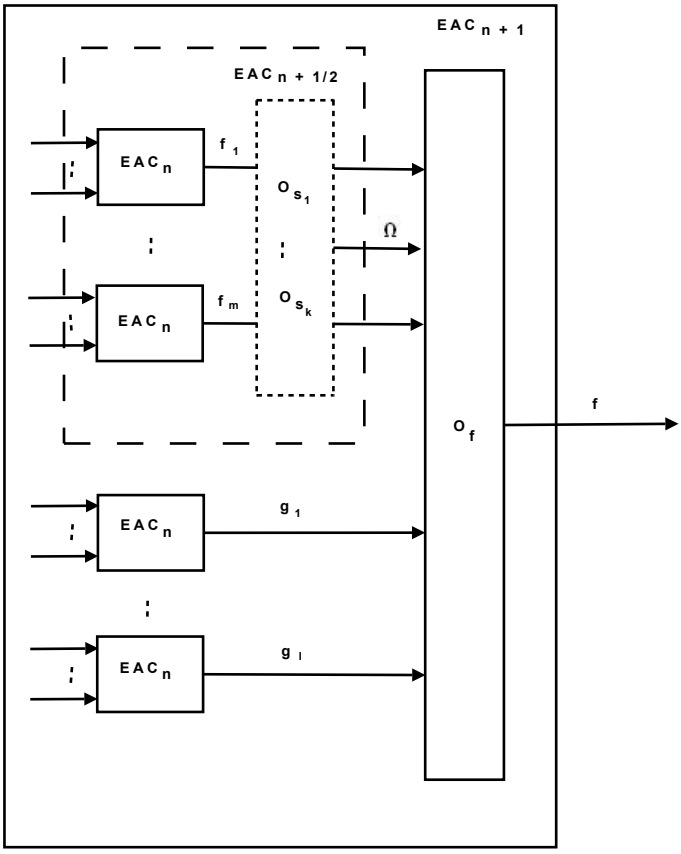

Figure 1: Extended Analog Computer — model of levels

unique output, in the case when the inputs are slightly deviated from the initial setting.

The Extend Analog Computer defined above has some interesting qualities. Some of them will be presented now. The results are taken from [20]. These machines can compute all the functions which can be computed by the GPAC. On level 1 the EAC can compute all differentially algebraic functions from C^ω. On the higher levels of the EAC such functions as the Euler Γ-function or the Riemman ζ-function can be obtained. We also know that a solution to Dirichlet problem for the Laplace's equation in the unit disc can be computed by the EAC.

Example 4.1 (Euler Γ-function)**.** Let us consider the Euler Γ-function defined by:

$$\Gamma(x) = \int_0^\infty t^x e^{-t} \frac{dt}{t}.$$

To show how the EAC can generate this function we introduce the following function:

$$\Gamma^*(x) = \int_1^\infty t^x e^{-t} \frac{dt}{t}$$

.

The fact the EAC can compute $\Gamma^*(x)$ will be presented below and the other part $\int_0^1 t^x e^{-t} \frac{dt}{t}$ of the integral $\Gamma(x)$ can be handled similarly. Let

$$f(x,y) = \int_{t=1}^{t=y} t^x e^{-t} \frac{dt}{t},$$

$$g(x,y) = f(x, \frac{1}{y}),$$

$$h(x) = \lim_{y \to 0} g(x,y),$$

where limits of partial derivatives of g are well-behaved. Now we have to show that $h(x)$ is EAC computable. Since $t^{x-1}e^{-t}$ is differentially algebraic so it can be generated by the EAC. Since $g(x,y)$ is a solution of the following differential equation with boundary values:

$$g(x,1) = 0 \quad \text{for all} \quad x > 0, \quad \partial_y g(x,y) = y^{x-1}e^{-y}(-\tfrac{1}{y^2}),$$

it is an EAC computable function. Now we can put the limit to compute $h(x)$ by an EAC.

In a similar manner one can show the Riemman ζ-function can be computed by an EAC.

5 Richardson's results and the EAC

Let us start with selected Richardson's results from [16] and some results due to N. C. A. da Costa, F. A. Doria from [5]. We start with the definition and some lemmas from [16]. Let $\mathcal{A}[x_1, x_2, \ldots, \mathbb{R}]$ be an algebra of supplementary functions in the variables $x_1, x_2, \ldots$ over the real numbers $\mathbb{R}$ which corresponds to the set of expressions representing those functions. We construct algebra $\mathcal{A}[x_1, x_2, \ldots, \mathbb{R}]$ as follows:

1. All real numbers belong to $\mathcal{A}$,
2. $x_i, \sin(x_i), \exp(x_i) \in \mathcal{A}$,
3. If $f, g \in \mathcal{A}$ then $f + g \in \mathcal{A}$ and $fg \in \mathcal{A}$,
4. If $f, g \in \mathcal{A}$ then $(f \circ g) \in \mathcal{A}$, the symbol $\circ$ denotes the composition of functions,
5. $\mathcal{A}$ is the smallest algebra closed under the above conditions.

Let us present some useful results, see [16].

Lemma 5.1. *$\mathcal{A}$ is closed under partial derivatives.*

Lemma 5.2. *If $f \in \mathcal{A}$, then there is $g \in \mathcal{A}$ so that:*

$$(\forall \bar{x} \in \mathbb{R}^n) g(\bar{x}) > 1 \text{ and}$$

$$(\forall \bar{x} \in \mathbb{R}^n, \bar{\Delta} \in \mathbb{R}^n)(|\Delta_i| \leq 1) \rightarrow (g(\bar{x}) > |f(\bar{x} + \bar{\Delta})|).$$

As the conclusion from Lemma 5.1 and Lemma 5.2 we can obtain the following lemma.

Lemma 5.3. *There is a constructive procedure such that for a given expression for $p \in \mathcal{P}$ one can obtain the expressions for functions $k_i \in \mathcal{A}$ satisfying the following condition: if $|\Delta_i| \leq 1, i = 1, \ldots, n$, then*

$$k_i(\bar{m}, \bar{x}) > |\partial_i(p^2(\bar{m}, \bar{x} + \bar{\Delta}))|, \tag{1}$$

where $\bar{m} \in \mathbb{N}^k$ and $\bar{x} \in \mathbb{R}^n, \bar{\Delta} \in \mathbb{R}^n$, $k, n \in \mathbb{N}$.

Now we can add some observation regarding the algebra $\mathcal{A}$ in the context of the Extended Analog Computer. By using only the definition of the algebra $\mathcal{A}$, we can give our lemma which connects the EAC with $\mathcal{A}$.

Lemma 5.4. *The* EAC *can compute all functions from $\mathcal{A}$.*

Proof. Elementary functions like $\exp(x)$ and $\sin(x)$ are differentially algebraic and therefore are EAC computable. By the definition, the EAC is closed under addition, multiplication and composition. So the EAC can generate all functions from $\mathcal{A}$. □

Directly from Lemma 5.3 and Lemma 5.4 we obtain the following fact.

Remark 5.1. All functions k_i from Lemma 5.3 are EAC computable functions.

Now let us recall another useful notion from [16].

Definition 5.1. *For given polynomial $p \in \mathcal{P}$, and k_i as in Lemma 5.3 let us define:*

$$f(\bar{m}, \bar{x}) = (n+1)^4[p^2(\bar{m}, \bar{x}) + \sum_{i=1}^{n} (\sin^2 \pi x_i) k_i^4(\bar{m}, \bar{x})],$$

where $\bar{m} \in \mathbb{N}^k$ and $\bar{x} \in \mathbb{R}^n$, $k, n \in \mathbb{N}$.

It can be easily observed that $f(\bar{m}, \bar{x})$, as the composition of EAC computable functions, is the EAC computable, too. The following result is proved in [16].

Theorem 5.1. *For p and f defined as above, the following conditions are equivalent: for every $\bar{m} \in \mathbb{N}^k$:*

1. *There are natural numbers $x_1, \ldots, x_n$ such that $p(\bar{m}, x_1, \ldots, x_n) = 0$.*

2. *There are nonnegative real numbers* $x_1, \ldots, x_n$ *such that* $f(\bar{m}, x_1, \ldots, x_n) = 0$.

3. *There are nonnegative real numbers* $x_1, \ldots, x_n$ *such that* $f(\bar{m}, x_1, \ldots, x_n) \leq 1$.

The additional function $\hat{f}(\bar{m}, x_1, \ldots, x_n) = f(\bar{m}, x_1^2, \ldots, x_n^2)$ will be introduced and used. It is easy to obserwe $\hat{f}$ is EAC computable too.

For the purpose of creating one-argument functions the following construction is used in [16]. Let $r(y) = y\sin(y)$, and $s(y) = y\sin(y^3)$. Then for given $\hat{f}(\bar{m}, x_1, \ldots, x_n)$, the following substitutions are made:

$$\begin{aligned} x_1 &= r(y), \\ x_2 &= (r \circ s)(y), \\ x_3 &= (r \circ s \circ s)(y), \\ &\vdots \\ x_{n-1} &= (r \circ \underbrace{s \circ \cdots \circ s}_{n-2})(y), \\ x_n &= (\underbrace{s \circ s \circ \cdots \circ s}_{n})(y). \end{aligned} \tag{2}$$

Finally, we obtain

$$g(\bar{m}, y) = \hat{f}(\bar{m}, r(y), r(s(y)), \ldots, r(s(s(\ldots s(y))\ldots)), s(s(s(\ldots s(y))\ldots))),$$

where g is defined on $\mathbb{R}$ and with values in $\mathbb{R}$, $\bar{m} \in \mathbb{N}^k$.

The above functions s and r are the EAC computable, so this construction can be done by the EAC, as the composition of EAC computable functions.

Now as a consequence of Theorem 5.1 we can prove the most important corollary for the main results.

Corollary 5.1. *For every* $\bar{m} \in \mathbb{N}^k$ *the following conditions are equivalent:*

1. *There are natural numbers* $x_1, \ldots, x_n$ *such that* $p(\bar{m}, x_1, \ldots, x_n) = 0$.

2. *There is a real number* y *such that* $g(\bar{m}, y) \leq 1$.

Moreover, p *and* g *are* EAC *computable.*

Proof. From the introduction to Theorem Two presented in [16] it is known that for any real numbers $y_1, \ldots, y_n$, and any $\delta > 0$, there is a real number y such that:

$$\begin{aligned} &|r(y) - y_1| < \delta \\ &|r(s(y)) - y_2| < \delta \\ &\vdots \\ &|r(\underbrace{s(s(\cdots s(y))\cdots)}_{n-2}) - y_{n-1}| < \delta \\ &\underbrace{s(s(s(\cdots s(y))\cdots))}_{n} = y_n. \end{aligned} \tag{3}$$

We first consider the second condition of our corollary. So, by the equality

$$g(\bar{m}, y) = \hat{f}(\bar{m}, r(y), r(s(y))), \ldots, r(s(s(\ldots s(y))\ldots), s(s(s(\ldots s(y))\ldots)),$$

and by the cited result (3) this condition holds iff there exist real numbers $y_1, \ldots, y_n$ for which $\hat{f}(\bar{m}, y_1, \ldots, y_n) \leq 1$.
Now $\hat{f}(\bar{m}, y_1, \ldots, y_n)$ was defined as $f(\bar{m}, y_1^2, \ldots, y_n^2)$, so there is the equivalent condition: there exist nonnegative real numbers $x_1, \ldots, x_n$ for which $f(\bar{m}, x_1, \ldots, x_n) \leq 1$. Next, it follows from Theorem 5.1 that the fact that there exist nonnegative real numbers $x_1, \ldots, x_n$ for which $f(\bar{m}, x_1, \ldots, x_n) \leq 1$ is equivalent to the statement that there exist natural numbers $x_1, \ldots, x_n$ for which $p(\bar{m}, x_1, \ldots, x_n) = 0$. So, we finally get:

$$(\exists y \in \mathbb{R}) g(\bar{m}, y) \leq 1 \equiv (\exists (x_1, \ldots, x_n) \in \mathbb{N}^n) p(\bar{m}, x_1, \ldots, x_n) = 0.$$

It is easy to observe that p ang g are EAC computable. Indeed, p is the polynomial and from the construction of g we see that g is EAC computable too. □

6 Main results

In this section we present our main results of this paper. We prove that the EAC can generate real functions which extend all partial recursive functions defined on $\mathbb{N}$.

The following theorem can be proved using facts quoted in the preliminaries and the previous two sections.

Theorem 6.1. *Every recursively enumerable set S can be generated by the* EAC.

Proof. Let D be recursively enumerable set. So D is also a Diophantine set. Then there exists a polynomial p for which the following condition holds:

$$x \in D \equiv (\exists z \in \mathbb{N}) p(x, z) = 0.$$

By Corollary 5.1, there exists an EAC computable function $g(x, y)$ such that

$$x \in S \equiv (\exists y \in \mathbb{R}) g(x, y) \leq 1,$$

where the set $S \subseteq \mathbb{R}$ has the following property of being identically with D on $\mathbb{N}$

$$(\forall n \in \mathbb{N})(n \in S \Leftrightarrow n \in D).$$

Let us present details of construction S by EAC. If

$$h(x, y) = 1 - g(x, y)$$

then the EAC can generate by Definition 4.1 the following set

$$S' = \{(x, y) \in \mathbb{R} : h(x, y) \geq 0\}$$

and also on the same level the set

$$S = \{x : (\exists y \in \mathbb{R})(x, y) \in S'\}.$$

Now we construct the set $\mathbb{N}$ of natural numbers as an intersection of N_1 and N_2, where

$$N_1 = \{x \in \mathbb{R}_+ : \sin(2x\pi) \geq 0\},$$

$$N_2 = \{x \in \mathbb{R}_+ : -\sin(2x\pi) \geq 0\}$$

and $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$. The function $\sin(2x\pi)$ is the EAC computable. So by Definition 4.1, the set $\mathbb{N}$ can be obtained by the EAC. Finally, the set D is simply the intersection $S \cap \mathbb{N}$. □

Because every recursive set is recursively enumerable we get the following corollary.

Corollary 6.1. *Every recursive set S can be generated by the* EAC.

Theorem 6.2. *Let f be a partial recursive function defined over $\mathbb{N}$. Then there exists a family $(M_n)_{n\in\mathbb{N}}$ of* EAC*s such that for each $n \in \mathbb{N}$ M_n generates a singleton $\{f(n)\}$ if $f(n)\downarrow$ or $\emptyset$ if $f(n)\uparrow$.*

Proof. For every partial recursive function the set of pairs $\{(n, f(n)) : n \in \mathbb{N}, f(n)\downarrow\}$ is recursively enumerable. It follows from the proof of Theorem 6.1 that the EAC can construct set G_f of pairs of real numbers such that for all pairs of natural numbers if $(a, b) \in G_f$, then $b = f(a)$ and moreover $\{(n, f(n)) : n \in \mathbb{N}, f(n)\downarrow\} \subseteq G_f$.

Let n be a given natural number. We construct M_n which generates a singleton $\{f(n)\}$ in the following manner. First we build the functions $(x - n)$ and $(n - x)$ and next the sets:

$$\Omega_n^1 = \{(x, y) \in \mathbb{R}^2 : x - n \geq 0\}$$

and

$$\Omega_n^2 = \{(x, y) \in \mathbb{R}^2 : n - x \geq 0\}.$$

Then on the same level we obtain the set

$$\Omega_n = \{(x, y) \in \mathbb{R}^2 : x - n = 0\}$$

as an intersection $\Omega_n^1 \cap \Omega_n^2$. Finally, as the intersection of sets G_f and Ω_n we obtain the singleton $\{(n, f(n))\}$ if $f(n)\downarrow$ or empty set if $f(n)\uparrow$ and on the same level by projection of set $G_f \cap \Omega_n$ we can obtain in M_n set $\{f(n)\}$ if $f(n)\downarrow$ or empty set if $f(n)\uparrow$. □

The above theorem shows us that the halting problem for partial recursive functions (i. e. "is the partial recursive function defined for given n or is not defined") can be reduced in the EAC version to the question: "is the set empty?". However, we must remember that we discuss not the unique EAC but the whole

family of EAC machines in Theorem 6.2, so the exact connection between halting problem and the emptiness of sets is a little more sophisticated.

Moreover we know now that for any partial recursive function f defined over $\mathbb{N}$ and for each $n \in \mathbb{N}$ we can obtain by EAC a singleton $\Lambda = \{f(n)\}$ if $f(n) \downarrow$ or $\Lambda = \emptyset$ if $f(n) \uparrow$. Let i be the identity function defined over $\mathbb{R}$. So we can compute on EAC function $i_{|\Lambda}$ and obtain value $f(n)$ if $f(n) \downarrow$.

7 Conclusions

The above result implies that for any recursively enumerable sets there exists the EAC machine which generates it, and therefore the value of every partial recursive function at a given point can be obtained by the EAC (i. e. using the identity function on the singleton domain). It still remains unsolved whether for a partial recursive function f defined on $\mathbb{N}$ the EAC can generate a function $\tilde{f}$ defined on $\mathbb{R}$ such that $(\forall n \in \mathbb{N})\tilde{f}(n) \approx f(n)$. Moreover we have seen that classical halting problem is an equivalent to answer by EAC on the question: "is the set empty?".

References

[1] 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), Volume 21,1-49, 1989.

[2] Bush, V.: *The Differential Analyzer. A New Machine for Solving Differential Equations*, J. Franklin Institute, Volume 212, 447-488, 1931.

[3] Bournez, O., Campagnolo, M. L., Graça, D. S., Hainry, E.: *Polynomial Differential Equations Compute All Real Computable Functions on Computable Compact Intervals*. Journal of Complexity, 23(3), 317-335, 2007.

[4] Campagnolo, M., Moore, C., Costa, J. F., *Iteration, Inequalities, and Differentiabiity in Analog Computers*. Journal of complexity, 16(4), 642-660, 2000.

[5] da Costa, N. C. A., Doria, F. A.: *Undecidability and Incompleteness in Classical Mechanics*, International J. Theoret. Physics, Volume 30, 1041-1073, 1991.

[6] Davis, M., Matijasevich, Y. Robinson, J.: *Hilbert's Tenth Problem. Diophantine Equations: Positive Aspects of a Negative Solution*, Proc. Symp. Pure Math., 28, 323-378, 1976.

[7] Graça, D. S.:*Some Recent Developments on Shannon's General Purpose Analog Computer*, Math. Log Quart., Volume 50(4-5), 473-485, 2004.

[8] Graça, D. S., Campagnolo, M., Buescu, J.: *Computability with Polynomial Differential Equations*. Applied Mathematics, 40(3), 330-349, 2008.

[9] Graça, D. S., Costa, J. F.:*Analog Computers and Recursive Functions over the Reals*, J. Complexity, Volume 19(5), 644-664, 2003.

[10] Holst, P. A.: *Svein Rosseland and the Oslo Analyser*, IEEE Annals of History of Computing, Volume 18(4), 16-26, 1996.

[11] Lipshitz, L., Rubel, L. A.: *A Differentialy Algebraic Replacement Theorem, and Analog Computation*. Proceedings of the A.M.S., Volume 99(2), 367-372, 1987.

[12] Matijasevich, Y.: *Enumerable Sets are Diophantine*. Dokl. Acad. Nauk, 191, 279-282, 1970.

[13] Mills, J. W.:*The Nature of the Extended Analog Computer*. Physica, Nonlinear Phenomena, Volume 237, No 9, 1236-1256, 2008.

[14] Odifreddi, P.: *Classical Recursion Theory*, Elsevier, 1989.

[15] Pour-El, M. B.: *Abstract Computability and Its Relation to the General Purpose Analog Computer*, Trans. Am. Math. Soc.,199, 1-28, 1974.

[16] Richardson, D.: *Some Undecidable Problems Involving Elementary Functions of a Real Variable*, The Journal of Symbolic Logic, Volume 33, Number 4, 1968, 514-520.

[17] Rubel, L. A.: *Some Mathematical Limitations of the General-Purpose Analog Computer*, Advances in Applied Mathematics, Volume 9,22-34, 1988.

[18] Rubel, L., A.: *Digital Simulation of Analog Computation, and Church's Thesis*. J. Symbolic Logic, 54, 1011-1017, 1989.

[19] Rubel, L. A.:*A Survey of Transcendentally Transcendental Functions*, Amer. Math. Monthly, Volume 96(9), 777-788, 1989.

[20] Rubel, L. A.: *The Extended Analog Computer*, Advances in Applied Mathematics, Volume 14, 39-50, 1993.

[21] Shannon, C.: *Mathematical Theory of the Differential Anlyzer*, J. Math. Phys. MIT, 20, 337-354, 1941.

[22] Siegelmann, H. T.: *Neural Networks and Analog Computation: Beyond the Turing Limit*, Birkhäuser, 1999.

[23] Whittaker, E. T., Watson, G. N. *"The Process of Continuation." 5.5 in A Course in Modern Analysis*, 4th ed. Cambridge, England: Cambridge University Press, 96-98, 1990.

Received 5th July 2008