Undecidability Over Continuous-time

JERZY MYCKA, *Institute of Mathematics, University of Maria Curie-Sklodowska, Lublin, Poland, E-mail: Jerzy.Mycka@umcs.lublin.pl*

JOSÉ FÉLIX COSTA, *Department of Mathematics, I.S.T., Universidade Técnica de Lisboa, Lisboa, Portugal, E-mail: fgc@math.ist.utl.pt*

Abstract

Since 1996, some models of recursive functions over the real numbers have been analyzed by several researchers. It could be expected that they exhibit a computational power much greater than that of Turing machines (as other well known models of computation over the real numbers already considered in the past fifteen years, like neural net models with real weights). The fact is that *they have not got* such a power. Although they decide the classical halting problem of Turing machines, they have almost the same limitations of Turing machines. Our profit on them has been to represent classical complexity classes (like P or NP) by analytical means, and possibly relate them by unusual ways.

Keywords: decidability, analog computation, real recursive functions

1 Introduction and motivation

The first presentation of the theory of recursive functions over the reals, analogous to Kleene's classical theory, was attempted by Cristopher Moore [9]. Real recursive functions are generated by a fundamental operator, called differential recursion. The other fundamental operator is the taking of infinite limits [12]. Between 1996 (since Moore's seminal paper) and 2002, we have been working with the single concept of *differential recursion*. In [4] it is shown that *a linearization of the differential recursion scheme* gives rise to an analog characterization of the class of (Kalmar's) elementary functions and of the Grzegorczyk hierarchy. In [3] and [8] it is proved that the GPAC (**G**eneral **P**urpose **A**nalog **C**omputer) of Claude Shannon [18] is not closed under iteration and that a subclass of real recursive functions coincides with the class of GPAC-computable functions (as in [18, 16]). In [12] we finally show how to capture higher computational classes through the limit operator. Manuel Campagnolo, a PhD student of one of the authors, showed also in [2] that other computational complexity classes can be captured through appropriate structured differential schemata or by adding simple (bounded) integration.

In [13] we try to show that our framework is versatile: from a careful and not so complex definition of the (countable) set of recursive functions over the reals we show by means of the toolbox of Analysis that: (a) Laplace transform can be used to quickly obtain useful real recursive functions and to measure their rate of growth, (b) the embedding of Turing machines into real recursive functions is trivial, (c) a (limit)

hierarchy of real recursive functions exist to classify hardness of functions.

However, nothing was said until now about the limitations of the model. Surprisingly, it can be, together with its variants, very well delimited using exactly the classical recipes of computability theory. We prove some undecidability properties of our model, together with useful undecidability theorems. We further provide a classification of uncountably many non-decidable properties and link our model of continuous-time computation with the properties of unpredictability and undecidability uncovered in the discrete-time and continuous-time dynamic systems by Cris Moore (see [10]) and Newton da Costa and Francisco Dória (see [15]) in the beginning of the nineties. Thus, our framework is a hyper-computation model that decides only a countable number of sets. We believe that our model, with constructs (*differential recursion*, *infinite limits*) that can be fine tuned *ad hoc*, is a good model for hyper-computation research.

We think that this paper *is really important within the context of our work* [2, 3, 4, 8, 9, 12, 13] for two main reasons: (a) it provides a link between a non-standard model of computation with the classical theory of computability and (b) it removes the wrong impression that working with real numbers rends all subsets of $\mathbb{N}$ and all functions over $\mathbb{N}$ computable as neural net models with real weights do (see, e.g., [19]).[1]

2 Real recursive functions

We give a definition of real recursive functions (see [13] for details), which is a derivative of the original definition found in [9] (without the limit taking operator). However, it is invented to provide greater expressive power and robustness. The set of real recursive functions from $\mathbb{R}^m$ to $\mathbb{R}^n$, for all non-negative integers m and n, is inductively defined as follows.[2]

Definition 2.1
The set of real recursive vector functions is generated from the real recursive scalars 0, 1, -1 and the real recursive projections $I_n^i(x_1, \ldots, x_n) = x_i,\ 1 \leq i \leq n,\ n > 0$, by the following operators:

composition: if f is a real recursive vector function with n k-ary components and g is a real recursive vector function with k m-ary components, then the vector function with n m-ary components, $1 \leq i \leq n$, $\lambda x_1 \ldots \lambda x_m.\ f_i(g_1(x_1, \ldots, x_m), \ldots, g_k(x_1, \ldots, x_m))$ is real recursive too.

differential recursion: if f is a real recursive vector function with n k-ary components and g is a real recursive vector function with n $(k + n + 1)$-ary components, then the vector function h of n $(k + 1)$-ary components which is the solution of the Cauchy problem, $1 \leq i \leq n$, $h_i(x_1, \ldots, x_k, 0) = f_i(x_1, \ldots, x_k)$, $\partial_y h_i(x_1, \ldots, x_k, y) = g_i(x_1, \ldots, x_k, y, h_1(x_1, \ldots, x_k, y), \ldots, h_n(x_1, \ldots, x_k, y))$ is a real recursive vector function whenever each component of h is of the class C^1 on the largest interval containing 0 in which a unique solution exists.

infinite limits: if f is a real recursive vector function with n $(k + 1)$-ary components, then the vector functions h, h', h'' with n k-ary components, $1 \leq i \leq n$,

[1] Although the $ARNN$ model was not conceived only as a super-Turing device and has many interesting structural complexity relations.

[2] It is important to notice that the following definition is based on vector operations.

$h_i(x_1,\ldots,x_k) = \lim_{y\to\infty} f_i(x_1,\ldots,x_k,y)$, $h'_i(x_1,\ldots,x_k) = \liminf_{y\to\infty} f_i(x_1,\ldots,x_k,y)$, $h''_i(x_1,\ldots,x_k) = \limsup_{y\to\infty} f_i(x_1,\ldots,x_k,y)$, are real recursive too.

Arbitrary real recursive vector functions can be defined by assembling scalar real recursive function components. If f is a real recursive vector function, than each of its components is a real recursive scalar function.

For differential recursion we restrict the domain to an interval of continuity. This will preserve the analytic character of functions being defined. Let us point out the fact that this definition has as feature the property of a real recursive computable equation relation. It is not a general case for an analog computation. Constant functions 0_n, 1_n, -1_n which are n-ary can be derived from unary constant functions by means of projections. Unary constant functions can be derived by differential recursions. Let us present two examples of real recursive function definitions: $+(x,0) = I^1_1(x) = x$, $\partial_y + (x,y) = 1_3(x,y,+(x,y))$, $\delta(x) = \liminf_{y\to\infty}(\frac{1}{1+x^2})^y$.

In [12] we prove that all Turing computable functions have real recursive extensions. Thus whenever we have a computable function in the classical sense, we can immediately consider one of its real recursive extensions.

Let us denote by $\pi : \mathbb{N}^2 \to \mathbb{N}$ the standard pair coding with $\pi_1, \pi_2 : \mathbb{N} \to \mathbb{N}$ such that $\pi_1(\pi(m,n)) = m, \pi_2(\pi(m,n)) = n$. E. g., we take the bijection $\pi(m,n) = 2^m(2n+1) - 1$, and their inverses $\pi_1(x) = (x+1)_1$ (first prime projection [3]) and $\pi_2(x) = \frac{1}{2}(\frac{x+1}{2^{\pi_1(x)}} - 1)$. Of course, in the sense of our last paragraph, π, π_1, and π_2 are extensible to real recursive functions. Also real recursive is the extension to the real numbers of the bijection $\tau : \bigcup_{n>0} \mathbb{N}^n \to \mathbb{N}$ given by $\tau(n_1,\ldots,n_k) = 2^{n_1} + 2^{n_1+n_2+1} + \ldots + 2^{n_1+\ldots+n_k+k-1} - 1$.

Definition 2.2

The collection of descriptions is inductively defined as follows: i^j_n is a n-ary description of I^j_n, $1 \le j \le n$; 1_n is a n-ary description of $\lambda x_1 \ldots x_n.\ 1$, $\bar{1}_n$ is a n-ary description of $\lambda x_1 \ldots x_n.\ -1$, 0_n is a n-ary description of $\lambda x_1 \ldots x_n.\ 0$;

if $\langle h\rangle = \langle h_1,\ldots,h_m\rangle$ is a k-ary description of the real recursive vector function h and $\langle g\rangle = \langle g_1,\ldots,g_k\rangle$ is a n-ary description of the real recursive vector function g, then $c(\langle h\rangle,\langle g\rangle)$ is a n-ary description of the composition of h and g;

if $\langle h\rangle = \langle h_1,\ldots,h_n\rangle$ is a k-ary description of the real recursive vector function h and $\langle g\rangle = \langle g_1,\ldots,g_n\rangle$ is a $k+n+1$-ary description of the real recursive vector function g, then $dr(\langle h\rangle,\langle g\rangle)$ is a $k+1$-ary description of the solution of a differential recursion for h, g (if such a solution exists);

if $\langle h\rangle = \langle h_1,\ldots,h_m\rangle$ is a $n+1$-ary description of the real recursive vector function h, then $l(\langle h\rangle)$, $li(\langle h\rangle)$, $ls(\langle h\rangle)$ is a n-ary description of an appropriate infinite limit (respectively lim, $\liminf$, $\limsup$) of h;

if $\langle f_1\rangle,\ldots,\langle f_m\rangle$ are n-ary descriptions of real recursive k-ary scalars $f_1,\ \ldots,\ f_m$, then $v(\langle f_1,\rangle,\ldots,\langle f_m\rangle)$ is a k-ary description of the real recursive vector function $f = (f_1,\ldots,f_m)$.

Example 2.3

Let us exhibit some descriptions as example: $\lambda x\lambda y.\ x+y$ has the description $dr(i^1_1, 1_3)$, $\lambda x.\ x-1$ then has the description $c(dr(i^1_1,1_3), v(i^1_1,\bar{1}_1))$ and $dr(0_1, i^1_3)$ corresponds to $\lambda x\lambda y.\ xy$. Consequently $\lambda x.\ x^2$ has the description $c(dr(0_1,i^1_3), v(i^1_1,i^1_1))$.

[3] $(x)_1$ denotes here the greatest n such that 2^n divides x.

Vector functions can be assembled using the code given by τ applied to the list of codes of their components. Vice-versa, giving a vector function code, we can effectively compute the code of any of its components. Constants receive code numbers $3n$ (for -1_n), $3n+1$ (for 0_n), and $3n+2$ (for 1_n). Projections receive code $\pi(m,n)$, where m is the arity of the projection and n is the component being projected. Composition can be coded as $\pi(m,n)$, where m is the code of first vector function and n is the code of second vector function. The same holds for recursion. Since we have three different limit operators, we code them in the sequences $3n$, $3n+1$, and $3n+2$, where n is the code of the given description. We have *constants*, *projections*, *composition*, *differential recursion*, and *taking of limits*, so the final code is $5n$ (for constants), $5n+1$ (for projections), $5n+2$ (for composition), $5n+3$ (for differential recursion), and $5n+4$ (for limit constructor), where n is the code of the corresponding construct. Because the arity of the given description can be effectively computed, the final code will be taken to be $\pi(m,n)$, where m is the arity and n is the code of the description. This final step is not really needed, but in this way, we can look at non-negative integers as codes of 0-ary descriptions, of 1-ary descriptions, of 2-ary descriptions, and so on.

For a given non-negative integer n, $\{n\}$ stands for the real recursive function which code is n. The two projections of n, given by $\pi_1(n)$ and $\pi_2(n)$, respectively, provide the arity of the function and its description code. If something fails in decoding, like the first projection of n not corresponding to the arity of the description coded in the second projection, or if different descriptions of non-compatible arity are put together in a composition or in a differential recursion, then we assume the convention that $\{n\}$ stands for the function everywhere undefined. All these tasks can be fulfilled syntactically and are effectively computable. By ϕ_n^m we denote the m-ary real recursive function of code n. If m and n are not compatible or if n is the code of a ill-defined description, then ϕ_n^m denotes the function everywhere undefined. If we don't care about arity, then we just write ϕ_e, meaning $\phi_{\pi_2(e)}^{\pi_1(e)}$

By means of descriptions we can give the definition of η-hierarchy [4] as a family of classes $H_j = \{f : \eta(f) \leq j\}$. If $\eta(f) = j$, then j is the number of nested (non-parallel) infinite limits needed in the definition of f. It will be comfortable to think about the η-hierarchy as the measure of the difficulty of real recursive functions.

Example 2.4
The functions $+$, $\times$, exp, sin, cos, $\lambda x.\ \frac{1}{x}$, $/$, log, $\lambda x \lambda y.\ x^y$ are in H_0, the Kronecker δ function, the Heaviside Θ function, the signum function and absolute value are in H_1. The Bessel functions of the first kind J_v of order v (integer) are real recursive functions of the class H_0. The Euler's Γ-function is a real recursive function from the class H_1 and the Riemann ζ-function is a real recursive function from the class H_1.

Descriptions can be seen as programs. Classical descriptions of computable functions (see [7]) can be compiled into Turing machines, e.g., composition is compiled as sequential composition of programs, recurrence definitions are compiled as *FOR* loops, and minimalization is compiled as a *WHILE* loop for which termination is not guaranteed. In a similar way, analog descriptions can (partially) be compiled into analog engine units. We provided a proof in [8] that a larger collection of recursive functions over the reals not involving *limits* can be interpreted as GPAC circuits, Claude Shannon's model of the Differential Analyzer built by Vannevar Bush - the

[4] For a precise definition see [12].

analog computer of the forties, fifties, and sixties. In [17], Rubel proposes an extension of the GPAC, called the EAC (**E**xtended **A**nalog **C**omputer), where some limits can be interpreted too. When compared to the classical model, we see that our model has the following feature: in classical Turing / Kleene's theory all programs correspond to descriptions and vice-versa all descriptions correspond to programs; in our framework *all programs* (all GPAC circuits, possibly all EAC circuits) correspond to descriptions but we have more descriptions than programs.[5] These extra descriptions correspond to *hyper-analog computers*. Along the lines of this work we take descriptions as programs. And *since in the GPAC circuits variables can be connected with time, we talk about a continuous-time model of computation.*

3 Forever undecided

We know that the set of real valued functions is uncountable, and that the subset of functions mapping integers to integers is uncountable too. However the set of descriptions is countable. Then we conclude immediately the two following facts: (a) there are uncountable many functions that are not real recursive and (b) there are uncountable many functions that map integers to integers and are not real recursive functions.[6] In this section, we will focus in particular cases of functions of this kind, showing that classical common non-computable functions obtained through diagonalization can be lifted to the realm of the continuous, *disclaiming that our model is too strong, although as the reader can see in [12], it is stronger than the conventional models of computation.*

In the previous work [12, 13] is stated and proved (for the first time) that Moore's model of computation [9] once equipped with infinite limits exhibits the following property: *the halting problem is decidable by a real recursive characteristic function.* This result means that there exist real recursive functions that, when restricted to the non-negative integers, provide the characteristic of the classical halting problem, where non-negative integers are taken as the codes of deterministic Turing machines.

We know that in classical recursion theory (*à la Kleene*), the problem of deciding if a natural number belongs to the domain of a given (by its Gödel number) function (of arity one) is undecidable. The same result holds for the real recursive functions (one component of arity one). Let us denote by the problem of domain the following task: for some given real recursive function f and its description coded as the natural number n, is it possible to check whether $x \in \mathbb{R}$ is in the domain of f?

Proposition 3.1
The *problem of domain* is undecidable by real recursive functions.

Proof. Let us consider for simplicity (without any loss of generality) only real recursive functions of arity one. Let $D : \mathbb{N} \times \mathbb{R} \to \{0, 1\}$ be the characteristic function for the problem $\lambda n \lambda x$. "$\{n\}(x)$ is defined". Consider that D has a real recursive extension with the same name. If $d(n, x) = \frac{1}{1-D(n,x)}$, then by contraction we can define $\phi(x) = d(x, x)$. If D is a real recursive function (if it has some description), then ϕ has some code k. Now $\phi(k)$ is defined if and only if $D(k, k) = 0$, hence if and only

[5] Note that a program in the last sentence means a realizable physical device like the GPAC.

[6] Also, we can state that there are uncountable many functions that map "computable" numbers to "computable numbers" and are not real recursive.

if $\phi(k)$ is undefined. This is a contradiction, which means that D cannot be a real recursive function. ∎

We also know that, in classical recursion theory, the problem of deciding whether two given natural numbers are codes for the same function (of arity one) is undecidable. Once again, the same result holds for the real recursive functions (one component of arity one). Here we think about the problem of identity as the following question. For two given real recursive functions f_1, f_2 and their descriptions coded as the natural numbers n_1, n_2, is it possible to check whether these functions have the same domain and give always the same values for the same arguments?

PROPOSITION 3.2
The *problem of identity* of two real recursive functions given by their codes is undecidable by a real recursive function.

PROOF. Suppose that a real recursive extension of such a predicate I is given such that its restriction to the non-negative integers is given by $I = \lambda x \lambda m \lambda n.\text{“}\{n\}(x) = \{m\}(x)\text{”}$. Then for given real recursive function f we can define the real recursive function $0_1(f(x))$ corresponding to some code m, which for every x is equal to 0 or undefined. Now let m_c be a code of function $\lambda x.\ \delta(\frac{1}{x-c})$, for some real recursive constant c. This function has value 0 everywhere except at c, where it is undefined. To check whether the function f is defined at c it is sufficient to compute $I(c, m, m_c)$. However, from the proof of the previous proposition we know that such process is impossible for real recursive functions, hence I cannot be a real recursive function. ∎

We will say that a description is elegant (a concept introduced by Chaitin) if it has the property that no smaller description exists. Looking at the size of descriptions introduced in 2.2, we can size them in quite different ways. As happens with Chaitin's elegant LISP programs (e.g., [5]) we cannot prove that a description is elegant.

Let us assume some computable linear orders in the set of description numbers (i.e., in $\mathbb{N}$). Examples of such a order can be: $m \prec n$: lexicographical order (in the symbols used to assemble descriptions); $m \prec_s n$ - we compare the number of limits involved in the descriptions of codes m and n, then the number of differential recursions, then the number of compositions, and in the case that all numbers are equal we use the lexicographical order. All mentioned linear orders are computable, i.e., they can be decided by classical recursive functions, hence by real recursive functions too.

In classical recursion theory the total function that for each number n gives the least number m which codes for the same function as n is not computable. The result holds for the real recursive functions (one component of arity one). Let us describe the problem of minimal code in the following manner: for a fixed order $\prec$ of description numbers and some function f_1 given by its code n_1 we would like to know whether there exists another function f_2 with the code n_2 such that f_1, f_2 have the same extension and moreover $n_2 \prec n_1$?

PROPOSITION 3.3
The *problem of minimal code* is undecidable by real recursive functions.

PROOF. Consider the function $M : \mathbb{N} \to \mathbb{N}$ such that

$$M(n) = \min_{i \in \mathbb{N}} \{n_i : n_i \textit{ are codes for the same function as } n\},$$

where the minimum is given in the sense of some linear computable order discussed previously. If M has a real recursive extension, then the test for identity I of two functions given by their codes at a real recursive point x is real recursive too, since it is a real recursive extension of the predicate "$M(m') = M(n')$", where m', n' are codes of $\{m\}$, $\{n\}$ at x, respectively. This is a contradiction since I does not have real recursive extensions. Hence M is not real recursive. ∎

We give a corollary of the above proposition. In the case of the order $\prec_s$, the function M in the previous proof gives us information about the minimal number of limits needed to define a given function. Since M is not real recursive, we can conclude that the problem of establishing the minimal level of the η-hierarchy (no matter the number of differential recursions and compositions) is not decidable in our framework.

A kind of minimalization μ-operator can be defined over real recursive functions, derived from limits in [11], providing a real recursive function from a real recursive function, searching for zeros in the last argument of a function. The shortest definition of this operator (see [9, 11] for details) is as follows: $\mu_y f(x_1, \ldots, x_k, y) = \inf\{y : f(x_1, \ldots, x_k, y) = 0\}$, where infimum chooses the number y with the smallest absolute value and for two y with the same absolute value the negative one. In [11] we prove a variant of the following result. *If f is a real recursive vector with n $(k+1)$-ary components, then the vector h with n k-ary components $h_i(x_1, \ldots, x_k) = \mu_y(f_i(x_1, \ldots, x_k, y) = 0)$, $1 \leq i \leq n$, is real recursive.* With minimalization, the function everywhere undefined $\lambda x.\ \perp$ becomes trivially real recursive, e.g., as $\mu_y(|y| + 1 = 0)$.

A real recursive number can be identified with a real recursive function (this terminology can be found in [1]) with arity zero (a real recursive constant [7]). Examples of real recursive numbers are the integers, but also the rationals, and the algebraic numbers (see [13] for a full proof). Transcendental numbers such as π, Nepper's e, Euler's γ, but also Chaitin's halting probability Ω are real recursive in different levels of the limit hierarchy. However, their number is countable. As analog quantities these numbers are given once and for all as entire quantities (see [9]) and not as decimal expansions as the usual definition using Turing machines. Let us consider the following problem of deciding whether a given real is real recursive: for a given $x \in \mathbb{R}$, can we obtain x as a real recursive constant?

Proposition 3.4
The *problem of deciding whether a given real number is real recursive* is undecidable by any real recursive function.

Proof. Suppose that the characteristic of this problem $\chi : \mathbb{R} \to \{0, 1\}$ is a real recursive function. We have that $\chi(x) = 1$ if and only if $\exists n \in \mathbb{N}(\{n\}(0) = x)$. Applying the same techniques we used in [12] to construct the arithmetical hierarchy, we have that $\chi(x) = 1$ is decidable if and only if "$\{n\}(0) = x$" is decidable. But according with our proof of Proposition 3.1 we know that "$\{n\}(0) = x$" cannot be decidable. Hence χ is not real recursive and the given problem is undecidable. ∎

[7] This definition turns to be equivalent to a number $x \in \mathbb{R}$ such that there exists a real recursive function $f : \mathbb{R} \to \mathbb{R}$ of arity one with the property $f(0) = x$ (see [9]).

4 Unpredictability and undecidability in dynamic systems

Now we use another notion of the classical theory of computability in the analog context. A set $S \subseteq \mathbb{R}$ is called a real recursive set if it has a real recursive characteristic function. We proved in [13] that the set of natural numbers $\mathbb{N}$, the set of integer numbers $\mathbb{Z}$, the set of rational numbers $\mathbb{Q}$, the set of algebraic numbers are all real recursive sets. For $\mathbb{Z}$ it is sufficient to use $\delta(sin(\pi x))$ as a characteristic function $\chi_{\mathbb{Z}}$. Then $\chi_{\mathbb{N}}(x) = \chi_{\mathbb{Z}}(x)\Theta(x)$. Canonical descriptions for integers can be easily defined.

Herein we proceed to show how to adapt the same results of classical computability theory (e.g., [6]), namely Rice's theorem, that will bridge our work to Cris Moore's discovery of unpredictability and undecidability in discrete-time dynamic systems, but now in continuous-time.

PROPOSITION 4.1 (The analog *s-m-n* theorem)
For each $m,\ n \geq 1$, there is total real recursive $(m+1) - ary$ function s_n^m such that for all integer numbers $x_1,\ \ldots,\ x_m$, for all real numbers $y_1,\ \ldots,\ y_n$:

$$\phi_e^{m+n}(x_1, \ldots, x_m, y_1, \ldots, y_n) = \phi^n_{s_n^m(e,x_1,\ldots,x_m)}(y_1, \ldots, y_n).$$

PROOF. Given a description with code e, the first projection of e being $m+n$, the second projection being the code of a $(m+n)$-ary description, then we can use composition of description of code $\pi_2(e)$ with m n-ary fixed parameters (the canonical descriptions of n-ary constant integer functions $\lambda y_1 \ldots \lambda y_n.\ e_1$, .., $\lambda y_1 \ldots \lambda y_n.\ e_m$) and projections $(I_n^1, \ldots, I_n^n)$ to obtain a new description of code $s_n^m(e, e_1, \ldots, e_m)$, where the function s_n^m is computable and extensible to a real recursive function. Then our theorem holds. ∎

PROPOSITION 4.2 (The analog Rice's theorem)
Suppose that $\mathcal{B}$ is a non-empty set of real recursive functions (of arity one) not coinciding with the whole class. Then the problem $\lambda x.$ "$\phi_x \in \mathcal{B}$" is undecidable.

PROOF. We may assume without any loss of generality that the function everywhere undefined, $\lambda x.\ \perp$, does not belong to $\mathcal{B}$ (if not, we prove the result for the complement of $\mathcal{B}$). Choose the function f over non-negative integers on the first argument and over the real numbers on the second argument such that

$$f(x,y) = \begin{cases} g(y) & \{x\}(x) \text{ is defined,} \\ \perp & \text{otherwise,} \end{cases}$$

where $g \in \mathcal{B}$, and f can be extended to a real recursive function (with the same name f). Using the analog s-m-n theorem we find a function s such that $f(x,y) = \phi_{s(x)}(y)$, for non-negative integers x . Thus we have $\{x\}(x)$ is defined if and only if $\phi_{s(x)} \in \mathcal{B}$. So we have reduced the problem $\lambda x.$ "$\{x\}(x)$ is defined" to the problem $\lambda x.$ "$\phi_x \in \mathcal{B}$". We conclude that our problem is undecidable. ∎

In other words, any property that real recursive functions can have, like being injective, or onto, or having an infinite domain, or having an infinite range, or being total, is undecidable unless it is trivial (corresponding to the empty set and to the set of all real recursive functions). Now, in terms of dynamical systems, these questions concern basins of attraction. These basins are not, in general, real recursive, i.e., there is no (hyper-)algorithm that will tell us whether or not a given point belongs

to them. Considering the iteration of a real recursive function as in [12, 13], we can even easily prove a statement (Theorem 10) from [10]:

PROPOSITION 4.3 (Moore's undecidability theorem)
The following decision problems about continuous-time dynamic systems f are undecidable:

1. Given a point x and an open set A, will x fall into A (meaning $f^n(x) \in A$, for some n)?
2. Given a point x and a periodic point p (i.e., $f^n(p) = p$, for some period n), will x converge to p? Will a dense set of points converge to p?

PROOF. Apply the analog Rice's theorem. ∎

A similar result was achieved in the beginning of the nineties in the context of classical mechanics of Hamiltonian systems by Newton da Costa and Francisco Dória (see [15]), using mathematical techniques developed in the sixties by Richardson and Suppes predicates. However, in their work there is not any idea of a recursive function over the real numbers or of developing a model of computation.

5 Concluding remarks

We made use of a conceptual programming language that can be interpreted partially in analog circuits of a kind pioneered by Vannevar Bush in the thirties with his Differential Analyzer, and extended later by Lee A. Rubel in the nineties working on the Shannon's GPAC model. Our *programming language* make use of infinite limits, a mathematical concept that is not in general physical realizable. Thus (invoking informally *the physical CT Thesis*) our computational model is beyond the Turing limit, although functions remain inductively constructible and countable in number. The hyper-computational character of our model has *nothing to do with the fact that it handles real numbers* and real numbers (as in [19]) can code for non-computable functions. The hyper-computational capabilities that our model exhibit are associated with the operator of taking limits.

Although powerful it is, our model suffers from the same classical limitations as the standard Turing model of computation: the classical results of computability theory can be parameterized to our programming language to show that, e.g., the *halting problem of analog computation* is not solvable. This is quite different from [19], since neural nets with real weights are not countable and the diagonalization method of classical computability cannot be applied to a set of non-countable nets. Herein we have the *same s-m-n theorem* and the *same Rice's theorem* as in standard computability but with different interpretations.

We conceived this paper to follow [12], namely to show that the computational capabilities of the model are bounded in a similar way as the Turing model is bounded. We believe that our most general framework, involving infinite limits, have enough ingredients to allow the translation from classical computability and classical computational complexity problems into Analysis. We do believe that such translations might be a solution to open problems described in analytic terms: we are now much involved in the definition of the analog classes P and NP (see [14]). Indeed, analog characterization of classical computational complexity classes is not, as we now see

it, a strong motivation to *keep analog computation alive*. Instead, we oversee two main directions for future research. First, we may try to answer the question *what is the intrinsic computational complexity associated to continuous dynamic systems, specified by means of differential equations?* The second major line of research is to investigate if *we can solve open problems of classical computability or computational (structural) complexity using the toolbox of Analysis.*

Acknowledgments

We thank to Boaz Trakhtenbrot, Nachum Dershowitz, and Alexander Rabinovich for having invited one of the authors to a Summer School in the University of Tel Aviv, where the results of this paper were publicly presented for the first time. They are precious to us for having credited our work. We always remember Cris Moore who was responsible for our interest in this model in the middle of nineties.

References

[1] John Bell and Moshe Machover. *A Course in Mathematical Logic*, Elsevier, North-Holland, 1997.

[2] Manuel L. Campagnolo. Computational complexity of real valued recursive functions and analog circuits, PhD dissertation, Universidade Técnica de Lisboa, 2001.

[3] Manuel L. Campagnolo, Cristopher Moore, and J. Félix Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4):642-660, 2000.

[4] Manuel L. Campagnolo, Cristopher Moore, and J. Félix Costa. An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 18(4):977–1000, 2002.

[5] Gregory J. Chaitin. *The Limits of Mathematics*, Springer, 1998.

[6] Nigel J. Cutland. *Computability, An Introduction to Recursive Function Theory*, Cambridge University Press, 1992.

[7] Martin Davis. *Computability and Solvability*, Dover, 1982.

[8] Daniel Graça and J. Félix Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5): 644-664, 2003.

[9] Cristopher Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162:23-44, 1996.

[10] Cristopher Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4:199-230, 1991.

[11] Jerzy Mycka. μ-Recursion and infinite limits. *Theoretical Computer Science*, 302:123-133, 2003.

[12] Jerzy Mycka and J. Félix Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835-857, 2004.

[13] Jerzy Mycka and J. Félix Costa. The computational power of continuous dynamic systems. *Machines, Computations, and Universality (MCU 2004)*, Lecture Notes in Computer Science 3354, Springer-Verlag, 2005, 163–174.

[14] Jerzy Mycka and J. Félix Costa. The P =/= NP conjecture in the context of real and complex analysis. *Journal of Complexity*, accepted for publication.

[15] Newton C. A. da Costa and Francisco A. Dória. Undecidability and incompleteness in classical mechanics. *International Journal of Theoretical Physics*, 30:1041-1073, 1991.

[16] Marian B. Pour-El. Abstract computability and its relations to the general purpose analog computer. *Transactions Amer. Math. Soc.*, 199:1-28, 1974.

[17] Lee A. Rubel. The extended analog computer. *Advances in Applied Mathematics*, 14:39-50, 1993.

[18] Claude Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337-354, 1941.

[19] Hava T. Siegelmann and Eduardo Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131: 331-360, 1994.

Received 8 September 2004.