ELSEVIER
Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Complexity ■■■ (■■■■) ■■■–■■■
Journal of COMPLEXITY
www.elsevier.com/locate/jco

Real recursive functions and their hierarchy

Jerzy Mycka[a,*], José Félix Costa[b]

[a] Institute of Mathematics, University of Maria Curie-Sklodowska, Pl. M. Curie-Sklodowskiej 1, Lublin 20-031, Poland

[b] Department of Mathematics, I.S.T., Universidade Técnica de Lisboa, Lisboa, Portugal

Received 30 September 2003; accepted 10 June 2004

Abstract

In the last years, recursive functions over the reals (Theoret. Comput. Sci. 162 (1996) 23) have been considered, first as a model of analog computation, and second to obtain analog characterizations of classical computational complexity classes (Unconventional Models of Computation, UMC 2002, Lecture Notes in Computer Science, Vol. 2509, Springer, Berlin, pp. 1–14). However, one of the operators introduced in the seminal paper by Moore (1996), the minimalization operator, has not been considered: (a) although differential recursion (the analog counterpart of classical recurrence) is, in some extent, directly implementable in the General Purpose Analog Computer of Claude Shannon, analog minimalization is far from physical realizability, and (b) analog minimalization was borrowed from classical recursion theory and does not fit well the analytic realm of analog computation. In this paper, we show that a most natural operator captured from analysis—the operator of taking a limit—can be used properly to enhance the theory of recursion over the reals, providing good solutions to puzzling problems raised by the original model.

Keywords: ■; ■; ■

1. Introduction and motivation

The classical theory of computation deals with functions on enumerable domains (especially sets of non-negative integers). Enumerable computation has been, since the 1930s, the most important computational model, mainly due to the unifying work of Turing. Turing clarified the notion of algorithm giving it a precise meaning, and introduced a coherent

* Corresponding author.
E-mail addresses: jerzy.mycka@umcs.lublin.pl (J. Mycka), fgc@math.ist.utl.pt (J.F. Costa).

0885-064X/$ - see front matter © 2004 Published by Elsevier Inc.
doi:10.1016/j.jco.2004.06.001

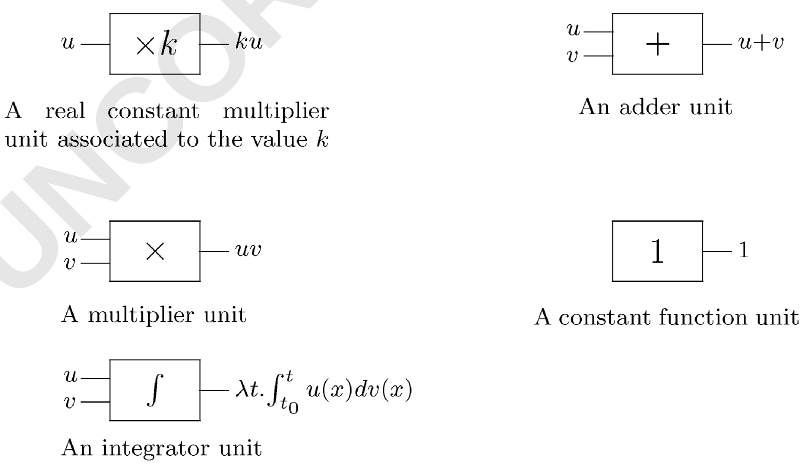
framework for discrete computation. In a short time, new results showing the relations of his model with other approaches, such as recursive functions (in the sense of Kleene) or Church's λ-calculus (for information about this subject see Odifreddi [15]), originated in a natural way consistent theoretical basis to standard computation theory.

Nevertheless, computers need not to be digital. In fact, the first computers were analog computers. In an analog computer, the internal states are continuous, rather than discrete as in digital computation. The first analog computers were especially well suited to solve ordinary differential equations. Unfortunately, because of the problem of a coherent theoretical basis to analog computation and the fact that analog computers technology almost did not improve in the last half century, when compared with its digital counterpart, analog computation was about to be forgotten.

We may classify analog models as discrete time models (e.g. [1]) or as continuous time models. In this paper, we are interested in the latter type. The basic model in this field is Shannon's General Purpose Analog Computer (GPAC) [22].

The GPAC is a computer whose computation evolves in continuous time. The outputs are generated from the inputs by means of a dependence defined by a finite directed graph (not necessarily acyclic) where each node is one of the following boxes.

- *Integrator*: A two-input, one-output unit with a setting for initial condition. If the inputs are unary functions u, v, then the output is the Riemann–Stieljes integral $\lambda t. \int_{t_0}^{t} u(x)\, \mathrm{d}v(x) + a$, where a and t_0 are real constants defined by the initial settings of the integrator.
- *Constant multiplier*: A one-input, one-output unit associated to a real number. If u is the input of a constant multiplier associated to the real number k, then the output is ku.
- *Adder*: A two-input, one-output unit. If u and v are the inputs, then the output is $u+v$.
- *Multiplier*: A two-input, one-output unit. If u and v are the inputs, then the output is uv.
- *Constant function*: A zero-input, one-output unit. The value of the output is always 1.



Representations of different types of units in a GPAC.

Although the above notion of GPAC [1] seems fairly intuitive and natural, the accepted definition is due to Pour-El and was introduced in [17]. Let us now present a precise version of her definition. In the following, I will denote a closed bounded interval with non-empty interior. We now introduce the concept of function generated by a GPAC for functions of one variable.

Definition 1. The unary function y is generated by a GPAC on I if there exist a set of unary functions $y_1, \ldots, y_n$ and a set of initial conditions $y_i(a) = y_i^*$, $i = 1, \ldots, n$, where $a \in I$, such that:

(1) $\mathbf{y} = (y_1, \ldots, y_n)$ is the unique solution on I of a system of ODEs of the form

$$A(x, \mathbf{y}) \frac{d\mathbf{y}}{dx} = b(x, \mathbf{y}) \tag{1}$$

satisfying the initial conditions, where $A(x, \mathbf{y})$ and $b(x, \mathbf{y})$ are $n \times n$ and $n \times 1$ matrices, respectively. Furthermore, each entry of A and b must be linear in $1, x, y_1, \ldots, y_n$.

(2) For some $1 \le i \le n$, $y = y_i$ on I.

(3) $(a, y_1^*, \ldots, y_n^*)$ has a domain of generation with respect to the above equation, i.e., there are closed intervals $J_0, J_1, \ldots, J_n$ (with non-empty interiors) such that $(a, y_1^*, \ldots, y_n^*)$ is an interior point of $J_0 \times J_1 \times \cdots \times J_n$ and, furthermore, whenever $(b, z_1^*, \ldots, z_n^*) \in J_0 \times J_1 \times \cdots \times J_n$, there exist unary functions $z_1, \ldots, z_n$ such that
   (i) $z_i(b) = z_i^*$ for $i = 1, \ldots, n$;
   (ii) $(z_1, \ldots, z_n)$ satisfy the Eq. (1) on some interval I^* with non-empty interior such that $b \in I^*$;
   (iii) $(z_1, \ldots, z_n)$ is unique on I^*.

The existence of a domain of generation indicates that the solution of the above equation remains unique for sufficiently small changes on the initial conditions.

Let us recall that a function $f(x)$ is differentially algebraic [20] if its derivatives satisfy a polynomial equation $P(x, f(x), \ldots, f^{(k)}(x)) = 0$ for some polynomial with rational coefficients. A function of several variables is differentially algebraic if it is a differentially algebraic function of each variable when the others are fixed. Provided with the above definition, Pour-El shows (although with some corrections made by Lipshitz and Rubel [12]), the following result:

Theorem 2. *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.*

Another important model of analog computation is Rubel's Extended Analog Computer (EAC) [21]. This model is similar to the GPAC, but we allow, in addition, other types of units, e.g. units that solve boundary value problems (here we allow several independent variables because Rubel is not seeking any equivalence with existing models). The EAC

[1] Some people believe that a model of computation which supports the setting of *real* parameters may also support hypercomputation, since the information contents of a real number is unlimited. On contrary, the computational power of the (*physical*) GPAC is not sensible to the setting of real numbers, like real constants, real multipliers, real initial conditions for integration.

permits all the operations of ordinary analysis, except the unrestricted taking of limits. The new units add an extended computational power relatively to the GPAC. For example, the EAC can solve the Dirichlet problem for Laplace's equation in the disk and can generate the Γ function (it is known that the GPAC cannot solve these problems [20]). It is not known if it exists a physical version of the EAC.

New approach was given by Moore in 1996. In the work [13], he defined a set of (vector-valued) functions on the reals (called R-recursive functions) in the analogous way to the classical recursive functions on the natural numbers. His model has also a continuous time of computation (a continuous integration instead of a discrete recursion). The class of real functions called R-recursive functions in [13] can be defined as follows:

Definition 3. The set of R-recursive vectors is generated from the R-recursive scalars $0, 1, -1$ and the R-recursive projections $I_n^i(x_1, \ldots, x_n) = x_i$, $1 \le i \le n, n > 0$, by the operators:

(1) Composition: if f is an R-recursive vector with n k-ary components and g is an R-recursive vector with k m-ary components, then the vector with n m-ary components $(1 \le i \le n)$

$$\lambda x_1 \ldots x_m . f_i(g_1(x_1, \ldots, x_m), \ldots, g_k(x_1, \ldots, x_m))$$

is R-recursive.

(2) Differential recursion: if f is an R-recursive vector with n k-ary components and g is an R-recursive vector with n $(k+n+1)$-ary components, then the vector h of n $(k+1)$-ary components which is the solution of the Cauchy problem for $1 \le i \le n$

$$h_i(x_1, \ldots, x_k, 0) = f_i(x_1, \ldots, x_k),$$

$$\partial_y h_i(x_1, \ldots, x_k, y) = g_i(x_1, \ldots, x_k, y, h_1(x_1, \ldots, x_k, y), \ldots, h_n(x_1, \ldots, x_k, y))$$

is R-recursive whenever a unique solution exists on the largest interval containing 0.

(3) μ-Recursion: if f is an R-recursive vector with n $(k+1)$-ary components, then the vector h with n k-ary components $(1 \le i \le n)$

$$h_i(x_1, \ldots, x_k) = \mu_y f_i(x_1, \ldots, x_k, y) = \inf\{y : f(x_1, \ldots, x_k, y) = 0\},$$

is R-recursive, for all $1 \le i \le n$, whenever the infimum chooses the number y with the smallest absolute value and for two y with the same absolute value the negative one.

(4) Arbitrary R-recursive vectors $f = (f_1, \ldots, f_n)$ can be defined by assembling scalar R-recursive components $f_1, \ldots, f_n$.

(5) If f is an R-recursive vector, than each of its components is an R-recursive scalar.

Exhaustive comments to the above definition will be given later. Here let us point out the fact that the set of R-recursive functions includes also partial functions. The name of R-recursive functions is used by Moore, however we should remember that in reality we have partiality here (partial R-recursive functions).

Moore's seminal paper gave rise to further development in R-recursive function theory for the following main reasons:

(A) Restricted forms of integration induce such classes of analog computation that they have counterparts in classical computation (see [3,5]).

(B) Moore did not properly identify the subclass of R-recursive functions defined without minimalization with Shannon's GPAC (cf. [13, Proposition 9]); in the paper [8] it is shown that there is a subclass of R-recursive functions matching exactly the GPAC-computable functions.

(C) Moore failed to construct the analog solution of the halting problem of classical computation. We show here that such solution exists. Moreover we prove that replacing minimalization (a counterpart of the classic concept) by infinite limits is a powerful idea, not only to elegant formulation of results, but to implement the levels of the arithmetical hierarchy into subclasses of real recursive functions. We also expect that differential recursion together with infinite limits can lift problems of classical computation to the field of mathematical analysis, which allow us to use stronger and more effective mathematical tools. It is important to remember that Moore's μ-operator can be derived from limits (see [14]), although the contrary might not be strictly true.

(D) Introduction of Heaviside Θ function as a basic function gives an iteration as a proper method of defining new functions in the field of analog computation (see [4]).

With respect to physical realizability, the drawback of Moore's paper [13] and of the present paper is the high degree of uncomputability of upper classes of (R-, real) recursive functions. The fragment identifiable to GPAC-computable functions is of course physical realizable e.g. by the differential analyser of Bush [18]. Some other functions given in [13] and therein are implementable by Rubel's EAC. However, there is unclear situation with respect to infinite limits. It is shown that in some physical models, limits have physical plausibility [9,24]. Our main purpose is devoted to find the place of classical computability notions in the analog realm. Then the new methods and tools can be used to analyse the well-known problems of computability.

Let us analyse closely aspects of the definition of R-recursive functions given by Moore [13]. One of its operators is the differential recursion. In the scalar case the operator defines a new function $h : R^{n+1} \to R$ given by the following equations: $h(\bar{x}, 0) = f(\bar{x})$, $\partial_y h(\bar{x}, y) = g(\bar{x}, y, h(\bar{x}, y))$. However, this operator creates some difficulties.

The problem of the interval of the definition: A solution of a differential equation need not to be unique or can diverge. Hence, we have to assume that h is defined only where a finite and unique solution exists.

This form of the definition is also not free from problems. Let us start with the equation $h(0) = 0$, $\partial_x h(x) = x/h(x)$, its solution is $\sqrt{x^2} = |x|$. In a similar manner we can obtain the sawtooth function as $\sin^{-1}(\sin x)$ as a solution of $\partial_x h(x) = \frac{\cos x}{\cos h(x)}$. We get non-analytic functions in both cases. This fact contradicts Moore's statement about analyticity of R-recursive functions defined without μ-recursion.

The natural connection, which can be expected, between R-recursive functions defined without μ-recursion and GPAC-computable functions is also broken by these mentioned functions. The situation is more problematic because we can define also such functions which are C^∞ but non-differentially algebraic (hence not GPAC-computable). We can observe this situation in the example $f(1) = \frac{1}{e}$, $\partial_x f(x) = f(x)/x^2$. The only continuous function f is given by

$$\lambda x.\begin{cases}\exp(\frac{-1}{x}) & x>0\\ 0 & x\le 0,\end{cases}$$

which is in fact non-GPAC-computable. The troubles arise from the full unbounded form of an integration. Such operation can lead us to functions which derivatives are not continuous.

Undefined-value problem: The Moore's approach has also another not obvious feature. We can find an assumption in his paper that $f(x)\cdot 0=0$ even when $f(x)$ is undefined or reaches infinity (see [13]). It is not a standard mathematical method to proceed in the case of such compositions. Also from the physics point of view it is doubtful because it involves infinite amount of resources (energy, forces).

The zero-value problem: The last remark is important especially in the case of partial functions. The problem, whether or not some point belongs to the domain is significant. For that purpose, Moore proposes the η operator, which also allows us to convert partial functions into total ones. Let us recall his definition.

Definition 4. For any function $f: R^{n+1}\to R$ let

$$\eta_y f(\bar{x},y)=\begin{cases}1 & \exists\, y f(\bar{x},y)=0,\\ 0 & \forall\, y f(\bar{x},y)\neq 0.\end{cases}$$

In his work, Moore proves that for any R-recursive function f the respective function given by the η operator, ηf is R-recursive too. But, in the proof the mentioned property (multiplication of infinity by 0) plays the main role. The importance of the η operator is significant. With its help it is possible to solve the halting problem for Turing machines and other undecidable problems. But such operation on undefined functions which is used for η makes the results not believable.

We should explain explicitly the minimalization operator. First, if an infinite number of zeros accumulate just above some positive y or just below some negative y, then the infimum operation returns that y even if it itself is not a zero. It can find zero also when they are isolated and discontinuous. Let us observe that μ-operator is borrowed from classical recursion theory. It adds computational power to the mentioned system. However, we cannot find the proper analogous construction in the known models of analog computation (GPAC, EAC). Meanwhile its physical realizability is uncertain.

2. Recursive functions over the reals with infinite limits

We give a new definition of real recursive functions, which is a derivative of the original definition found in [13]. However, it is invented to avoid problems involved in the latter. It is important to see that the following definition is based on the vector operations (a variation of Moore's definition).

Definition 5. The set of real recursive vectors is generated from the real recursive scalars $0, 1, -1$ and the real recursive projections $I_n^i(x_1,\ldots,x_n)=x_i$, $1\le i\le n$, $n>0$, by the operators:

(1) Composition: if f is a real recursive vector with n k-ary components and g is a real recursive vector with k m-ary components, then the vector with n m-ary components $(1 \le i \le n)$

$$\lambda x_1 \ldots x_m . f_i(g_1(x_1, \ldots, x_m), \ldots, g_k(x_1, \ldots, x_m))$$

is real recursive.

(2) Differential recursion: if f is a real recursive vector with n k-ary components and g is a real recursive vector with n $(k+n+1)$-ary components, then the vector h of n $(k+1)$-ary components which is the solution of the Cauchy problem for $1 \le i \le n$

$$h_i(x_1, \ldots, x_k, 0) = f_i(x_1, \ldots, x_k),$$

$$\partial_y h_i(x_1, \ldots, x_k, y) = g_i(x_1, \ldots, x_k, y, h_1(x_1, \ldots, x_k, y), \ldots, h_n(x_1, \ldots, x_k, y))$$

is real recursive whenever h is of the class C^1 on the largest interval containing 0 in which a unique solution exists. [2]

(3) Infinite limits: if f is a real recursive vector with n $(k+1)$-ary components, then the vectors h, h', h'' with n k-ary components $(1 \le i \le n)$

$$h_i(x_1, \ldots, x_k) = \lim_{y \to \infty} f_i(x_1, \ldots, x_k, y),$$

$$h'_i(x_1, \ldots, x_k) = \liminf_{y \to \infty} f_i(x_1, \ldots, x_k, y),$$

$$h''_i(x_1, \ldots, x_k) = \limsup_{y \to \infty} f_i(x_1, \ldots, x_k, y),$$

are real recursive in the domain containing these points, where these limits exist for all $1 \le i \le n$. [3]

(4) Arbitrary real recursive vectors can be defined by assembling scalar real recursive components.

(5) If f is a real recursive vector, than each of its components is a real recursive scalar.

[2] Why is $h(x) = |x|$ not in the system (using only differential recursion)? If we start with the Cauchy problem $f(0) = 1$, $\partial_y f(y) = 1/(2f(y))$, then we get $f(y) = \sqrt{y+1}$ defined in $(-1, \infty)$ (this interval contains 0). First, compose with the computable function $j(y) = y - 1$ defined in R, to get $k(y) = \sqrt{y}$ defined in $(0, \infty)$. Then we compose k with the solution of the Cauchy problem $g(0) = 0$, $\partial_y g(y) = 2y$, i.e., $g(y) = y^2$ defined in R, to obtain $h(x) = \sqrt{x^2}$ defined in $(0, \infty)$, which by the way is $h(x) = x$ and not $h(x) = |x|$. This means that composition will not allow to introduce non-analytic functions. Solution $\lambda x.|x|$ of the Cauchy problem $h(0) = 0$, $\partial_y h(y) = y/h(y)$ is not accepted because $\partial_y h(y)$ is not defined in the origin.

[3] These concepts are defined in the completion of the real numbers $R \cup \{-\infty, +\infty\}$. Let the function f be defined on a metric space S and assume real values. If $x_0 \in S$ and $O(x_0, \varepsilon)$ is a neighbourhood of x_0, then we define (see [7]) $\limsup_{x \to x_0} f(x) = \lim_{\varepsilon \to 0}[\sup_{x \in O(x_0, \varepsilon)} f(x)]$ and $\liminf_{x \to x_0} f(x) = \lim_{\varepsilon \to 0}[\inf_{x \in O(x_0, \varepsilon)} f(x)]$. In infinity we have then $\limsup_{y \to \infty} f(x) = \lim_{y \to \infty}[\sup_{x > y} f(x)]$, $\liminf_{y \to \infty} f(x) = \lim_{y \to \infty}[\inf_{x > y} f(x)]$. Because $\lambda y.[\sup_{x > y} f(x)]$ is a non-increasing function and $\lambda y.[\inf_{x > y} f(x)]$ is a non-decreasing function thus $\lim_{y \to \infty}[\sup_{x > y} f(x)] = \inf_y[\sup_{x > y} f(x)]$, $\lim_{y \to \infty}[\inf_{x > y} f(x)] = \sup_y[\inf_{x > y} f(x)]$. If $\lim_{x \to \infty} f(x)$ exists, then $\liminf_{x \to \infty} f(x) = \lim_{x \to \infty} f(x) = \limsup_{x \to \infty} f(x)$. It is important to remember that lim sup, lim inf are always defined (in the above completion).

Let us discuss carefully the definition. For differential recursion we restrict a domain to an interval of continuity. This will preserve the analyticity of functions in the process of defining. This eliminates a possibility of defining such functions as $\lambda x.|x|$.

Let us point out the fact that this definition has as its 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. For example, $1_k(x_1, \ldots, x_k) = 1$ can be defined as $1_1(I_k^1(x_1, \ldots, x_k)) = 1$. Unary constant functions can be derived by differential recursions: $0(0) = 0$, $\partial_y 0(y) = I_2^2(y, 0(y))$; $u(0) = c$, $\partial_y u(y) = 0(I_2^1(y, u(y)))$, where $c = 1, -1$.

From the physical point of view with such definition we are ready to use only a finite amount of energy. We excluded here the possibility of operations on undefined functions: our functions are strict in the meaning that for undefined arguments they are also undefined. But to obtain some interesting functions (like the mentioned η-function) we should improve the power of our system by an addition of the operators of infinite limits. Let us point out that introducing of infinite limits gets discontinuous functions.

We should also remember that in some cases we can use limits in some real point. This is possible by transforming them into infinite limits. For example, $\lim_{y \to \frac{\pi}{2}} \sin xy$ can be written as $\lim_{y\to\infty} \sin x(\arctan y)$.

To illustrate further this transformation let us point out that if f is a $(n+1)$-ary real recursive function, then its derivative
$\partial_y f(x_1, \ldots, x_n, y) = \lim_{\omega\to\infty} \omega(f(x_1, \ldots, x_n, y + \frac{1}{\omega}) - f(x_1, \ldots, x_n, y))$ is real recursive function, whenever such a limit exists. For example, if we take $\lambda y.1/y$ then $\lim_{\omega\to\infty} (1/(y + \frac{1}{\omega}) - 1/y)\omega = \lim_{\omega\to\infty} \omega(y - y - \frac{1}{\omega})/[(y + \frac{1}{\omega})y] = -1/y^2$ is a real recursive function.

Derivatives are physical realizable: the class of differential algebraic functions is closed under derivatives, making a large class of derivatives physical realizable. Since the extended analog computer also is close to physical implementation,[4] the larger class of EAC-computable functions are also closed under derivatives.

Let us give some examples of functions generated with the definition of real recursive functions.

Proposition 6. *The functions* $+, \times, -, \exp, \sin, \cos, \lambda x.\frac{1}{x}, /, \ln, \lambda xy.x^y$ *are real recursive functions.*

Proof. Let us define $+(x, 0) = I_1^1(x) = x$, $\partial_y + (x, y) = 1_3(x, y, +(x, y))$. Analogously we can get $\times(x, 0) = 0_1(x)$, $\partial_y \times (x, y) = I_3^1(x, y, \times(x, y))$, hence we have by a composition $-(x, y) = +(x, \times(-1, y))$. The function of an exponentation can be defined as $\exp(0) = 1$, $\partial_y \exp(y) = I_2^2(y, \exp(y))$. Furthermore, vector $(\sin(x), \cos(x))$ and its components can be defined by such differential recursion:

$$\begin{pmatrix} \sin \\ \cos \end{pmatrix}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \partial_y \begin{pmatrix} \sin \\ \cos \end{pmatrix}(y) = \begin{pmatrix} I_3^3 \\ -I_3^2 \end{pmatrix}(y, \sin y, \cos y).$$

[4] As some extension of GPAC, for details see [21].

Now for $\lambda x.\frac{1}{x}$ we define $h(x) = \frac{1}{x+1}$ in the following way: $h(0) = 1$, $\partial_x h(x) = \times(-1, \times(h(x), h(x)))$ (h is defined in the interval $(-1, \infty)$) and later we can compose h with $-(x, 1)$. The division is simply a composition of $\times$ and $\frac{1}{x}$ (with the domain equal to $(0, \infty)$, but we can extend the division to the negative numbers via a definition by cases). In the case of $\ln(x)$, we start with definition $\ln(x+1)$ by $\ln(1) = 0$, $\partial_x \ln(x+1) = \frac{1}{I_2^1(x, \ln x)+1}$ to finish with a translation of argument, next $x^0 = 1_1(x)$, $\partial_y x^y = g(x, y, x^y) = \ln x \cdot x^y$. □

We can construct also other special real recursive functions.

Proposition 7. *The Kronecker δ function, the signum function, and absolute value are real recursive functions. The Heaviside Θ function (equal to 1 if $x \geq 0$, otherwise 0), the binary maximum* max, *the square-wave function s, the function p such that $p(x) = 1$ for $x \in [2n, 2n+1)$ and $p(x) = 0$ for $x \in [2n+1, 2n+2)$, and the floor function are all real recursive too.*

Proof. Here we will deal with a less rigor in definitions, however, they always can be transformed into a strict form. It is sufficient to take the following definitions: if $\delta(0) = 1$ and for all $x \neq 0$ we have $\delta(x) = 0$, then let us define $\delta(x) = \liminf_{y\to\infty}(\frac{1}{1+x^2})^y$. From the function $\lambda x\, y.\frac{2}{1+\exp(-xy)} - 1$, we obtain

$$sgn(x) = \liminf_{y\to\infty} \frac{2}{1+\exp(-xy)} - 1 = \begin{cases} 1, & x > 0 \\ 0, & x = 0, \\ -1, & x < 0 \end{cases} \quad \text{and } |x| = sgn(x)x.$$

Let $\Theta(x) = (sgn(x) + \delta(x) + 1)/2$, $\max(x, y) = y + (x-y)\Theta(x-y)$, $s(x) = \Theta(\sin(\pi x))$.

The function p can be given by $\lambda x.s(x)(1 - \delta(\sin(\frac{(x-1)\pi}{2})))$.

Finally, the floor function has the below definition

$$\lfloor x \rfloor = w(x)p(2x) + w\left(x - \tfrac{1}{2}\right)(1 - p(2x)),$$

where $w(x) = j$ if $x \in [j, j + \frac{1}{2})$. Such function w can be defined by the differential recursion: $w(0) = 0$, $\partial_x w(x) = 4\sin^2 2\pi x\,\Theta(-\sin 2\pi x)$. □

Function g of paragraph (2) in the Definition 5 can exhibit quite different dependencies on its variables. Consider a scalar function of two variables for three different cases in $\partial_y f(x, y) = g(x, y, f(x, y))$. Then g depends on x: $g(x, y, z) = x$ in the definition of $\lambda xy.xy$; g depends on y: $g(x, y, z) = x/(1+y^2)$ in the definition of $\lambda xy.x \arctan y$; g depends on z: $g(x, y, z) = xz$ in the definition of $\lambda xy.xe^y$.

In some examples, we can use in constructions the predicate of equality $eq = \lambda xy.\delta(x - y)$. Sometimes, we will use Θ to control whether points are in given intervals. Then for $x \in [a, \infty)$ we have the characteristic function $\Theta(x-a)$ and for $x \in [a, b]$ we can define $\Theta_{[a,b]}(x) = \Theta(x-a)\Theta(b-x)$.

Let us add that by computable reals (points) we understand values of some real recursive functions for an argument equal 0. Of course, an argument can be changed to a computable real t by a composition of given real recursive function with $x + t$. In this sense e, π are computable reals: $e = \exp(1)$, $\pi = 4 \arctan(1)$ where $\arctan(0) = 0$, $\partial y \arctan(x) = \frac{1}{1+x^2}$. Also Euler's constant $\gamma = \lim_{n\to\infty}(\sum_{k=1}^{n} \frac{1}{k} - \ln n)$ is a computable real number because it can be established by real recursive expression $-\lim_{z\to\infty} \int_0^z e^{-x} \ln x \, dx$.

In some examples we can use in constructions the predicate of equality. However, models of analog computation are not necessarily connected with the property of testing exact equality. In the case of BSS computability [1] the equality predicate is included with important consequences for the strength of this model (see [2]).

3. η-Hierarchy

Here, we approach a new problem. Are there different levels of difficulty in a computation if it goes beyond the Turing computability? The natural measure of a function's difficulty can be join with the degree of (dis)continuity. The above considerations lead us to the conception of η-hierarchy which describe the level of nesting limits in the definition of a given function.

We should start with the notion of syntactic n-ary descriptions of real recursive vectors. Let us introduce some kind of symbols called basics descriptors for all basic real recursive functions. The combination of such descriptions for given real recursive functions will form a new description of another function. Let us start with basic functions: i_k^j is a k-ary description for projection I_k^j for all $1 \leq j \leq k$; $1_k, \bar{1}_k, 0_k$ are k-ary descriptions for constants $1, -1, 0$ used with k variables. We must add also operator symbols (descriptors) for all introduced operators: dr—for a differential recursion, c—for a composition, l, ls, li for a respective kind of limits (lim, lim sup, lim inf).

Definition 8. The collection of descriptors of real recursive vectors is inductively defined as follows:

- i_n^j is a n-ary description of I_n^j, $1 \leq j \leq n \in N$;
- 1_n is a n-ary description of $f(x_1, \ldots, x_n) = 1$, for all $(x_1, \ldots, x_n) \in R^n$, $n \in N$;
- $\bar{1}_n$ is a n-ary description of $f(x_1, \ldots, x_n) = -1$, for all $(x_1, \ldots, x_n) \in R^n$, $n \in N$;
- 0_n is a n-ary description of $f(x_1, \ldots, x_n) = 0$, for all $(x_1, \ldots, x_n) \in R^n$, $n \in N$;
- if $\langle h \rangle = \langle h_1, \ldots, h_m \rangle$ is a k-ary description of the real recursive vector h and $\langle g \rangle = \langle g_1, \ldots, g_k \rangle$ is a n-ary description of the real recursive vector 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 h and $\langle g \rangle = \langle g_1, \ldots, g_n \rangle$ is a $(k+n+1)$-ary description of the real recursive vector g, then $dr(\langle h \rangle, \langle g \rangle)$ is a $(k+1)$-ary description of the function defined as in the point (2) of Definition 5;
- if $\langle h \rangle = \langle h_1, \ldots, h_m \rangle$ is a $(n+1)$-ary description of the real recursive vector 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, lim inf, lim sup) of h (defined as in the point (3) of Definition 5);
- 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 $f = (f_1, \ldots, f_m)$.

Let us give an example of a construction of descriptions.

Example 9. We will construct the description of the function $\lambda x.\frac{1}{x}$. From the construction given in the Proposition 6 we have

$\lambda x\, y.x + y$ 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 $\langle \lambda x\, y.xy\rangle = dr(0_1, i_3^1)$.
Consequently, $\lambda x.x^2$ has the description $c(dr(0_1, i_3^1), v(i_1^1, i_1^1))$
and now $\langle \lambda xz. - z^2\rangle = c(dr(0_1, i_3^1), v(\bar{1}_2, c(dr(0_1, i_3^1), v(i_2^2, i_2^2))))$.

Finally, the description of $\lambda x.\frac{1}{x}$ is equal $c(dr(1_0, \langle \lambda xz. - z^2\rangle), \langle \lambda x.x - 1\rangle)$, which has the following full form:

$$c(dr(1_0, c(dr(0_1, i_3^1), v(\bar{1}_2, c(dr(0_1, i_3^1), v(i_2^2, i_2^2))))), c(dr(i_1^1, 1_3), v(i_1^1, \bar{1}_1))).$$

Now, we can find the η-number for a description of some function *f*.

Definition 10. For a given *n*-ary description *s* of a vector *f* let $E_i^k(s)$ (the η-number with respect to *i*th variable of the *k*-component) be defined as follows:

(1) $E_i^1(0_n) = E_i^1(1_n) = E_i^1(\bar{1}_n) = 0$;
(2) $E_i^m(c(\langle h\rangle, \langle g\rangle)) = \max_{1\le j\le k}(E_j^m(\langle h\rangle) + E_i^j(\langle g_j\rangle))$, where *h* is a *n* components *k*-ary vector and *g* is a *k*-components *m*-ary vector;
(3) for a differential recursion we distinguish two cases:
- $i \le k$:

$$\begin{aligned}&(E_i^j(dr(\langle f\rangle, \langle g\rangle))\\ &\quad = \max(E_i^1(\langle f_1\rangle), \ldots, E_i^1(\langle f_n\rangle), E_i^1(\langle g_1\rangle), \ldots, E_i^1(\langle g_n\rangle),\\ &E_{k+1}^1(\langle g_1\rangle), \ldots, E_{k+1}^1(\langle g_n\rangle))\end{aligned}$$

- $i = k + 1$:

$$\begin{aligned}&E_i^j(dr(\langle f\rangle, \langle g\rangle))\\ &\quad = \max_{0\le m\le n}(\max(E_{k+m+1}^1(\langle g_1\rangle), \ldots, E_{k+m+1}^1(\langle g_n\rangle)))\end{aligned}$$

where *f* is a *n* components *k*-ary vector and *g* is a *n* components $(k + n + 1)$-ary vector;

(4) $E_i^k(l(\langle h\rangle)) = E_i^k(li(\langle h\rangle)) = E_i^k(ls(\langle h\rangle)) = \max(E_i^k(\langle h\rangle), E_{n+1}^k(\langle h\rangle)) + 1$, where *h* is a *k* components $(n + 1)$-ary vector.

The main idea of the above definition is to count nested limits in descriptions. We should distinguish in the point (3) the case $i = k + 1$ (differential recursion is given with respect to this variable); in this case $\langle f\rangle$ is not important for the counting.

For the *n*-ary description *s* of *m* components we can define now $E(\langle h\rangle) = \max_k \max_i E_i^k(\langle h\rangle)$ for $1 \le i \le n$, $1 \le k \le m$. Now, we can deal with the η-number for a real recursive functions.

Definition 11. For a given real recursive function *f*, let $\eta(f)$ be defined as the minimum of $E(\langle f\rangle)$ for all possible descriptions of the function *f*.

We are ready to conclude with definition of η-hierarchy as a family of $H_j = \{f : \eta(f) \leq j\}$. It will be comfortable to think about the η-hierarchy as the measure of the difficulty of real recursive functions. If $f \in H_j$, then j nested limits is used to define f. However, —as we can see in the next section—we can patch functions defined by infinite limits, so j can be seen as the number of nested (non-parallel) η needed to patch the function f to the total function.

Here is the way of other equivalent definition: if f is a real recursive function, then $E(f) = j$ if at most j nested η operations are necessary to create f_{total} such that f_{total} is everywhere defined and if $f(\bar{x}_0)$ is defined, then $f_{\text{total}}(\bar{x}_0) = f(\bar{x}_0)$.

Let us start with recalling of some real recursive functions from previous propositions.

Example 12. From the constructions given in Propositions 6, 7 we have $+, \times, -, \exp, \sin, \cos, \lambda x.\frac{1}{x}, \ /, \ \ln, \lambda xy.x^y$ are in H_0, the Kronecker δ function, the signum function and absolute value are in H_1. The Heaviside function Θ, the binary maximum max, the square-wave function s and the floor function are in H_1.

Let us give here the examples of some functions which have important significance in mathematics and can be expressed in terms of real recursiveness.

Example 13. The Bessel functions of the first kind J_v of order v (integer) are real recursive functions of the class H_0.

Let us start with the differential equation: $y''x^2 + y'x + (x^2 - v^2)y = 0$, $v \in Z$. Then this equation has a general solution equal to $AJ_v + BY_v$, where Y_v is the Bessel function of the second kind of order v. We can transform this equation to the form: $y'' = \frac{-xy' - (x^2 - v^2)y}{x^2}$, $x \neq 0$. Now, we are ready to present a quasi-linear differential equation of a type used by Pour-El in her definition of GPAC-computable functions by an introduction of auxiliary variables: $y_1 = y$, $y_2 = y'$, $y_3 = y''$, $y_4 = -(x^2 - v^2)$, $y_5 = y_4 y_1$, $y_6 = -x$, $y_7 = y_6 y_2$, $y_8 = y_7 + y_5$, $y_9 = x^2$, $y_{10} = \frac{1}{y_9}$, $y_{11} = y_{10}^2$ (now $y_3 = y_{10} y_8$).

We have

$$\begin{bmatrix} 1 &&&&&&&&&& \\ & 1 &&&&&&&&& \\ && 1 &&&&&&&& \\ -y_4 && -y_1 & 1 &&&&&&& \\ &&&& 1 &&&&&& \\ & -y_6 &&& -y_2 & 1 &&&&& \\ &&& -1 && 1 & 1 &&&& \\ &&&&&&& 1 &&& \\ &&&&&&& y_{11} & 1 && \\ &&&&&&&& -2y_{10} & 1 & \\ &&&&&& -y_{10} && -y_8 && 1 \end{bmatrix} \begin{bmatrix} y_1' \\ y_2' \\ y_4' \\ y_5' \\ y_6' \\ y_7' \\ y_8' \\ y_9' \\ y_{10}' \\ y_{11}' \\ y_3' \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ -2x \\ 0 \\ -1 \\ 0 \\ 0 \\ 2x \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

By an addition of proper initial conditions: $y(1) = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} \cos(\sin\phi)\cos v\phi \, d\phi$, $y'(1) = -\frac{2}{\pi}\int_0^{\frac{\pi}{2}} \sin(\sin\phi)\sin\phi\cos v\phi \, d\phi$ for v even, $y(1) = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} \sin(\sin\phi)\sin v\phi \, d\phi$, $y'(1) =$

$\frac{2}{\pi}\int_0^{\frac{\pi}{2}} \cos(\sin\phi)\sin\phi\sin v\phi\, d\phi$ for v odd, we get as a solution the function J_v. The above consideration is correct because the initial conditions are real computable, i.e. they can be presented by real recursive functions.

Because the Bessel functions of the first kind are expressed as GPAC-computable function (without any limit operation), hence they are in H_0.

Example 14. The Euler's Γ-function is a real recursive function from the class H_1.

Let us recall that $\Gamma(x) = \int_0^\infty t^{x-1}\exp(-t)\,dt$. It is simple to observe $\Gamma(x) = \lim_{s'\to\infty}\int_0^{s'} s^{x-1}\exp(-s)\,ds$. Because $s^{x-1}\exp(-s)$ is a real recursive function and $\int_0^{s'} s^{x-1}\exp(-s)ds$ is in H_0 hence Γ is in H_1. Let us add that Marion Pour-El (see [17]) proved that Γ is not GPAC-computable so its class is most probably strictly H_1.

Example 15. The Riemann zeta function ζ is a real recursive function from the class H_1.

The following equation stands $\zeta(x) = \frac{1}{\Gamma(x)}\int_0^\infty \frac{t^{x-1}}{1-\exp(-t)}\,dt$, for $x > 0$, where the right-hand side can be defined simply by real recursive operators using the previous results. It is clear from the form of the expression $\frac{1}{\Gamma(x)}\int_0^\infty \frac{t^{x-1}}{1-\exp(-t)}\,dt$ that ζ is also in H_1.

4. η-Function

We gave the general definition of real recursive functions. For proper analysis of functions it is important to control the domain and singularities of functions. We can postulate new operators which may check the points: are they in the domain of some functions or not.

Definition 16. For any function $f: R^{n+1} \to R$ let [5]

$$\eta_y f(\bar{x}, y) = \begin{cases} 1 & \text{if } \lim_{y\to\infty} f(\bar{x}, y) \text{ exists,} \\ 0 & \text{otherwise,} \end{cases}$$

$$\eta_y^i f(\bar{x}, y) = \begin{cases} 1 & \text{if } \liminf_{y\to\infty} f(\bar{x}, y) \text{ exists,} \\ 0 & \text{otherwise,} \end{cases}$$

$$\eta_y^s f(\bar{x}, y) = \begin{cases} 1 & \text{if } \limsup_{y\to\infty} f(\bar{x}, y) \text{ exists,} \\ 0 & \text{otherwise.} \end{cases}$$

Defined in this way $\eta_y f(\bar{x}, y)$ is a characteristic function for the set of such $\bar{x}$ that $\lim_{y\to\infty} f(\bar{x}, y)$ is well defined (without singularities). Analogously $\eta_y^i f(\bar{x}, y)$, $\eta_y^s f(\bar{x}, y)$ play the same role, respectively, for $\liminf_{y\to\infty} f(\bar{x}, y)$, $\limsup_{y\to\infty} f(\bar{x}, y)$. The problem arises whether such operators are real recursive operators. If the answer to the question, whether we can define them by standard operators, is yes, we may patch any partial function to total one. For example, let the function f be total and $F_{\text{total}}(\bar{x}) = \lim_{y\to\infty}(\eta_y f(\bar{x}, y))$

[5] Whenever we say that lim, lim sup, lim inf are defined we want to say that they belong to R.

$f(\bar{x}, y)$, $F(\bar{x}) = \lim_{y\to\infty} f(\bar{x}, y)$. The function $F_{\text{total}}(\bar{x})$ is total and has such a property that if $F(\bar{x})$ is defined, then $F_{\text{total}}(\bar{x}) = F(\bar{x})$. For points which are not in the domain of F we have $F_{\text{total}}(\bar{x}) = 0$.

The key problem in our investigation of the operators η, η^i, η^s is a question: is the class of real recursive functions closed under them. This would be true if functions obtained by these operators from real recursive functions can be constructed as real recursive functions.

Proposition 17. *The functions $\eta_y g$, $\eta^i_y g$, $\eta^s_y g$ are total real recursive functions if g is total real recursive function.*

Proof. Let us start with a construction for $\eta^i_y g(\bar{x}, y)$. If we define $h(\bar{x})$ as $\liminf_{y\to\infty} g(\bar{x}, y)$, then h can be undefined in two cases:
either $\liminf_{y\to\infty} g(\bar{x}, y) = \infty$ or $\liminf_{y\to\infty} g(\bar{x}, y) = -\infty$. But

$$(\liminf_{y\to\infty} g(\bar{x}, y) = -\infty) \vee (\liminf_{y\to\infty} g(\bar{x}, y) = \infty) \equiv \limsup_{y\to\infty} \frac{1}{|g(\bar{x}, y)| + 1} = 0$$

and, moreover, $\limsup_{y\to\infty} \frac{1}{|g(\bar{x},y)|+1}$ is always defined with values between 0 and 1. Finally, we have $\eta^i_y g(\bar{x}, y) = sgn(\limsup_{y\to\infty} \frac{1}{|g(\bar{x},y)|+1})$.

The same method gives us the function $\eta^s_y g(\bar{x}, y)$ when we start with the equivalence $(\limsup_{y\to\infty} g(\bar{x}, y) = +\infty) \vee (\limsup_{y\to\infty} g(\bar{x}, y) = -\infty) \equiv \liminf_{y\to\infty} \frac{1}{|g(\bar{x},y)|+1} = 0$.

Now let us finish by the proper construction of $\eta_y g(\bar{x}, y)$. In this case, two conditions are needed: both $\liminf_{y\to\infty} g(\bar{x}, y)$ and $\limsup_{y\to\infty} g(\bar{x}, y)$ are defined and they are equal.

Let us define

$$h^i(\bar{x}) = \liminf_{y\to\infty} (\eta^i_y g(\bar{x}, y) \cdot g(\bar{x}, y)),$$

$$h^s(\bar{x}) = \limsup_{y\to\infty} (\eta^s_y g(\bar{x}, y) \cdot g(\bar{x}, y)).$$

Then if $\liminf_{y\to\infty} g(\bar{x}, y)$ is defined we have $h^i(\bar{x}) = \liminf_{y\to\infty} g(\bar{x}, y)$ otherwise $h^i(\bar{x}) = 0$. The same property holds for $h^s(\bar{x})$.

It is sufficient to write the following equation to get the final result:

$$\eta_y g(\bar{x}, y) = \eta^s_y g(\bar{x}, y) \cdot \eta^i_y g(\bar{x}, y) \cdot eq(h^i(\bar{x}), h^s(\bar{x})). \qquad \square$$

Now, we can turn to some application of the η operator. We consider a possibility of a process of Turing machines simulation by real recursive functions.

A Turing machine can be given by the following description. It consists of an infinite tape for storing the input, output, and scratch working, and a finite set of internal states. All elements on a tape are strings. Without loss of generality, we can choose some alphabet for these strings, the binary alphabet is a practical choice.

The machine works in steps. In one step it scans the symbol from the current position of the tape (under the head of the machine), changes this symbol according to current state of the machine and moves the position of the tape to left or right with a transformation of state. Some states are distinguished as final, when the machine reaches one of them then it stops. Our Turing machine model obey to the following rules (classical constraints): (a) input is finite and (b) output is finite, no matter the length of computation, being it finite or infinite.

Proposition 18. *There are real recursive functions from the class H_1, which can simulate any Turing machine.*

Proof. Let us recall (see [11]) that we can construct some analytic function, such that a process generated by this function can be considered as a simulation of the activity of a given Turing machine m. This function is of course vector-valued real recursive function $f_M\colon R^2 \to R^2$. As the arguments of such function we take: x encodes the right half of tape and the current state s, and y the left halt of tape. For the Turing machine with n states (hence $0 \le s < n$) and m tape symbols, let us define: $x = s + n\sum_{i=0}^{\infty}(m+1)^i a_i$, $y = \sum_{i=1}^{\infty}(m+1)^{i-1}a_{-i}$, where a_0 is a code of a symbol under the head of the Turing machine, and for all i codes $0 \le a_i < m$ (blank symbol has a code 0). Then with the auxiliary function[6] $h_p(x) = (\frac{\sin \pi x}{p \sin \frac{\pi x}{p}})^2$ and S_{s,a_0} equal to the new state, A_{s,a_0} equal to the printed symbol, Δ_{s,a_0} equal to -1, 1 for moves in the left or the right direction, respectively, or 0 for final states (which by the convention are without any movement) we have:

$$f_M(x,y) = \sum_{s=0}^{n-1}\sum_{a_0=0}^{m-1} \Delta_{s,a_0}^2 h_{(m+1)n}(x - s - na_0) \times \left(\frac{1+\Delta_{s,a_0}}{2}(x_r, y_r) + \frac{1-\Delta_{s,a_0}}{2}\sum_{a_{-1}=0}^{m-1} h_{m+1}(y - a_{-1})(x_l, y_l)\right),$$

where

$$(x_r, y_r) = \left(S_{s,a_0} + \frac{x - s - na_0}{m+1}, (m+1)y + A_{s,a_0}\right),$$

$$(x_l, y_l) = \left(S_{s,a_0} + (m+1)(x - s + n(A_{s,a_0} - a_0)) + na_{-1}, \frac{y - a_{-1}}{m+1}\right).$$

Such f_M is in the class H_0. The iteration f_M^n can be given in the following method, $f, g\colon R \to R^2$:

$$f(0) = g(0) = (x, y),$$

$$2\cos^2\left(\frac{\pi t}{2}\right)\partial_t g(t) = (f_M(f(t)) - f(t))\Theta_2(\sin \pi t),$$

$$\sin^2(\pi t) r(t) \partial_t f(t) = \sigma(g(t) - f(t))\Theta_2(-\sin \pi t),$$

[6] This form of a definition of h_p is only a notion for analytic functions, which can be obtained by simple trigonometric transformations.

where $\Theta_2(x) = x^2\Theta(x)$, $\sigma = \int_0^1 \Theta_2(\sin \pi x)\, dx$, $r(0) = 0$, $\partial_x r(x) = 2\Theta_2(\sin \pi x) - \sigma$. The above equations are vectorial equations: in fact we have here a system of 4 equations performing iteration of vector f_M.

The nth iteration of the function f_M satisfies the following equation: $f_M^n(x, y) = f(2n) = g(2n)$ for natural n. It can be explained in the following way: as t changes from 0 to 1 then f is constant and g goes through the distance from (x, y) to $f_M(x, y)$. For $t \in [1, 2]$ the function g is fixed and f catches up, hence $f(2) = g(2) = f_M(x, y)$. If $t > 2$ then the same cycle begins again.

Because $\Theta_2 \in H_1$ hence the vector f_M^n is in the class H_1, however, let us mention that Θ_2 is continuous with its derivative. □

Let us mention, that we can modify f_M^n to have the same value in the next step, if the state in the previous one was a final one. Then by use of an infinite limit on f_M^n we get the output function of M in H_2.

It can be mentioned that the process of simulation is especially important for universal Turing machines. The results in this area proved in last years [19] give us the interesting restrictions of the size of such machines (for example, there exists a universal Turing machine for 5 states and 5 symbols) what leads us to significant simplicity of the constructed function. It is worth to point out that defined in this form f_M is analytical (see [11]). This fact has as a consequence a lower level of the complexity of the simulation.

Let us signal a few important questions concerned to Turing machines. The first problem is known as the halting problem: does the machine M for input (x, y) reach the final state? There is not a natural recursive characteristic function of this problem. The method of simulation of Turing machines given above can resolve it in the simple way with real recursive functions.

Proposition 19. *For any Turing machine M, there exists a real recursive function which is the characteristic function of the halting problem for M.*

Proof. We can define $F'_M(x, y, z) = f_M^{\lfloor z \rfloor}(x, y)$, then let

$$H_M(x, y) = (\eta_z F'_M(x, y, z))H(\lim_{z\to\infty}(\eta_z F'_M(x, y, z)) \cdot F'_M(x, y, z)),$$

where $H(x, y) = 1$ if the state written in x is final, 0 otherwise. The function $H(x, y)$ can be defined as $\sum_{s\in F}\sum_{a=0}^{m-1} h_{(m+1)n}(x - s - na)$. The function H_M is a real recursive characteristic function of the halting problem for the machine M. □

To obtain the function computed by M, it is enough to iterate the steps up to the reaching of the final state by the machine. If the machine M ends in the final state for some tape (x, y), then there exists such $n_0 \in N$ that the sequence $f_M^n(x, y)$ is constant for $n \geq n_0$. We can define the function $F_M(x, y)$ computable by M as

$$F_M(x, y) = \lim_{z\to\infty}\left[f_M^{\lfloor z \rfloor}(x, y) \times g(H(\lim_{z\to\infty}(\eta_z F'_M(x, y, z)) \cdot F'_M(x, y, z)))\right],$$

where g is a function not defined at 0, otherwise it takes value 1 (for example given as $\lim_{y\to\infty} \frac{1}{1-\exp(-|x|y)}$). Then F_M is defined whenever lim exists and the value of the function H is 1 (i.e. the Turing machine M reaches for the initial tape (x, y) a final state), otherwise is undefined.

Let us turn more deeply into the problems of computation beyond the power of Turing machines (hypercomputation). The problem of infinity which can appear in the sequel of not finishing computation introduced troubles into the computability theory and practice. The first step to improve this situation is directed to change the behaviour of a Turing machine. For this purpose we may use an accelerated Turing machine [6]. Its description is the same as for a standard Turing machine, but a temporal pattern of steps is given. Each subsequent step is performed in half the time of the step before. Such machines could complete an infinity of steps in two time units only. This feature of accelerated Turing machines gives us the power to puzzle out the halting problem by programming the following algorithm: mark the first square on the tape by 0, change it only in the final (last) step to 1, if after 2 time units we have 0 in the distinguished square, then machine does not halt, otherwise it halts. However, some difficulties arise also in this model. Let us imagine the machine changing value of one square from 1 to 0 and conversely in all steps using only one non-final internal state. We can hesitate what is on the tape after all steps (in infinity), because in this case the computation diverges. The accelerated Turing machine can be simulated in the same way as the standard Turing machine with only one modification: in the definition of $F_M(x, y)$ it is not necessary to have the result (z_x, z_y) with a final state i written in z_x. Hence, the convergent infinite computations and finite computations both give the correct result, however the divergent computations have undefined result.

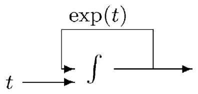
The above remarks prove that η operator gives us the additional power to standard models of computation by controlling the domain of computable functions and machines. Such possibility is an effect of checking in a finite amount of time an infinite number of a computation elements. The standard objection to such extensions of computable systems is their unphysical character. That in the limit of physical reality models would not exhibit super-Turing capabilities is believed since the beginning of Computer Science. Penrose in [16] stresses this fact before he talks about the (non-computable) ultimate physical theory to come and the human mind: *Now, where do we stand with regard to computability in classical theory? It is reasonable to guess that, with general relativity, the situation is not significantly different from that of special relativity—over and above the differences in causality and determinism that I have just been presenting. Where the future behaviour of the physical system is determined from initial data, then this future behaviour would seem (by similar reasoning to that I presented in the case of Newtonian theory) also to be computably determined by that data (apart from unhelpful type of non-computability encountered by Pour-El and Richards for the wave equation, as considered above—and which does not occur for smoothly varying data). Indeed, it is hard to see that in any of the physical theories that I have been discussing so far there can be any significant "non-computable" elements. It is certainly to be expected that "chaotic" behaviour can occur in many of these theories, where very slight changes in initial data can give rise to enormous differences in resulting behaviour. But, as I mentioned before, it is hard to see how this type of non-computability—i.e. "unpredictability"—could be of any "use" in a device which tries to "harness" possible non-computable elements in physical laws.*

Theory of n-body dynamics and general relativity may provide counterarguments to Penrose statement. In fact, we know that some results for Newtonian physics [24] or general relativity [9] may be used to harness devices more powerful than a standard Turing machine.

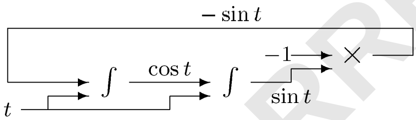
5. Comparison with analog computers

We start with some considerations connected with the GPAC. GPAC is a model of analog computation introduced by Shannon (indeed a student of Vannevar Bush): this fact gives us a strong motivation for theoretical development. The proof we recall below of the relationship of GPAC-computability and the recursive functions over the reals gives us also a strong basis for further research on Cris Moore framework. Moreover, GPAC is a model of a real computer, designed by Bush, i.e., GPAC is physically realizable in a strong sense: integrators are physical devices built since the 19th century. To better understanding of this notion let see at the beginning of the section the example of the definition of a GPAC-computable function.

Example 20. The exponential circuit is given on the below picture (with the initial condition $\exp(0) = 1$).



Let us recall now the vector $(\sin x, \cos x)$. We can present the construction of these functions by the following scheme of units:



Its initial conditions are $\sin(0) = 0$ and $\cos(0) = 1$. The output w of the integrator unit $\int$ obeys $dw = u\,dv$ where u and v are its upper and lower inputs, respectively.

The first example shows local feedback characteristic of a linear system.

We introduce now a further concept. By an analogy with the recursive functions of Kleene, whenever a function is defined only with composition and differential recursion ($f \in H_0$), we call f a primitive real recursive function. [7]

Proposition 21. *Every primitive real recursive function f defined on the closed domain $D \in R^n$ is GPAC-computable function.*

Proof. The constants $-1, 0, 1$ are clearly GPAC-computable. The primitive real recursive functions are defined by compositions and differential recursion. We have to show that

[7] There is a slight difference since (classical) primitive recursive functions are always total and primitive real recursive functions can be partial.

GPAC-computability is preserved by these two operators. It is obvious for a composition. For a differential recursion, we can observe that the function f defined by it is on D bounded with its derivative. Theorem 9 from [8] states that such function f with the mentioned properties is GPAC-computable. □

However, let us point out that there are functions (like $\lambda x.|x|$ in the interval $[-1, 1]$), which are bounded with their derivatives but they or some of their derivatives are not continuous.

Theorem 22. *Every GPAC-computable function with real recursive numbers as parameters is real recursive function.*

Proof. It is sufficient to use the Theorem 8 from [8]. It states that the class of GPAC-computable functions is identical to such minimal class of functions, which contains $-1, 0, 1$ and is closed under composition and integration with added restriction that a defined function and its derivatives are bounded. Let us assume that all the constant units of the GPAC are associated to real recursive numbers. Then, of course, this class is embodied in the class of real functions (our form of a differential recursion generates wider set of functions than a integration with restriction that a defined function and its derivatives are bounded used in [8]). □

Now, we can give the first account that real recursive functions include such functions, which are not GPAC-computable.

Proposition 23. *The class of real recursive functions is a proper superset of the class of GPAC-computable functions.*

Proof. The above lemma is obvious from our result that Γ Euler function and ζ Riemann function are real recursive functions and from the result of Marion Pour-El [17] that these functions are not GPAC-computable. □

6. Hierarchies: arithmetical and analytical

We will proceed now with the relations of natural numbers taken from the arithmetical hierarchy. The class $\Sigma_0^0 = \Pi_0^0$ contains only such relations, which have recursive characteristic functions. The upper stages of this hierarchy can be constructed from the lower ones in the following way:

$$\Sigma_{n+1}^0 = \{P : (\exists P' \in \Pi_n^0) P(\bar{m}) \equiv \exists s\, P'(\bar{m}, s)\},$$

$$\Pi_{n+1}^0 = \{P : (\exists P' \in \Sigma_n^0) P(\bar{m}) \equiv \forall s\, P'(\bar{m}, s)\},$$

where $P \subseteq N^k$, $P' \subseteq N^{k+1}$, $k \geq 1$. To complete our hierarchies we can add the following equation $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$, $n \geq 0$.

Now, let us correlate this infinite hierarchy of sets and relations to the η-hierarchy. We must return to the Turing machine and its simulation by real recursive functions.

From Proposition 18 and from the fact that all natural recursive sets and relations have Turing computable total characteristics we get the following conclusion:

Corollary 24. *Every recursive set or relation (with argument from N) is in H_2, i.e.* $\Sigma_0^0 = \Pi_0^0 \subset H_2$.

The next element of our investigation has to deal with higher levels of arithmetical hierarchy. For this purpose, we need to analyse the method of use of quantifiers.

For every function $f: R^{n+1} \to R$, we can construct such real recursive function $\rho_f : R^n \to R$ that

$$\rho_f(\bar{x}) = \begin{cases} 1 & \exists y \in N f(\bar{x}, y) = 0, \\ 0 & \forall y \in N f(\bar{x}, y) \neq 0. \end{cases}$$

To this effect we start with a description of the function $f_c(\bar{x}, y) = 1 - \delta(f(\bar{x}, y))$. This function has the following property $f_c(\bar{x}, y) = 1 \equiv f(\bar{x}, y) \neq 0$, $f_c(\bar{x}, y) = 0 \equiv f(\bar{x}, y) = 0$. It is easy to observe that now

$$\lim_{z\to\infty} \prod_{j=0}^{z} f_c(\bar{x}, j) = \begin{cases} 0 & \exists y \in N f(\bar{x}, y) = 0, \\ 1 & \forall y \in N f(\bar{x}, y) \neq 0. \end{cases}$$

Hence $\rho_f(\bar{x}) = 1 - \lim_{z\to\infty} \prod_{j=0}^{\lfloor z \rfloor} f_c(\bar{x}, j)$. We should indicate two points. The first, real recursive functions are closed under the product operation. It can be defined as an iteration of the function $t_f : R^{n+2} \to R^{n+2}$, $t_f(\bar{x}, y, i) = (\bar{x}, y f(\bar{x}, i), i + 1)$ hence

$$\prod_{i=0}^{n} f(\bar{x}, i) = I_3^2(t_f^n(\bar{x}, 1, 0)).$$

The second, let us analyse the stage of η-hierarchy, which contains $\rho_f(\bar{x})$ if $f \in H_i$, where $i \in N$ is a given number. The function f_c is in H_{i+1} and consequently by properties of an iteration $\prod_{j=0}^{n} f_c(\bar{x}, j) \in H_{i+1}$. Finally, we can claim that $\rho_f \in H_{i+2}$.

Theorem 25. *The sets and relations from* Σ_i^0, Π_i^0 *belong to* H_{i+2} *for* $i \geq 0$.

Proof. It is clear from the above considerations that if a relation $R \in N^{k+1}$ is in H_i, $i \geq 0$ (which, of course, means its characteristic function $\chi_R \in H_i$), then the relation $P(\bar{x}) = \exists y R(\bar{x}, y)$ has the characteristic function χ_P equal to $\rho_{1-\chi_R}$. Because the natural recursive relations are in H_2, so the function χ_R is at least in H_2. Moreover, a normalization of the function f to two values 0, 1 realized by f_c is not needed in the case of $f = \chi_R$. Hence the relation P is in H_{i+1}. For relation $Q(\bar{x}) = \forall y R(\bar{x}, y)$ it is sufficient to observe that the characteristic function $\chi_Q(\bar{x}) = 1 - \chi_{\exists\neg R}(\bar{x})$ belongs to H_{i+1} too. Using the above results as an inductive step with an additional assumption that natural recursive relations are in H_2 we obtain the thesis of this theorem. □

Let us analyse only one aspect of the analytical hierarchy yet. We can deal with especially important class Π_1^1. The class of Π_1^1 relations is defined by a function quantifier used on an arithmetical relation: $R \in N^{k+1}$ is Π^1_1 if $R(\bar{x}, y) \equiv \forall f : N \to N\, Q(\bar{x}, f(y))$, where Q from some level of the arithmetical hierarchy.

Proposition 26. *The relation $R \in \Pi^1_1$ is in H_6.*

Proof. We use the result from [10], which states that R is Π^1_1 iff the following condition holds:

$$R(\bar{x}) = (\exists f : N \to N)(\forall y \in N) P(\bar{x}, \hat{f}(y)),$$

where $\hat{f}(y) = \langle f(0), \ldots, f(y)\rangle$ is a course-of-value function, $\langle \ldots \rangle$—a recursive coding of natural numbers, P—a recursive relation.

We can change the above formula to the equivalent form. For $f(y) = z \equiv R_f(y, z)$ and $\hat{f}(y) = \langle z_0, \ldots, z_y\rangle$ we have

$$R_{\hat{f}}(y, z) \equiv (\forall i \le y) R_f(i, z_i) \wedge z = \langle z_0, \ldots, z_y\rangle.$$

With these relations we obtain

$$\begin{aligned} R(\bar{x}) &\equiv (\exists R_f \subset N^2)(\forall y \in N) R_{\hat{f}}(y, z) \wedge P(\bar{x}, z) \\ &\equiv (\exists R_f \subset N^2)(\forall y \in N)(\forall i \le y) R_f(i, z_i) \wedge z = \langle z_0, \ldots, z_y\rangle \wedge P(\bar{x}, z). \end{aligned}$$

Every relation $Q \subset N^2$ can be coded into real number a_Q from $[0, 1)$, in such a method that $\langle x, y\rangle$th cipher in a binary expansion of a_Q is equal to 1 iff $Q(x, y)$, otherwise is equal to 0. We omit the possibility $a_Q = 1$ because it can hold only for trivial always satisfied relations. Then we can write the following equivalence:

$$\begin{aligned} R(\bar{x}) &\equiv (\exists a \in [0, 1))(\forall y \in N)(\forall i \le y) a[\langle i, z_i\rangle] = 1, \\ \wedge\, z &= \langle z_0, \ldots, z_y\rangle \wedge K_P(\bar{x}, z) = 1, \end{aligned}$$

K_P is a characteristic function of P. Let us abbreviate the innermost expression $a[\langle i, z_i\rangle] = 1 \wedge z = \langle z_0, \ldots, z_y\rangle \wedge K_P(\bar{x}, z) = 1$ as the function $\zeta(a, i, \bar{x})$ which gives the value 1 iff all elements of the conjunction are satisfied, 0 otherwise. This function is build from natural recursive functions without limits, hence $\zeta \in H_2$. The result of this consideration is a formula: $R(\bar{x}) \equiv (\exists a \in [0, 1))(\forall y \in N)(\forall i \le y)\zeta(a, i, \bar{x}) = 1$. Because the bounded quantifier can be modeled by an iteration we can transform the above equation into the below $R(\bar{x}) \equiv (\exists a \in [0, 1))(\forall y \in N)\zeta'(a, y, \bar{x}) = 1$, $\zeta'(a, y, \bar{x})$ is a realisation of $(\forall i \le y)\zeta(a, i, \bar{x})$, $\zeta' \in H_2$. The universal quantifier can be coded with only one limit (see the previous theorem), hence the relation of $R(\bar{x})$ is equivalent to the condition $\exists a \in [0, 1)\zeta''(a, \bar{x}) = 1$, where $\zeta'' \in H_3$.

Let us define the extension ζ^+ of a function ζ'' on all positive reals in the following way: $\zeta^+(y, \bar{x}) = \zeta''(y - \lfloor y \rfloor, \bar{x})$. Then $\exists a \in [0, 1)\zeta''(a, \bar{x}) = 1$ is equivalent to the fact that $\limsup_{y\to\infty} \zeta^+(y, \bar{x}) = 1$, if $\forall a \in [0, 1)\zeta''(a, \bar{x}) = 0$, then $\limsup_{y\to\infty} \zeta^+(y, \bar{x}) = 0$. Hence, the characteristic function of every relation $R \in \Pi^1_1$ is equal to $\limsup_{y\to\infty} \zeta^+(y, \bar{x})$ and belongs to H_6. □

The levels of the analytical hierarchy and their relation to the η-hierarchy can be analysed with the μ-operator like in [13]. Because the μ-operator may be replaced by infinite limits (see [14]) the remaining part of the analytical hierarchy can be obtained in this way.

7. Conclusions

In the final remarks of his paper [13], Moore consider the possibility of taking limits and questioned himself if the hierarchy of real recursive functions would be quite the same. One of the authors tried to prove the equivalence between the taking of limits and the use of minimalization. In [14], he presents the proof that minimalization can be expressed in terms of infinite limits.

The fact that limits and differential recursion are interchangeable is obvious since the exponential function can be seen either as solution of $\partial_y f(y) = I_2^2(y, f(y))$, with initial condition $f(0) = 1$, or as $\lim_{y\to\infty}(1 + x/y)^y$. A more general problem can be stated in the following way: given the scheme of differential recursion

$$h_i(x_1, \ldots, x_k, 0) = f_i(x_1, \ldots, x_k),$$

$$\partial_y h_i(x_1, \ldots, x_k, y) = g_i(x_1, \ldots, x_k, y, h_1(x_1, \ldots, x_k, y), \ldots, h_n(x_1, \ldots, x_k, y)),$$

for $1 \le i \le n$ and defining $\psi_i(x_1, \ldots, x_k) = \lim_{y\to\infty} h_i(x_1, \ldots, x_k, y)$, to find sufficient conditions on f_i and g_i such that ψ_i is definable by the same scheme of differential recursion on some variable x_i, $1 \le i \le k$, but with no limits in the definition. We would like to find the decidable procedure to identify descriptions with such a property that they can be *reduced* to differential recursion.

In a second step, we foresee that a fragment of the language can be made to coincide with the class of functions computable by the Rubel's EAC. This task will probably be easier than the first, although we know in advance that we have to make some changes in the set of our basic operators in order to deal with the basic components of the EAC such like the *inverters*.

The third task we further envisage is to inspect the realization of some enlarged class of defined functions in the limits of physical reality. We have reasons to believe that the n-body dynamics has hypercomputation capabilities and we would like to explore them. Xia's paper [24] showing that an infinite number of mechanical events can happen in finite time opens a way of thought. We are not aware of anyone who has tried to translate the halting problem into the n-body problem in classical mechanics: we have to show that the subset of initial data that go off to infinity in finite time codes a universal machine. In Tipler's book [23], he conjectured that universal initial data exists, and, as far as we know, it seems that the universal initial data is of measure zero in the space of all initial data.

The natural counterpart of the η-hierarchy in mathematical analysis is the hierarchy of Baire classes. It would be important to find the relation between these two hierarchies, especially to demonstrate the non-collapsing character of the η-hierarchy. In the forthcoming paper we hope to present some results in this direction.

Acknowledgements

We thank to Cris Moore (and the very cat Spootie) for suggesting the project that gave rise to so much research and now to this ultimate paper on his original framework. José Félix Costa thank his students Manuel Campagnolo and Daniel Graça for all the mathematical fights during the last years; he, who performed so well the "Black Knight", realizes now that so much emerged from a few mistakes in a paper.

References

[1] L. Blum, F. Cucker, M. Shub, S. Smale, Complexity and Real Computation, Springer, Berlin, 1998.
[2] P. Boldi, S. Vigna, Equality is a jump, Theoret. Comput. Sci. 219 (1–2) (1999) 49–64.
[3] M.L. Campagnolo, The complexity of real recursive functions, C.S. Calude, M.J. Dinneen, F. Peper(Eds.), Unconventional Models of Computation, UMC 2002, Lecture Notes in Computer Science, Vol. 2509, Springer, Berlin, 2002, pp. 1–14.
[4] M.L. Campagnolo, C. Moore, J.F. Costa, Iteration, inequalities, and differentiability in analog computers, J. Complexity 16 (4) (2000) 642–660.
[5] M.L. Campagnolo, C. Moore, J.F. Costa, An analog characterization of the Grzegorczyk hierarchy, J. Complexity 18 (4) (2002) 977–1000.
[6] B.J. Copeland, Even Turing machines can compute uncomputable functions, in: C.S. Calude, J. Casti, M.J. Dinneen (Eds.), Unconventional Models of Computation, Springer, Berlin, 1998.
[7] Encyclopaedia of Mathematics, Upper and lower bounds, Kluwer Academic Publishers, Dordrecht, 1993.
[8] D. Graça, J.F. Costa, Analog computers and recursive functions over the reals, J. Complexity 19 (2003) 644–664.
[9] G. Etesi, I. Nemeti, Non-Turing computations via Malament–Hogarth space-times, Internat. J. Theoret. Phys. 41 (2002) 341–370.
[10] S.K. Kleene, Arithmetical predicates and quantifiers, Trans. Amer. Math. Soc. 79 (1955) 312–340.
[11] P. Koiran, C. Moore, Closed-form analytic maps in one and two dimensions can simulate universal Turing machines, Theoret. Comput. Sci. 210 (1) (1999) 217–223.
[12] L. Lipshitz, L.A. Rubel, A differentially algebraic replacement theorem, and analog computation, Proc. Amer. Math. Soc. 99 (2) (1987) 367–372.
[13] C. Moore, Recursion theory on the reals and continuous-time computation, Theoret. Comput. Sci. 162 (1996) 23–44.
[14] J. Mycka, μ-recursion and infinite limits, Theoret. Comput. Sci. 302 (2003) 123–133.
[15] P. Odifreddi, Classical Recursion Theory, North-Holland, Amsterdam, 1989.
[16] R. Penrose, The Emperor's New Mind, Oxford University Press, Oxford, 1989.
[17] M.B. Pour-El, Abstract computability and its relations to the general purpose analog computer, Trans. Amer. Math. Soc. 199 (1974) 1–28.
[18] S. Puchta, On the role of mathematics and mathematical knowledge in the invention of Vannevar Bush's early analog computers, IEEE Ann. History Comput. 18 (4) (1996) 49–59.
[19] Y. Rogozhin, Small universal Turing machines. Universal machines and computations, Theoret. Comput. Sci. 168 (2) (1996) 215–240.
[20] L.A. Rubel, Some mathematical limitations of the general-purpose analog computer, Adv. Appl. Math. 9 (1988) 22–34.
[21] L.A. Rubel, The extended analog computer, Adv. Appl. Math. 14 (1993) 39–50.
[22] C. Shannon, Mathematical theory of the differential analyzer, J. Math. Phys. MIT 20 (1941) 337–354.
[23] F. Tipler, The Physics of Immortality: Modern Cosmology, God and the Resurrection of the Dead, Anchor, 1997.
[24] Z. Xia, The existence of noncollision singularities in Newtonian systems, Ann. Math. 135 (3) (1992) 411–468.