Computability on Reals, Infinite Limits and Differential Equations

Bruno Loff [a,b,*], José Félix Costa [a,b], Jerzy Mycka [c]

[a] *Instituto Superior Técnico*
Technical University of Lisbon
Lisbon, Portugal
[b] *Centro de Matemática e Aplicações Fundamentais do Complexo Interdisciplinar*
University of Lisbon
Lisbon, Portugal
[c] *Institute of Mathematics*
University of Maria Curie-Sklodowska
Lublin, Poland

Abstract

We study a countable class of real-valued functions inductively defined from a basic set of trivial functions by composition, solving first-order differential equations and the taking of infinite limits. This class is the analytical counterpart of Kleene's partial recursive functions. By counting the number of nested limits required to define a function, this class can be stratified by a potentially infinite hierarchy — a hierarchy of infinite limits. In the first meaningful level of the hierarchy we have the extensions of classical primitive recursive functions. In the next level we find partial recursive functions, and in the following level we find the solution to the halting problem.

We use methods from numerical analysis to show that the hierarchy does not collapse, concluding that the taking of infinite limits can always produce new functions from functions in the previous levels of the hierarchy.

Key words: Real Recursive Functions, Infinite Limits, Differential Equations, Computability on Reals

Contents

* Corresponding author. Address is R. Vitorino Nemésio, N^o 8, 4^o Esq, 1750-307, Lisboa, Portugal.
Email addresses: bruno.loff@gmail.com (Bruno Loff), fgc@math.ist.utl.pt (José Félix Costa), Jerzy.Mycka@umcs.lublin.pl (Jerzy Mycka).

1. Introduction

For most of its existence computer science was seen as a rogue branch of mathematics. Nonetheless several of its results had a profound effect on the way people think about mathematics, and on what people have come to expect of mathematics. Yet there is often a psychological distinction between doing *computer science* and doing *real* or *pure mathematics*, as if computer science was really about computing machines and so it is not actually mathematics, which is about numbers, sets, integrals and so on. This fictitious gap is constantly being bridged by the discovered connections between areas such as mathematical logic and set theory, recursive functions and arithmetics, computability and Diophantine equations and code analysis and statistics. An area that seems to have remained largely unconnected with computer science is mathematical analysis. Since its early days both computer science and computer engineering made a right turn towards the discrete. During the first half of the century there was a big argument on whether to adopt a discrete or a continuous model of computation. Strong support over the discrete model, by people such as von Neumann, as well as the invention of the transistor and the construction of a powerful and solid theory of computability over natural numbers all contributed to the favoring of binary computers over their analog counterparts. Until the late fifties this was not an obvious choice, given the success that analog computers such as Vannevar Bush's differential analyzer had during the war. Connections between computer science and mathematical analysis appear only recently, as there is a resurgent interest in the study of analog computation theory.

The differential analyzer [1] was formalized by Claude Shannon in [2], and since then plenty of work has been done. In the late eighties Blum, Shub and Smale constructed a model to study computation over the reals [3]. Throughout the nineties Hava Sieglemann and Eduardo Sontag have studied the computational power of analog recurrent neural networks [4]. In 1996 Cris Moore published a seminal paper, *Recursive theory on the reals and continuous-time computation* [5], where he defines an inductive class of vector valued functions over $\mathbb{R}$. He shows several interesting results: many well-known functions of analysis are in this enumerable class, and several uncomputable functions are there also. Some work was done since then, using this definition [6, 7, 8, 9], but unfortunately some of Moore's assumptions were not very consensual among people interested in the field. Most of these controversial assumptions were consequences of Moore's attempt to bring the minimalization operator — used in the classical recursive functions — into a continuous context. So in 2002 two of us gave a similar definition of Moore's inductive class of functions, replacing minimalization with the taking of infinite limits. This solved the issues brought about by minimalization, and strengthened the bond between this model of real computation and analysis, since infinite limits are a classical subject of the later. We strengthen the bond further in this paper, by showing relevant results for both computer science — more specifically real computation — and mathematical analysis, using methods from both fields.

The class of real recursive functions is defined inductively in a similar manner to Kleene's recursive functions. The basic functions are 1, -1, 0 and the projections, and the class is the closure of these basic functions for composition, solving of first order differential equations and taking of infinite limits. One defines a hierarchy, called the η-hierarchy, based on the minimum number of nested infinite limits required to describe some real recursive function. From a computational perspective, the η-hierarchy measures how many computational steps are required to compute some function or number. If one has access to some machine that can perform ω computational steps,[1] then one can solve the halting problem for Turing machines, and if the machine can compute ω steps ω times, or simply ω^2 steps, then one can decide the totality of Turing-computable functions. Very similarly one can solve the halting problem using one infinite limit and decide the totality of Turing-computable functions using two nested infinite limits. In this sense the number of infinite limits required to define some function gives a very natural way of ascertaining its noncomputability. The η-hierarchy may also be relevant to physics. Should there be, in the physical world, certain processes that generate an infinite amount of events in a finite amount of time (see [12, 13]), they may be elegantly modeled using real recursive functions. From a mathematical perspective the η-hierarchy can tell us how complex a function is. It is interesting to notice that most, if not all, of the functions studied in analysis are in the first two or three levels of the hierarchy. Basic functions of analysis, like $+$,

[1] Computation over an infinite number of steps has been studied; see [10] for an example, and [11] for an overview.

exp and sin are all in the first level of the hierarchy, but only in the next level can we find the Gauss Γ function and Riemann's ζ function. In the third level we find the everywhere continuous and nowhere differentiable Weierstraß function, and several other pathological functions. We can only speculate about interesting properties of functions that lie in even higher levels.

Real recursive functions were introduced as a tool for studying analog computation. Their original purpose is to provide a theoretical equivalent to Kleene's recursive functions, which are used to study classical computability. Nevertheless, of the variety of functions studied in mathematical analysis one is still to be found by the authors that is not real recursive. It is thus reasonable to expect that some results derived from studying this class are of interest to mathematical analysis. We believe we present one such result in this paper.

The class of real recursive functions we will soon describe is closed under operations of finding solutions to first order differential equations and the taking of infinite limits. Thinking briefly about the relationship between these two operations one may observe that they seem to be related. For instance, $\mathrm{e}^x = \lim_{y\to\infty}\left(1+\frac{x}{y}\right)^y$, and also $\mathrm{e}^x = y(x)$ where $y(0)=1$ and $\frac{\mathrm{d}y}{\mathrm{d}x} = y$. The number π can be expressed by a differential equation that gives arctan, since $\pi = 4\arctan(1)$, and we also know, e.g., that $\pi = \lim_{y\to\infty}\frac{2^{4y+1}y!^4}{(2y+1)(2y)!^2}$. The field of topology gives us another example of this phenomenon, now regarding distributions: if t is a function of arity 2 such that $t(x,h)$ results, for each h, in the plot of a triangle with area 1 and height h, then Dirac's delta function, which will be denoted by $\mathfrak{d}$, can be given by $\mathfrak{d}(x) = \lim_{h\to\infty} t(x,h)$; we also know that $\int_0^x \mathfrak{d}(x)\mathrm{d}x = \Theta(x)$, where Θ is the Heaviside function. These and other examples may lead us to wonder if this is a universal phenomenon. We will use the toolbox of computer science to show that while we can always express the solution of a first order differential equation through infinite limits, we cannot always do the opposite.

The paper hereby presented continues the work of [14] and [15]: in this paper we show that the η-hierarchy does not collapse, an open problem since the beginning of real recursive function theory. Every result is new unless specifically stated. We start by defining our inductive class of functions and discussing some fundamentals of real recursion theory in section 2. In section 3 we define the η-hierarchy, which is based on the number of nesting limits required to describe a function in our class. We proceed in section 4 by studying integration and differentiation in the context of this class of functions. We show that any number of differential equations, relating functions in our inductive class, can be solved by exploring the asymptotical behavior of a numerical approximation. We then show that there is no universal real recursive function Ψ, in section 5. But in section 6 we show that if we restrict the η-hierarchy to any arbitrary level n, a universal function for that level of the hierarchy, Ψ_n, exists. We conclude by showing that the η-hierarchy does not collapse, which is the main result of this paper.

2. Basic definitions and fundamental results

Definition 2.1 *The* class of real recursive vector functions, $\mathrm{REC}(\mathbb{R})$, *is the smallest class of real-valued vector functions containing, for all n, the scalar atoms $\bar{1}^n \equiv \lambda x_1 \ldots x_n.-1$, $1^n \equiv \lambda x_1 \ldots x_n.1$, $0^n \equiv \lambda x_1 \ldots x_n.0$,*[2] *and, for $1 \leqslant i \leqslant n$, $I_i^n \equiv \lambda x_1 \ldots x_n.x_i$, and closed under the operations of:*

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 h, with n m-ary components, given by:*

$$h_i(x_1,\ldots,x_m) = f_i(g_1(x_1,\ldots,x_m),\ldots,g_k(x_1,\ldots,x_m)),$$

for $1 \leq i \leq n$, is real recursive.

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 $(k+1+n)$-ary components, then consider the vector function h of n $(k+1)$-ary components which is the solution of the initial value problem:*

[2] Note that we could obtain the n-ary 1^n, $\bar{1}^n$ and 0^n from their zero-ary equivalents, but we use the n-ary functions for simplicity and convenience.

$$h_i(x_1, \ldots, x_k, 0) = f_i(x_1, \ldots, x_k),$$
$$\partial_y h_i(x_1, \ldots, x_k, y) = g_i(x_1, \ldots, x_k, y, h_1(x_1, \ldots, x_k, y), \ldots, h_n(x_1, \ldots, x_k, y)),$$

for $1 \leq i \leq n$. h is real recursive whenever for every tuple $x_1, \ldots, x_k$:[3]

(i) The function given by $\lambda y.h(x_1, \ldots, x_k, y)$ is continuous and defined on the largest open interval in which a unique continuous solution exists almost everywhere.

(ii) The function given by $\lambda y.g(x_1, \ldots, x_k, y, h(x_1, \ldots, x_k, y))$ is defined in this interval.

Infinite Limits *If f is a real recursive vector function with n $(k+1)$-ary components, then the vector functions $h, h^\dagger, h^{\dagger\dagger}$ with n k-ary components, given by:*

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

for $1 \leq i \leq n$, are real recursive. [4]

Aggregation *Arbitrary real recursive vector functions can be defined by assembling scalar real recursive functions.*

This class is as in [14] except for differential recursion. Here we only require that the function obtained by differential recursion is continuous, rather than having to be in C^1. The equation must be satisfied almost everywhere on the mentioned interval, and g has no required behaviour on the remaining points. The class of real recursive vector functions is enumerable and contains, according to our experimentation so far, all functions and numbers which were not specifically designed not to be in it, including several extensions of Turing-uncomputable functions and numbers. It is, in this sense, a very natural class of functions to do analysis in and as we shall soon see an equally natural measure of non-computability will arise from it.

No study of computability could proceed reasonably without a mechanism of self-reference, so we create one now.

Definition 2.2 *The set of* descriptions of real recursive functions *is inductively defined as follows:*

- $\langle I_j^n \rangle$ *is an n-ary description of I_j^n.*
- $\langle 1^n \rangle$ *is an n-ary description of 1^n.*
- $\langle \bar{1}^n \rangle$ *is an n-ary description of $\bar{1}^n$.*
- $\langle 0^n \rangle$ *is an n-ary description of 0^n.*
- *If $\langle h \rangle$ is a k-ary description of the real recursive vector function h and $\langle g \rangle$ is a description of the real recursive vector function g with k components, then $\langle c(h, g) \rangle$ is a n-ary description of the real recursive vector function obtained by composition of h and g.*
- *If $\langle f \rangle$ is a k-ary description of the real recursive vector function f and $\langle g \rangle$ is a $(k+1+n)$-ary description of the real recursive vector function g, then $\langle dr(f, g) \rangle$ is a $(k+1)$-ary description of the real recursive vector function defined by differential recursion of f and g.*
- *If $\langle h \rangle$ is a $(n+1)$-ary description of the real recursive function h, then $\langle l(h) \rangle$, $\langle li(h) \rangle$, $\langle ls(h) \rangle$ are n-ary descriptions of the real recursive vector functions obtained by the appropriate infinite limit — respectively* $\lim, \liminf$ *and* $\limsup$ *— of h.*
- *If $\langle f_1 \rangle, \ldots, \langle f_m \rangle$ are descriptions of real recursive n-ary scalar functions $f_1, \ldots, f_m$, then $\langle v(f_1, \ldots, f_m) \rangle$ is an n-ary description of the real recursive vector function $f = (f_1, \ldots, f_m)$.*

As in the previous definition we use angles — $\langle$ and $\rangle$ — when referring to descriptions. We assume that there is an effective enumeration of the set of descriptions. This way, $\langle \phi_e^{m,n} \rangle$ is the description coded by e, or the eth description, with m n-ary components. $\phi_e^{m,n}$ is the function described by $\langle \phi_e^{m,n} \rangle$ and $W_e^{m,n}$ and $E_e^{m,n}$ are, respectively, the domain and range of $\phi_e^{m,n}$. When omited m and n are assumed to be 1. Having

[3] Some conditions of existence and uniqueness for this type of differential equation are given in [16, 17]. These restrictions on h and g mean that differential recursion is a partial operation.

[4] We could obtain the same class using only $\limsup$ (see [9]), but we use all three limits for convenience.

established clearly that the functions in this class are vector-valued, we will refer to them simply as real recursive functions.

Definition 2.3 *A* real recursive operator Ω*, is a (possibly partial) mapping from* $\mathrm{REC}(\mathbb{R})^n$ *to* $\mathrm{REC}(\mathbb{R})$*.* Ω *is said to be an* effective real recursive operator *when there is an effective procedure to obtain the description of the function* $\Omega(f_1,\ldots,f_n)$ *from the descriptions of* $f_1,\ldots,f_n$*.*

Notice that one should not confuse the operator that transforms functions with the effective process which transforms the syntactic descriptions of functions. Nevertheless, we will only deal with effective real recursive operators, and we will thus avoid using the word *effective* every time we refer to a real recursive operator.

We now begin to study $\mathrm{REC}(\mathbb{R})$. To provide insight to the reader unfamiliarised with this class we will prove a few propositions, important to the work below. Those who have already gained some knowledge and intuition with real recursive functions may simply read the propositions and skip the proofs. The following proposition is not new.

Proposition 2.4 *In* $\mathrm{REC}(\mathbb{R})$ *we can find:*

(a) The functions $\lambda x.\frac{1}{x}$, log *and* exp *of arity* 1 *and the functions* $+$, $\times$, $/$ *and* $\lambda xy.x^y$ *of arity* 2*.*

(b) Kronecker's δ *and Heaviside's* Θ*, given by*

$$\delta(x)=\begin{cases}1 & \text{if } x=0\\ 0 & \text{otherwise}\end{cases} \quad \text{and } \Theta(x)=\begin{cases}1 & \text{if } x\geqslant 0\\ 0 & \text{otherwise.}\end{cases}$$

(c) The sawtooth wave function r *and the square wave function* s*.*

(d) The floor and absolute value functions.

(e) Characteristics eq *of equality,* le *of inequality and* lt *of proper inequality.*

Proof. We proceed as in [14]. One can express $+$ and $\times$ using simple recursion schemes: $+(x,0)=x+0=I_1^1(x)=x$, $\partial_y+(x,y)=\partial_y(x+y)=1^3(x,y,x+y)=1$, $\times(x,0)=x\times 0=0^1(x)=0$ and $\partial_y\times(x,y)=\partial_y(x\times y)=I_1^3(x,y,x\times y)=x$. These show that the functions $+$ and $\times$ are both real recursive. The inverse as well as the logarithm functions are obtained by first defining shifted functions, and then undoing the shift by means of composition: first $\frac{1}{0+1}=1^0=1$, $\partial_x\frac{1}{x+1}=-\left(\frac{1}{x+1}\right)^2$, and then $\frac{1}{x}=\frac{1}{(x-1)+1}$. We set: $\log(0+1)=0$, $\partial_x\log(x+1)=\frac{1}{x+1}$, $\delta(x)=\lim\limits_{y\to\infty}\left(\dfrac{1}{x^2+1}\right)^y$ and $\Theta(x)=\left(\lim\limits_{y\to\infty}\dfrac{1}{1+2^{-xy}}\right)+\dfrac{1}{2}\delta(x)$. The square function is easily real recursive by $s(x)=\Theta(\sin(\pi x))$ and the example below. We can build the sawtooth using the recursion scheme $r^\dagger(0)=0$ and $\partial_y r^\dagger(y)=4\sin(\pi x)^2 s(x)-1$. Now we get $r(y)=s(x)r^\dagger(-x)+(1-s(x))r^\dagger(-x-1)$. As for the remaining functions, we set: $\lfloor x\rfloor=x-r(x)$, $|x|=(2\Theta(x)-1)x$, $eq(x,y)=\delta(y-x)$, $le(x,y)=\Theta(y-x)$ and $lt(x,y)=le(x,y)-eq(x,y)$. □

Besides a historical motive there are reasons to define a class of vector-valued functions, instead of using scalar functions. Some functions we would like to see in this class are obtainable only through higher-order differential equations. In these cases we try to separate the higher order derivatives into the several different components of the vector.

Example 2.5 *We can show that* sin *and* cos *are real recursive functions. In the differential recursion schema*

$$\bar f(0)=\begin{pmatrix}f_1\\ f_2\end{pmatrix}(0)=\begin{pmatrix}0\\ 1\end{pmatrix},\qquad \partial_x\bar f(x)=\partial_x\begin{pmatrix}f_1\\ f_2\end{pmatrix}(x)=\begin{pmatrix}f_2\\ -f_1\end{pmatrix}(x),$$

the solution can be recognised as $\sin(x)=I_1^2(\bar f(x))$ *and* $\cos(x)=I_2^2(\bar f(x))$*.*

Definition 2.6 *The* floored iteration operator $\mathfrak{I}$ *is defined by the expression*

$$(\mathfrak{I}f)(\bar x,y)=f\circ f\circ f\circ\ldots\circ f(\bar x)\qquad (\lfloor|y|\rfloor \text{ times}).$$

We abbreviate $f^{\lfloor y\rfloor}(\bar x)\equiv(\mathfrak{I}f)(\bar x,y)$, and use the convention $f^0(\bar x)=\bar x$.[5] It is important to notice that the function f is iterated a positive integer number of times. While it has been shown before that $\mathrm{REC}(\mathbb{R})$ is closed under iteration, we provide a new proof of the same fact, now with a completely smooth iteration scheme.

[5] If $f(\bar x)$ is undefined, on the other hand, we conventionalise that $f^0(\bar x)$ is also undefined.

Proposition 2.7 *The floored iteration operator is real recursive.*
Proof. Consider an arbitrary real recursive n-ary function f with n components. Let g and h be two real recursive $(n+1)$-ary functions with n components, such that

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

$$\partial_y h(\bar{x}, y) = (f(s(y)g(\bar{x}, y) + (1 - s(y))\bar{x}) - g(\bar{x}, y))\frac{\pi}{2}\sin(\pi y)s(y),$$

$$\partial_y g(\bar{x}, y) = \frac{(h(\bar{x}, y) - g(\bar{x}, y))\pi\sin(\pi y)}{\cos(\pi y) - 1 + \delta(\cos(\pi y) - 1)}s(-y),$$

We can set

$$f^{\lfloor y \rfloor}(\bar{x}) = g(\bar{x}, 2\lfloor |y| \rfloor).$$

These functions can be explained the following way: as y changes from 0 to 1, g is constant and h goes through the distance from $\bar{x}$ to $f(\bar{x})$. For $y \in [1, 2]$, h is constant and g catches up, hence $g(\bar{x}, 2) = h(\bar{x}, 2) = f(\bar{x})$. If $y > 2$, then the same cycle begins again, and thus, for every natural number n, $f^n(\bar{x}) = h(2n) = g(2n)$. □

Example 2.8 *We show in Figure 1 the plot of g and h for five iterations of the logistic function $\lambda x.\alpha x(1-x)$ with $\alpha = 3.9$ and initial value 0.22.*

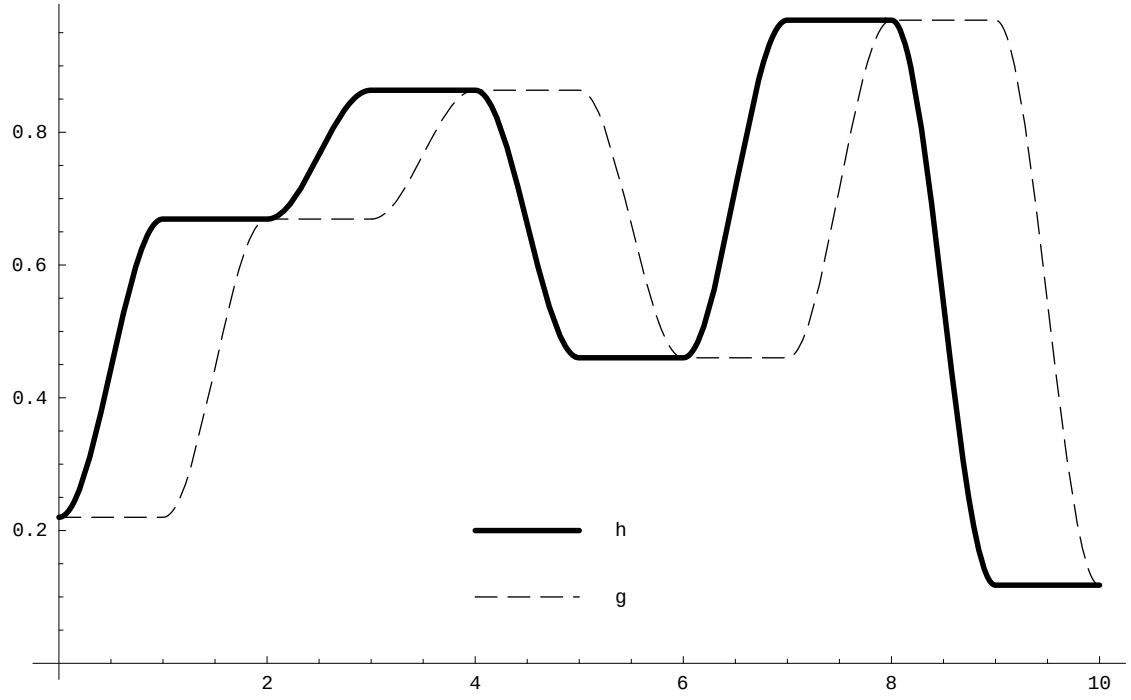

Fig. 1. Iteration of the logistic function.

Very pathological functions of analysis are real recursive too. Take, for instance, the everywhere continuous and nowhere differentiable Weierstraß function, w, given by

$$w(x) = \sum_{n=0}^{\infty} a^n cos(b^n \pi x), \qquad 0 < a < 1, \quad ab > 1 + \frac{1}{3}\pi.$$

Proposition 2.9 *The Weierstraß function is real recursive for all real recursive numbers a and b.*

Proof. The $\sum$ operator is easily real recursive, by means of an iteration,[6] and $w(x) = \lim_{y\to\infty} \sum_{n=0}^{\lfloor y \rfloor} a^n cos(b^n \pi x)$. □

We conventionalise that a function $f : \mathbb{N}^m \to \mathbb{N}$ is in REC($\mathbb{R}$) whenever there is a real recursive extension to it, i.e., a real recursive function $f^\dagger : \mathbb{R}^m \to \mathbb{R}$ such that, for all $n \in \mathbb{N}$, $f^\dagger(n) = f(n)$ if $f(n) \downarrow$ and $f^\dagger(n) \uparrow$ whenever $f(n) \uparrow$. We use an identical convention for functions with mixed signatures, e.g., $f : \mathbb{N}^2 \times \mathbb{R} \times \mathbb{N} \to \mathbb{N} \times \mathbb{R}$. For primitive recursive functions we can use a canonical extension, as explained in [18].

Definition 2.10 *The* natural tuple coding functions $\mathbf{p}_n : \mathbb{N}^n \to \mathbb{N}$ *and* $\mathbf{p}_{n,i} : \mathbb{N} \to \mathbb{N}$ *are such that, for all* $1 \leqslant i \leqslant n$, $\mathbf{p}_{n,i}(\mathbf{p}_n(m_1, m_2, \ldots, m_n)) = m_i$ *and* $\mathbf{p}_n(\mathbf{p}_{n,1}(m), \ldots, \mathbf{p}_{n,n}(m)) = m$.

Proposition 2.11 *Natural tuple coding functions are real recursive.*

Proof. $\mathbf{p}_n$ is real recursive, as we can see by the expression $\mathbf{p}_n(m_1, \ldots, m_n) = 2^{m_1} \times 3^{m_2} \times \ldots \times (p_n)^{m_n}$, where p_n is the nth prime. We also have a real recursive characteristic of divisibility, h, such that $h(d, n) = 1$ if $d|n$ and $h(d, n) = 0$ otherwise. h can be given by the following expression[7] $h(d, n) = \left(\frac{\sin(n\pi)}{d \sin(\frac{n\pi}{d})}\right)^2$. Now take

$$\mathbf{p}^\dagger_{n,i}(c, x) = \begin{cases} (c + 1, \dfrac{x}{p_i}) & \text{if } p_i|x, \\ (c, x) & \text{otherwise.} \end{cases}$$

We can obtain this definition by cases setting

$$\mathbf{p}^\dagger_{n,i}(c, x) = \left(c + h(p_i, x), h(p_i, x)\frac{x}{p_i} + (1 - h(p_i, x))x\right),$$

and we do this for every $1 \leqslant i \leqslant n$. We then set $\mathbf{p}_{n,i}(x) = I^2_1\left(\mathbf{p}^{\dagger\,\lfloor x \rfloor}_{n,i}(0, x)\right)$. □

It is important to understand that these functions can only code *natural* numbers. The property given in Definition 2.10 does not hold when m or m_k are not natural numbers.

We now study the relationship between the Diophantine and real recursive functions. Remember that the set of Diophantine functions is the same as the set of partial recursive functions — this result comes from the work of Yuri Matijasevic, Julia Robinson, Martin Davis and Hilary Putnam (see, e.g., [19]). We show that all Diophantine functions are real recursive, providing a new proof of the known result that every classical recursive function has a real recursive extension.

Definition 2.12 *An n-ary function f is* Diophantine *if there is a $(n+1+m)$-ary polynomial P such that:*

$$\langle z, \bar{x}\rangle \in G_f \quad \textit{if and only if} \quad \exists y_1 \ldots \exists y_m P(\bar{x}, z, y_1, \ldots, y_m) = 0$$

where G_f is f's graph,[8] *all variables are in $\mathbb{N}^+$.*

Proposition 2.13 *All Diophantine functions are real recursive.*[9]

Proof. Clearly a polynomial is real recursive, because the functions sum and product and the operator composition are enough to define any polynomial. Now to find a number z such that the tuple $\langle z, \bar{x}\rangle$ is in G_f, we can search for the zeros of the polynomial using a zero search operator. Let $\mathfrak{Z}[p]$ be an $(n+1)$-ary function of $(n+1)$ components, where p is an $(n+1+m)$-ary scalar function:

$$(\mathfrak{Z}p)(\bar{x}, zY) = \begin{cases} (\bar{x}, zY) & \text{if } p(\bar{x}, \mathbf{p}_{m+1,1}(zY), \ldots, \mathbf{p}_{m+1,m+1}(zY)) = 0, \\ (\bar{x}, zY + 1) & \text{otherwise.} \end{cases}$$

See that this function is real recursive because the function $Z(x) = \sin(\pi 2^{x-1})$, is real recursive and such that $Z(0) = 1$ and $Z(n) = 0$ for all $n > 0$. This allows us to use products and sums to express $\mathfrak{Z}$, as long as

[6] See [14] for the $\prod$ operator, for instance. Notice that although the 2-ary sum is real recursive, and we can compose it to show that the n-ary sum is also real recursive, the $\sum$ operator sums an *arbitrary* and *non fixed* number of terms, and cannot be shown real recursive by this method.

[7] Although the function h is undefined whenever $d|n$, it is only shown to replace the more complex expression: $\frac{1}{d}\sum_{i=1}^{\lfloor \frac{d}{2} \rfloor}(-1)^{i+1}\binom{d-i-1}{i}(2\cos(\frac{n\pi}{d}))^{d-2i} = \frac{\sin n\pi}{d \cos \frac{n\pi}{d}}$, which is also real recursive and defined everywhere.

[8] A function's graph is the set of tuples $\{\langle z, \bar{x}\rangle | f(\bar{x}) \downarrow \text{ and } z = f(\bar{x})\}$. Notice that the graph can be partial, i.e., if a function f is not defined for some point $\bar{x}$, then there is no tuple $(z, \bar{x})$ in G_f.

[9] The discussion in page 7 gives the precise meaning of this.

the numbers involved are natural numbers, in a similar manner as in the proof of proposition 2.11. Suppose that f is an arbitrary Diophantine function as in Definition 2.12. Then there is some z such that $\langle z, \bar{x}\rangle \in G_f$ if and only if the iteration of $(\mathfrak{Z}P)(\bar{x}, 1)$ converges. So f is real recursive because it can be given by

$$f(\bar{x}) = \mathbf{p}_{m+1,1}\left(I^{n+1}_{n+1}\left(\lim_{y\to\infty}\left[(\mathfrak{Z}P)^{\lfloor y\rfloor}(\bar{x}, 1)\right]\right)\right).$$

□

So all Turing-computable (or $\mathbb{N}$-recursive, or Diophantine, or λ-definable) functions are real recursive. The converse is not true, though: as will soon become evident, there is a real recursive function which solves the halting problem for Turing machines. In fact there is a real recursive function which decides all predicates in the arithmetical hierarchy (see [14]).

Definition 2.14 *The operators* $\mathfrak{sup}$ *and* $\mathfrak{inf}$*, of supremum and infimum over an interval, are defined by the expressions*

$$(\mathfrak{sup} f)(\bar{x}, a, b) = \sup_{y\in[a,b)} f(\bar{x}, y), \qquad (\mathfrak{inf} f)(\bar{x}, a, b) = \inf_{y\in[a,b)} f(\bar{x}, y),$$

where f *is an* $(n+1)$*-ary real recursive function defined in* $[a, b)$*.*

The following proposition is known since [9]. We provide a simpler proof.

Proposition 2.15 $\mathfrak{sup}$ *and* $\mathfrak{inf}$ *are both real recursive.*

Proof. It is sufficient to transform a given function f from the interval $[a, b)$ to a periodic function on $[0, \infty)$. We define the new periodic function

$$F(\bar{x}, y, a, b) = f(\bar{x}, a + (y \bmod (b - a)))$$

where $(y \bmod z)$ is the number in $[0, z)$ such that $y = nz + (y \bmod z)$ for some natural number n, which can be obtained by iteration. Now we have that

$$(\mathfrak{sup} f)(\bar{x}, a, b) = \limsup_{y\to\infty} F(\bar{x}, y, a, b), \text{ and}$$

$$(\mathfrak{inf} f)(\bar{x}, a, b) = \liminf_{y\to\infty} F(\bar{x}, y, a, b).$$

All functions used in the above construction are real recursive, and so both $\mathfrak{sup}$ and $\mathfrak{inf}$ are real recursive. □

Previous work on real recursive functions characterised from an analytical viewpoint several hierarchies and classes of functions, such as the elementary functions and the Grzegorczyk hierarchy in [7], the Ritchie hierarchy in [20], P and NP in [21, 18]. It was shown in [6, 22] that the GPAC is not closed under iteration and that a subclass of real recursive functions coincides with the class of GPAC-computable functions. Olivier Bournez and Emmanuel Hainry proved in [23, 24] that a restricted infinite limit operator makes the class of real recursive functions an *exact* extension to the real numbers — in the sense of computable analysis — of the classical recursive functions.

3. The η-hierarchy

We now define and study the η-hierarchy, which is the main subject of this text. It is based on the minimum number of nested limits needed to describe a given real recursive function. The shown definition for the η-hierarchy is different than the previous definitions, such as the one in [14], but gives the same hierarchy. This result comes from some algebraic manipulation and from the realization that the η-number for composition, as defined here, can be achieved by an iteration scheme, using the previous definition.

Definition 3.1 *The* η-number of a description $\langle f\rangle$*,* $E(\langle f\rangle)$*, is inductively defined as follows:*

(i) $E(\langle 0^n\rangle) = E(\langle \bar{1}^n\rangle) = E(\langle 1^n\rangle) = E(\langle I^n_j\rangle) = 0$*,*
(ii) $E(\langle c(f, g)\rangle) = \max(E(\langle f\rangle), E(\langle g\rangle))$*,*
(iii) $E(\langle dr(f, g)\rangle) = \max(E(\langle f\rangle), E(\langle g\rangle))$*,*
(iv) $E(\langle l(h)\rangle) = E(\langle li(h)\rangle) = E(\langle ls(h)\rangle) = E(\langle h\rangle) + 1$*, and*
(v) $E(\langle v(f_1, \ldots, f_n)\rangle) = \max(E(\langle f_1\rangle), \ldots, E(\langle f_n\rangle))$*.*

The η-number of a real recursive vector function f, $E(f)$, *is the minimum* η*-number for all descriptions of* f. *The* η-hierarchy *is an* $\mathbb{N}$*-indexed family of real recursive vector functions, where the* n*th level of the* η*-hierarchy,* H_n, *is:*

$$\mathrm{H}_n = \{f \in \mathrm{REC}(\mathbb{R}) | E(f) \leqslant n\}$$

The scalar functions $\lambda x.\frac{1}{x}$, log, exp, sin and cos of arity 1, as well as $+$, $\times$, $/$ and $\lambda xy.x^y$ of arity 2, are in H_0. Kronecker's δ and Heaviside's Θ are in H_1, and Weierstraß' function is in H_2 for every $a, b \in \mathrm{H}_1$. The iteration operator takes H_i's functions to $\mathrm{H}_{\max(i,1)}$. The supremum and the infimum operators take H_i's functions to $\mathrm{H}_{\max(i+1,2)}$.

Now we study the η operators, which should not be mistaken with the η-hierarchy.

Definition 3.2 *For a real recursive* $(n+1)$*-ary function* f, *the* η operator *gives an* n*-ary function,* ηf, *such that:*

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

The η^i *and* η^s *operators are defined in the same way, but with respect to* $\liminf$ *and* $\limsup$.

It was previously known that the η operators are real recursive. We provide new, tighter bounds for the place of ηf, $\eta^s f$ and $\eta^i f$ in the η-hierarchy.

Proposition 3.3 *If* f *is a total function in* H_i, *then* ηf, $\eta^s f$ *and* $\eta^i f$ *are in* H_{i+1}.

Proof. Take for $\eta^s f$ the following expression:

$$(\eta^s f)(\bar{x}) = 1 - \delta\left(\liminf_{y\to\infty} \frac{1}{(f(\bar{x}, y))^2 + 1}\right).$$

A similar expression gives us $\eta^i f$. Let $f^\dagger$ be given by $f^\dagger(\bar{x}, y) = \frac{1}{1+\exp(f(\bar{x},y))}$. One has that $\lim_{y\to\infty} f(\bar{x}, y)$ converges if and only if the supremum and infimum limits of f converge and are equal. The supremum and infimum limits of f are equal if and only if the supremum and infimum limits of $f^\dagger$ are equal. For this reason ηf can be set as

$$(\eta f)(\bar{x}) = (\eta^s f)(\bar{x}) \times (\eta^i f)(\bar{x}) \times eq\left(\liminf_{y\to\infty} f^\dagger(\bar{x}, y), \limsup_{y\to\infty} f^\dagger(\bar{x}, y)\right).$$

□

These powerful operators where originally used to show that the halting problem of classic computability theory has a real recursive characteristic, as done in [14]. Yet, clearly, not all real valued vector functions are real recursive, since the cardinality of $\mathrm{REC}(\mathbb{R})$ is ${}_0\aleph = {}_0\beth$, while the cardinality of the set of real valued vector functions is ${}_2\beth$. In [25] it is shown that many predicates undecidable in classical computability have no real recursive characteristics, e.g., the predicates about real recursive functions given by the expressions $\phi_n(n) \downarrow$ and $\phi_n = \phi_m$.[10] A result of the present text is that many of these predicates do have real recursive characteristics when restricted to some level in the η-hierarchy, by means of a function in a higher level.

4. Integration and differentiation in REC($\mathbb{R}$)

The class of functions we have been studying aims to provide a framework to study continuous phenomena from a computational perspective. The differential recursion operator, introduced by Cristopher Moore in [5], is an abstraction of the integration unit available in Shannon's GPAC [2]. It was proven in [22] that the class of GPAC-computable functions almost coincides with H_0. It is legitimate to ask if differential recursion is really necessary, and the answer is no: we can remove the differential recursion operator from our inductive definition of $\mathrm{REC}(\mathbb{R})$, replacing it with an iteration operator. In the classical case this is also true (see [26]), but unlike Kleene's recursive functions, where one can replace recursion with iteration simply by adding functions to the base, in order to obtain differential recursion by means of iteration we will also need infinite limits.

Proposition 4.1 *Let* f *be a real recursive total* $(n+1)$*-ary* k*-component function.*

[10] In [25] the related problems are called, respectively, the problem of domain and the problem of identity.

(i) The Riemann integral operator $\mathfrak{int}$ *such that:*

$$(\mathfrak{int}f)(\bar{x},a,b)=\int_a^b f(\bar{x},y)\mathrm{d}y$$

defined only in points where the Riemann integral is defined,[11] *is real recursive.*

(ii) The derivative operator $\mathfrak{dif}$*, given by:*

$$(\mathfrak{dif}f)(\bar{x},y)=(\partial_y f)(\bar{x},y)$$

defined only in points where the derivative of f *is defined, is real recursive.*

Proof. We begin with the $\mathfrak{int}$ operator. Remember that by Proposition 2.15 we have that if f is real recursive, then the function $\mathfrak{sup}f$, such that $(\mathfrak{sup}f)(\bar{x},a,b)=\sup_{y\in[a,b)}f(\bar{x},y)$, is also real recursive. The infimum of the superior Darboux sums for all partitions between a and b can be given by:

$$\overline{\int_a^b} f(\bar{x},y)\mathrm{d}y=\lim_{z\to\infty}\sum_{i=1}^{\lfloor z\rfloor}(\mathfrak{sup}f)\left(\bar{x},a+(i-1)\frac{(b-a)}{\lfloor z\rfloor},a+i\frac{(b-a)}{\lfloor z\rfloor}\right)\times\frac{(b-a)}{\lfloor z\rfloor}.$$

A very similar definition can be given for the supremum of the inferior Darboux sums, and we can then conclude that

$$(\mathfrak{int}f)(\bar{x},a,b)=\int_a^b f(\bar{x},y)\mathrm{d}y=\overline{\int_a^b} f(\bar{x},y)\mathrm{d}y\times eq^{\uparrow}\left(\overline{\int_a^b} f(\bar{x},y)\mathrm{d}y,\underline{\int_a^b} f(\bar{x},y)\mathrm{d}y\right),$$

where $eq^{\uparrow}(x,y)=1+\sum_{z=1}^{\infty}(x-y)$ is 1 if $x=y$ and undefined otherwise. The $\mathfrak{dif}$ operator is similar. If we define two operators $\mathfrak{dif}^{+}$ and $\mathfrak{dif}^{-}$ such that:

$$(\mathfrak{dif}^{\pm}f)(\bar{x},y)=\lim_{\varepsilon\to\infty}\varepsilon\left(f\left(\bar{x},y\pm\frac{1}{\varepsilon}\right)-f\left(\bar{x},y\right)\right)$$

Then the $\mathfrak{dif}$ operator becomes:

$$(\mathfrak{dif}f)(\bar{x},y)=(\mathfrak{dif}^{+}f)(\bar{x},y)\times eq^{\uparrow}\left((\mathfrak{dif}^{+}f)(\bar{x},y),(\mathfrak{dif}^{-}f)(\bar{x},y)\right)$$

□

Definition 4.2 *The* second class of real recursive functions, $\mathrm{REC}(\mathbb{R})^{\dagger}$*, is the inductive class containing the atoms* $\bar{1}^n$*,* 1^n*,* 0^n *(for all* n*), the atoms* I_i^n *(for all* n*,* $1\leqslant i\leqslant n$*), and the* 2*-ary functions* $+$*,* $\times$ *and* $/$*, closed under the operators of composition, infinite limits and aggregation — as in Definition 2.1 — and for the floored iteration operator — given in Definition 2.6.*

Proposition 4.3 $\mathrm{REC}(\mathbb{R})^{\dagger}=\mathrm{REC}(\mathbb{R})$

Because $\mathrm{REC}(\mathbb{R})$ contains all basic functions in $\mathrm{REC}(\mathbb{R})^{\dagger}$ and, by Proposition 2.7, is closed for all operators which define $\mathrm{REC}(\mathbb{R})^{\dagger}$, we trivially have that $\mathrm{REC}(\mathbb{R})^{\dagger}\subseteq\mathrm{REC}(\mathbb{R})$. We will prove below that $\mathrm{REC}(\mathbb{R})\subseteq\mathrm{REC}(\mathbb{R})^{\dagger}$, sufficing to show that $\mathrm{REC}(\mathbb{R})^{\dagger}$ is closed for differential recursion, which we will do by studying the asymptotical behaviour of the Euler approximation.

Definition 4.4 *The Euler approximation operator, written* Λ*, takes a real recursive* n*-ary function* f *and a real recursive* $(n+1+k)$*-ary function* g*, both with* k *components, and gives a real recursive* $(n+2)$*-ary function with* k *components,* $\Lambda f,g$*. The* i*th component of* $\Lambda f,g$ *is given by*

$$(\Lambda f,g)_i(\bar{x},y,z)=\left((\lambda g)^{\lfloor z\rfloor}\left(\bar{x},0,f(\bar{x}),\frac{y}{|z|+1}\right)\right)_{n+1+i},$$

where λg *is defined by the expression:*

$$(\lambda g)(\bar{x},y,\bar{v},\delta)=(\bar{x},y+\delta,\bar{v}+\delta\times g(\bar{x},y,\bar{v}),\delta)\,.$$

[11] That is, in points where the superior and inferior integrals are defined and equal.

Proposition 4.5 *Let h be an $(n+1)$-ary function given by the differential recursion of $f, g \in \mathrm{REC}(\mathbb{R})^\dagger$. For every $\bar{x}$ such that $h(\bar{x}, y)$ is defined for $y \in (a, b)$, we have that*

$$h(\bar{x}, y) = \lim_{z \to \infty} (\Lambda f, g)(\bar{x}, y, z)$$

for every $y \in (a, b)$.

Proof. Since h is continuous for $y \in (a, b)$, the mean-value theorem tells us that, for all $y \in (a, b)$, $\delta > 0$ such that $(y + \delta) \in (a, b)$,

$$h(\bar{x}, y + \delta) = h(\bar{x}, y) + \delta \times \partial_y h(\bar{x}, d) = h(\bar{x}, y) + \delta \times g(\bar{x}, d, h(\bar{x}, d)),$$

for some $d \in (y, y + \delta)$. As z goes to ∞, $\delta = \frac{y}{|z|+1}$ goes to zero, and so, since Λ sets $d = y$ in the iteration of λg, and since as δ goes to zero d goes to y, we have that the shown limit converges to h. □

Figure 2 illustrates the functioning of the Λ operator, where $f = (0, 1)$ and $g(x, z_1, z_2) = (z_2, -z_1)$.

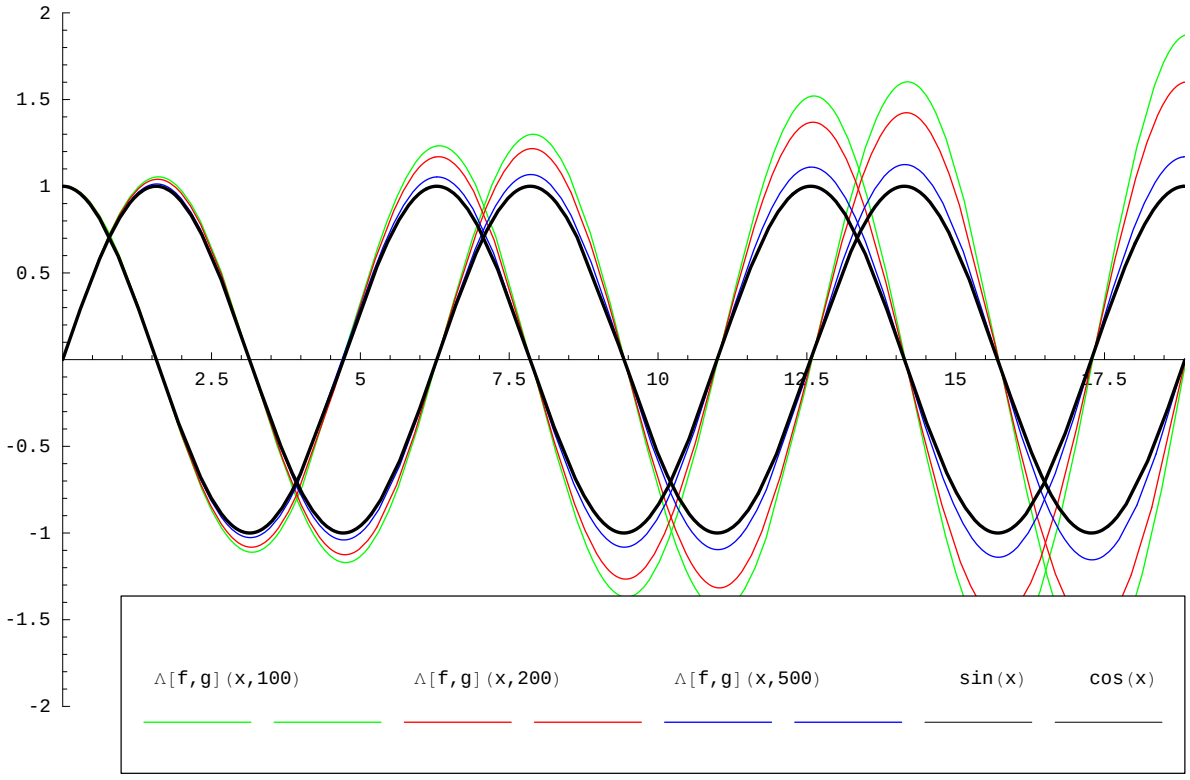

Fig. 2. Euler approximation of sin and cos.

From Proposition 4.5 the following corollary becomes evident.

Proposition 4.6 *If h is in $\mathrm{REC}(\mathbb{R})$, then there is a function $h^\dagger$ in $\mathrm{REC}(\mathbb{R})^\dagger$ which coincides with h in its domain.*

It is not necessarily the case that outside (a, b) the Euler approximation remains undefined. If the function h — obtained by differential recursion from f and g — is undefined outside (a, b) because g is undefined in these points, then the Euler approximation will also be undefined outside (a, b). The issue is when h is undefined because it diverges for some y inside the interval where g is defined. Take, for instance, $f = 1$ and $g(y, z) = z^2$. While g is a total function we get $h(y) = \frac{1}{y+1}$ for every $y > -1$ and undefined everywhere else — since h must be continuous; on the other hand, the Euler approximation of this function converges to $h^\dagger$, given by $h^\dagger(y) = \frac{1}{y+1}$ for all y. We eliminate this difference by restricting the domain.

Proposition 4.7 *Let h be an $(n+1)$-ary real recursive function which is in H_k and for every $\bar{x}$ there is an open interval (a,b) where the function $h(\bar{x},\cdot)$ is continuous. Then there is a real recursive function $H \in \mathrm{H}_{k+1}$ such that*

(i) $H(\bar{x},y) = 1$ *for all* $y \in (a,b)$,
(ii) if $\lim_{y\to a^+} h(\bar{x},y)$ *is undefined, then* $H(\bar{x},a) = 0$, *and*
(iii) if $\lim_{y\to b^-} h(\bar{x},y)$ *is undefined, then* $H(\bar{x},b) = 0$.

Proof. We can simply take $H(\bar{x},y) = \eta_z h(\bar{x}, y - \frac{y}{z})$. □

Proposition 4.8 [12] *For every real recursive function $f \in \mathrm{H}_k$, there are two real recursive functions χ_f and τ_f in H_{k+6}, such that*

$$\chi_f(\bar{x}) = \begin{cases} 1 & \text{if } \bar{x} \in Dom(f), \\ 0 & \text{otherwise;} \end{cases} \quad \text{and} \quad \tau_f(\bar{x}) = \begin{cases} f(\bar{x}) & \text{if } \bar{x} \in Dom(f), \\ 0 & \text{otherwise.} \end{cases}$$

Proof. The proof is done by induction on REC($\mathbb{R}$). The case for the basic functions and aggregation being trivial, we proceed to the remaining operators.

Composition. Let h be given by $h(\bar{x}) = f \circ g(\bar{x})$. Then we can set $\chi_h(\bar{x}) = \chi_g(\bar{x}) \times \chi_f(\tau_g(\bar{x}))$, and $\tau_h(\bar{x}) = \chi_g(\bar{x}) \times \tau_f(\tau_g(\bar{x}))$.

Differential recursion. Suppose h is given by differential recursion on f and g. By definition there is for every $\bar{x}$ some open interval (a,b) such that $Dom(h(\bar{x},\cdot)) = (a,b)$. We must consider two cases:
(1) $b = b^{(1)}$ (or $a = a^{(1)}$) is the smallest positive y (resp. the largest negative y) for which $(\bar{x},y,h(\bar{x},y)) \notin Dom(g)$.
(2) $b = b^{(2)}$ (or $a = a^{(2)}$) is the positive (resp. negative) point for which h diverges. [13]
Take H to be the function explained in the previous proposition. From the results regarding minimalization — particularly Theorem 4.1 in [9] —, we conclude that there exists a real recursive function h_1^b such that

$$h_1^b(\bar{x},y^\dagger) = \mu_y \left[y = y^\dagger \vee H(\bar{x},y) = 0 \right]$$

This function is in H_{k+1+5}, and the minimalization is always defined. If h diverges on $b^{(2)} \leqslant y^\dagger$, then $h^b(\bar{x},y^\dagger) = b^{(2)}$. If not, then $h^b(\bar{x},y^\dagger) = y^\dagger$. Now we construct a function, h_2^b, which is valued 1 for $y \in (-\infty, b^{(2)}]$, equal to 0 for $y \in (b^{(2)}, b^{(1)})$ and undefined everywhere else:

$$h_2^b(\bar{x},y) = le(y, h_1^b(\bar{x},y)).$$

We claim that similar functions can now trivially be constructed for a. To deal with case (1) we establish a recursion scheme very similar to the recursion scheme of h:

$$s(\bar{x},0) = \chi_f(\bar{x}), \partial_y s(\bar{x},y) = -eq(\chi_g(\bar{x},y,t(\bar{x},y)),0),$$
$$t(\bar{x},0) = \tau_f(\bar{x}), \partial_y t(\bar{x},y) = eq(s(\bar{x},y),1)\tau_g(\bar{x},y,t(\bar{x},y)).$$

For case (2) this recursion scheme gives exactly $t = h$, but for case (1) the behaviour is different: for $y \in [0,b)$ we have $s(\bar{x},y) = 1$ and $t(\bar{x},y) = h(\bar{x},y)$; for $y \in [b,+\infty)$ we have $s(\bar{x},y) < 1$ and $t(\bar{x},y) = h(\bar{x},b)$. To decide if y is such that $(\bar{x},y,h(\bar{x},y)) \in Dom(g)$, we use the function:

$$\chi_h^\dagger(\bar{x},y) = le(1, s(\bar{x},y)).$$

In order to cover both cases, we consider the differential recursion scheme

$$u(\bar{x},0) = \chi_f(\bar{x}), \partial_y u(\bar{x},y) = -eq(H(\bar{x},v(\bar{x},y)),0) - eq\left(\chi_h^\dagger(\bar{x},H(\bar{x},v(\bar{x},y)) \times v(\bar{x},y)),0\right),$$
$$v(\bar{x},0) = 0, \partial_y v(\bar{x},y) = eq(u(\bar{x},y),1),$$

and set $\chi_h(\bar{x},y) = le(u(\bar{x},y),1)$, $\tau_h(\bar{x},y) = h(\bar{x},\chi_h(\bar{x},y) \times y) \times \chi_h(\bar{x},y)$.

Infinite limits. Should some function h be given by $h(\bar{x}) = \lim_{y\to\infty} f(\bar{x},y)$, then we must consider several cases. The limit $\lim_{y\to\infty} \tau_f(\bar{x},y)$ can be undefined, equal to some $z \neq 0$ or equal to 0. In the first

[12] It was shown in [25] that there was no real recursive function which would, given a code n of ϕ_n and a point x, decide if $x \in W_n$. This does *not* contradict Proposition 4.8.
[13] Say, with an asymptote, as in $\frac{1}{x-1}$, or with an infinite oscillation, as with $\sin\left(\frac{1}{x-1}\right)$: in both cases $b^{(2)} = 1$.

two cases, the former limit has the same behaviour as the later, but in the case of convergence to 0 one needs to distinguish a convergence of both f and τ_f from a convergence of τ_f but not of f. Observing that $\tau_{S\circ f}(\bar{x}, y) = (\tau_f(\bar{x}, y) + 1)\chi_f(\bar{x}, y)$, where S is the successor function, all these cases are dealt by the definition

$$\chi_h = (\eta\tau_f) \times (\eta\tau_{S\circ f}) \times eq\left(\lim_{y\to\infty}[(\eta\tau_f)\tau_f] + 1, \lim_{y\to\infty}[(\eta\tau_{S\circ f})\tau_{S\circ f}]\right),$$

the parameters being omitted for convenience. We use similar definitions for the supremum and infimum limits, and set $\tau_h(\bar{x}) = \lim_{y\to\infty}(\eta\tau_f)(\bar{x})\tau_f(\bar{x}, y)$. □

Proof of Proposition 4.3. Take a function h in $\mathrm{REC}(\mathbb{R})$, and let $h^\dagger$ be a function in $\mathrm{REC}(\mathbb{R})^\dagger$ equal to h in its domain. Since χ_h is real recursive and total, then by Proposition 4.6 we have $\chi_h \in \mathrm{REC}(\mathbb{R})^\dagger$. It becomes evident that h is also in $\mathrm{REC}(\mathbb{R})^\dagger$, setting $h(\bar{x}) = \dfrac{h^\dagger(\bar{x})}{\chi_h(\bar{x})}$. □

We can now conclude that if h is a real recursive function in H_k, then one can define it with iteration using no more than $k + d + 6$ nested limits, where d is the number of nested differential recursions. This means that by increasing d one could arbitrarily increase the number of nested limits required to define h in $\mathrm{REC}(\mathbb{R})^\dagger$. Now we show that this is not the case, since all these d limits, which are required to take the number of approximation steps to the infinite, can be replaced by a single limit.

Proposition 4.9 *If h is in H_k, then one can show that h is in $\mathrm{REC}(\mathbb{R})^\dagger$ using $k + 7$ nested limits.*

Proof. Take the expression which testifies that h is in $\mathrm{REC}(\mathbb{R})^\dagger$, as constructed in the proof of Proposition 4.3. Consider every sub-expression of the form $(\Lambda f, g)$. This expression gives, for some n, an $(n + 2)$-ary function which converges uniformly — on the last argument and for every closed interval — to the corresponding function given by differential recursion. Since this is the case, one says that $\lim_{z\to\infty}(\Lambda f, g)(\bar{x}, y, z)$ converges uniformly to the function obtained by differential recursion on f and g. Because of this uniform convergence the infinite limit can be exchanged with other infinite limits, e.g.,

$$\lim_{y\to\infty}\lim_{z\to\infty}(\Lambda f, g)(\bar{x}, y, z) = \lim_{z\to\infty}\lim_{y\to\infty}(\Lambda f, g)(\bar{x}, y, z).$$

One can thus easily prove by induction on $\mathrm{REC}(\mathbb{R})^\dagger$ that every such uniformly convergent infinite limit $\lim_{z_1\to\infty}, \ldots, \lim_{z_j\to\infty}$ can be moved into a chain of limits at the beginning of the description which testifies that h is in $\mathrm{REC}(\mathbb{R})^\dagger$. Again because of uniform convergence all of these limits can be replaced by a single limit. Observing that $h(\bar{x}) = \frac{h^\dagger(\bar{x})}{\chi_h(\bar{x})}$, one concludes that h can be defined with $k + 7$ nested limits. □

5. The universal Ψ function

In this section we use classical methods of computability to show that there is no universal real recursive function. We proceed to give the notion of low-rank code, and state that there is no real recursive restriction of the universal function to low-rank codes.

Proposition 5.1 *There is no real recursive function I such that for all $m, n \in \mathbb{N}$, $x \in \mathbb{R}$:*

$$I(m, n, x) = \begin{cases} 1 & \text{if } \phi_m(x) = \phi_n(x), \\ 0 & \text{if } \phi_m(x) \neq \phi_n(x). \end{cases}.$$

We consider above that if $\phi_m(x) \uparrow$ and $\phi_n(x) \uparrow$, then $\phi_m(x) = \phi_n(x)$.

Proof. The proof is similar to the one in [25]. Let us suppose that the mentioned function I is real recursive. Let $f \equiv \phi_m$ be a total real recursive function with code m. The function g, given by $g(x) = f(x)+I(m, \lfloor x\rfloor, x)$, is total and real recursive. Let n be the code for g. If $g(n) = f(n)$, then $I(m, n, n) = 1$ and so $g(n) = f(n)+1$. If $g(n) \neq f(n)$, then $I(m, n, n) = 0$ and $g(n) = f(n)$. By contradiction we conclude that I is not real recursive. □

While there is currently no conceptual *machine* to give an operational meaning to the class of real recursive functions, we can still define what we mean by a *universal real recursive function*, and conclude the following.

Theorem 5.2 *There is no universal real recursive function, i.e., there is no real recursive scalar function Ψ of arity 2 such that, for all $n \in \mathbb{N}$, $x \in \mathbb{R}$, $\Psi(n, x) = \phi_n(x)$.*

Proof. Let us suppose that Ψ was real recursive. Let S be the real recursive successor function. By proposition 4.8 there are two real recursive functions τ_Ψ and $\tau_{S\circ\Psi}$ which are total equivalents of Ψ and $S\circ\Psi$. Let $a=\tau_\Psi(m,x)$, $b=\tau_\Psi(n,x)$, $A=\tau_{S\circ\Psi}(m,x)$ and $B=\tau_{S\circ\Psi}(n,x)$. See that if $\phi_m(x)$ and $\phi_n(x)$ are both defined and equal or both undefined then $a=b$ and $A=B$. If $\phi_m(x)$ and $\phi_n(x)$ are both defined and different, then $a\neq b$ and $A\neq B$. Finally, if $\phi_m(x)$ is undefined and $\phi_n(x)$ is defined, we will necessarily have $a\neq b$ if $\phi_n(x)\neq 0$ or $A\neq B$ otherwise. We can conclude that I is real recursive, since

$$I(m,n,x)=eq(\tau_\Psi(m,x),\tau_\Psi(n,x))\times eq(\tau_{S\circ\Psi}(m,x),\tau_{S\circ\Psi}(n,x)).$$

By contradiction we see that Ψ cannot be real recursive. □

One should note that although the results above are for 1-ary functions, they easily extend to n-ary functions by Proposition 6.4 below.

Definition 5.3 *We say that n is a* low-rank code*, and write $\mathbb{L}(n)$, if the nth description, $\langle\phi_n\rangle$, has the smallest η-number needed to define ϕ_n, i.e., if $E(\langle\phi_n\rangle)=E(\phi_n)$.*

Proposition 5.4 *There is no real recursive restriction of Ψ for low-rank codes, i.e., there is no real recursive scalar function ψ of arity 2 such that if $\mathbb{L}(n)$, then $\psi(n,x)=\phi_n(x)$.*

The proof can be done using the same method, by showing that there is no restriction of I for low-rank codes. We will need this result in order to prove that the η-hierarchy does not collapse in theorem 6.2.

6. The universal Ψ_n functions

In this section we will make use of Proposition 5.4 and of the alternative inductive structure for $\mathrm{REC}(\mathbb{R})$, explained in Definition 4.2 and supported by Propositions 4.3 and 4.9, to show that the η-hierarchy does not collapse. Specifically, we will prove that:

Proposition 6.1 *There is a universal real recursive function for every level of the η-hierarchy, i.e., for every level H_n of the η-hierarchy, there is a function Ψ_n of arity 2 such that whenever $E(\langle\phi_e\rangle)\leqslant n$, we have $\Psi_n(e,x)=\phi_e(x)$.*

The function Ψ_n is probably not in H_n, but it suffices to show that it exists in a higher level of the η-hierarchy. To do this, we will construct real recursive functions to manipulate an analogue of an *execution stack,* working with a set of alternative descriptions.

Theorem 6.2 *The η-hierarchy does not collapse, i.e., there is no number n such that $\mathrm{REC}(\mathbb{R})\subseteq\mathrm{H}_n$.*

Proof. Let us suppose that there is a number n such that $\mathrm{REC}(\mathbb{R})\subseteq\mathrm{H}_n$. We know by Proposition 6.1 that a universal function for H_n, namely Ψ_n, exists. Since, by hypothesis, all real recursive functions need at most n nested limits, then there is a restriction to low-rank codes of the universal function: $\psi=\Psi_n$. By Proposition 5.4 we see that this function cannot exist, so we have shown by contradiction that the η-hierarchy does not collapse. □

Definition 6.3 *The set of* alternative descriptions of real recursive functions *is inductively defined as follows:*

- *Alternative descriptions $\langle\langle\bar{1}^n\rangle\rangle$, $\langle\langle 1^n\rangle\rangle$, $\langle\langle 0^n\rangle\rangle$, $\langle\langle I_j^n\rangle\rangle$, $\langle\langle c(h,g)\rangle\rangle$, $\langle\langle l(h)\rangle\rangle$, $\langle\langle li(h)\rangle\rangle$, $\langle\langle ls(h)\rangle\rangle$ and $\langle\langle v(f_1,\ldots,f_m)\rangle\rangle m$ are as in Definition 2.2.*
- *$\langle\langle+\rangle\rangle$, $\langle\langle\times\rangle\rangle$ and $\langle\langle/\rangle\rangle$ are alternative descriptions of the 2-ary functions $+$, $\times$ and $/$, respectively.*
- *If $\langle\langle h\rangle\rangle$ is an n-ary alternative description of the real recursive vector function h, then $\langle\langle I(h)\rangle\rangle$ is an $(n+1)$-ary alternative description of the real recursive vector function $\mathfrak{I}h$ — as in Definition 2.6.*

We use two angles when referring to alternative descriptions. We consider an effective enumeration of the set of alternative descriptions, such that $\langle\langle\sigma_e^{m,n}\rangle\rangle$ is the alternative n-ary description with m components coded by e and $\sigma_e^{m,n}$ is the function described by $\langle\langle\sigma_e^{m,n}\rangle\rangle$. This enumeration will have the expected nice properties, e.g., one can effectively obtain the code of $\langle\langle f_1\rangle\rangle$ from the code of $\langle\langle v(f_1,\ldots,f_m)\rangle\rangle$. Additionally, we demand some *tokens* are enumerated as well, which we will use as an auxiliary tool: for each n there should be a code for $\langle\langle\mathrm{swt},n\rangle\rangle$ and $\langle\langle\mathrm{dup},n\rangle\rangle$. The meaning of these tokens will soon become clear.

We have shown in Proposition 2.11 that we can code tuples of natural numbers. The proposition below, which appears as Lemma 2.9 in [9], tells us that we can also code tuples of real numbers, and we use these functions to manipulate stacks of real numbers.

Proposition 6.4 *There are three real recursive functions γ, γ_1 and γ_2 in* H_3*, such that for all $x, y \in \mathbb{R}$:*

$$\gamma(\gamma_1(x), \gamma_2(x)) = x, \gamma_1(\gamma(x, y)) = x, \gamma_2(\gamma(x, y)) = y,$$

and $\gamma_1(0) = \gamma_2(0) = 0$.

We will represent an empty stack by the number 0 and the stack $\#x_n \dots x_1$ of n real numbers, where $\#$ marks the top of the stack, with the number $\gamma(x_n + \Theta(x_n), \dots \gamma(x_1 + \Theta(x_1), 0) \dots)$. The Θ function is used to distinguish the empty stack from the stack with the number 0. We can then build four *basic stack manipulation functions*. The psh function, which pushes a value on top of the stack, is given by $\mathrm{psh}(S, x) = \gamma(x + \Theta(x), S)$. The pop function removes the top of the stack: $\mathrm{pop}(S) = \gamma_2(S)$. The top function gives the value on the top of the stack, and 0 if the stack is empty: $\mathrm{top}(S) = \gamma_1(S) - \Theta(\gamma_1(S)) + \delta(\gamma_1(S))$. The emp function gives 1 if the stack is empty and 0 otherwise: $\mathrm{emp}(S) = \delta(S)$. These basic functions are in H_3. We abbreviate $\mathrm{top}(\mathrm{pop}^{\lfloor n-1 \rfloor}(S)) \equiv \mathrm{tp}(S, n)$. More complex stack manipulation functions can be defined using the four basic functions. The function swt, for instance, pushes the top of the stack into the $(n+1)$th position: $\mathrm{swt}(S, n) = I_1^4(\mathrm{swt}^{\dagger \lfloor 2n+2 \rfloor}(S, 0, n, 0))$, where

$$\mathrm{swt}^\dagger(S_1, S_2, n, r) = \begin{cases} (\mathrm{pop}(S_1), 0, n, \mathrm{top}(S_1)) & \text{if } r = 0 \text{ and } n > 0, \\ (\mathrm{pop}(S_1), \mathrm{psh}(S_2, \mathrm{top}(S_1)), n-1, r) & \text{if } r \neq 0, \text{ and } n > 0, \\ (\mathrm{psh}(S_1, r), S_2, 0, 0) & \text{if } r \neq 0 \text{ and } n = 0, \\ (\mathrm{psh}(S_1, \mathrm{top}(S_2)), \mathrm{pop}(S_2), 0, 0) & \text{if } r = 0 \text{ and } n = 0. \end{cases}$$

The function dup duplicates the top n elements of the stack: $\mathrm{dup}(S, n) = I_1^4(\mathrm{dup}^{\dagger \lfloor 3n+1 \rfloor}(S, 0, 0, n))$, where

$$\mathrm{dup}^\dagger(S_1, S_2, S_3, n) = \begin{cases} (\mathrm{pop}(S_1), \mathrm{psh}(S_2, \mathrm{top}(S_1)), 0, n-1) & \text{if } n > 0, \\ (S_1, S_2, S_2, 0) & \text{if } n = 0, \text{ and } \mathrm{emp}(S_3), \\ (\mathrm{psh}(S_1, \mathrm{top}(S_2)), \mathrm{pop}(S_2), S_3, 0) & \text{if } n = 0 \text{ and not } \mathrm{emp}(S_2), \\ (\mathrm{psh}(S_1, \mathrm{top}(S_3)), S_2, \mathrm{pop}(S_3), 0) & \text{if } n = 0 \text{ and not } \mathrm{emp}(S_3), \end{cases}$$

We conclude that these functions are also in H_3. We can now understand the meaning of the tokens $\langle\langle \mathrm{swt}, n \rangle\rangle$ and $\langle\langle \mathrm{dup}, n \rangle\rangle$ mentioned above: their codes will represent the instructions *apply* swt *or* dup *to the stack, with n as the second argument.*

Proof of Proposition 6.1. To simulate a real recursive function step-by-step, given its code, we maintain two stacks. On the first stack we keep real values and on the second stack we maintain codes of descriptions of real recursive functions or of stack manipulation instructions. If the function in the top of the second stack is n-ary, it is expected that n real values are in the first stack, each corresponding to one argument, with the last argument on top. To implement the aggregation operator, it will be necessary to duplicate and switch the contents of the stack. Let $m_0^\dagger$ be given by:

$$m_0^\dagger(S_1, S_2) = \begin{cases} (\mathrm{psh}(\mathrm{pop}^{\lfloor n \rfloor}(S_1), 1), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle 1^n \rangle\rangle, \\ (\mathrm{psh}(\mathrm{pop}^{\lfloor n \rfloor}(S_1), -1), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle \bar{1}^n \rangle\rangle, \\ (\mathrm{psh}(\mathrm{pop}^{\lfloor n \rfloor}(S_1), 0), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle 0^n \rangle\rangle, \\ (\mathrm{psh}(\mathrm{pop}^{\lfloor n \rfloor}(S_1), \mathrm{tp}(S_1, n-i+1)), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle I_i^n \rangle\rangle, \\ (\mathrm{psh}(\mathrm{pop}^{\lfloor 2 \rfloor}(S_1), \mathrm{tp}(S_1, 2) + \mathrm{top}(S_1)), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle + \rangle\rangle, \\ (\mathrm{psh}(\mathrm{pop}^{\lfloor 2 \rfloor}(S_1), \mathrm{tp}(S_1, 2) \times \mathrm{top}(S_1)), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle \times \rangle\rangle, \\ (\mathrm{psh}(\mathrm{pop}^{\lfloor 2 \rfloor}(S_1), \mathrm{tp}(S_1, 2)/\mathrm{top}(S_1)), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle / \rangle\rangle, \\ (S_1, \mathrm{psh}(\mathrm{psh}(\mathrm{pop}(S_2), \langle\langle h \rangle\rangle), \langle\langle g \rangle\rangle)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle c(h, g) \rangle\rangle, \\ (\mathrm{pop}(S_1), \mathrm{psh}^{\lfloor |\mathrm{top}(S_1)| \rfloor}(\mathrm{pop}(S_2), \langle\langle h \rangle\rangle)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle I(h) \rangle\rangle, \\ (S_1, I_1^3(\mathrm{aggr}^{\lfloor n \rfloor}(S_2, \langle\langle v(f_1, \dots, f_n) \rangle\rangle, n))) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle v(f_1, \dots, f_n) \rangle\rangle, \\ (\mathrm{swt}(S_1, n), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle \mathrm{swt}, n \rangle\rangle, \\ (\mathrm{dup}(S_1, n), \mathrm{pop}(S_2)) & \text{if } \mathrm{top}(S_2) \text{ is } \langle\langle \mathrm{dup}, n \rangle\rangle, \\ (S_1, S_2) & \text{if } \mathrm{emp}(S_2). \end{cases}$$

where for every m-ary description $e = \langle\langle v(f_1, \ldots, f_n)\rangle\rangle$, aggr is such that for $k = n$,

$$\mathrm{aggr}(S, e, k) = (\mathrm{psh}(S, \langle\langle f_n\rangle\rangle), e, k-1),$$

and for $k \neq n$,

$$\mathrm{aggr}(S, e, k) = (\mathrm{psh}(\mathrm{psh}(\mathrm{psh}(S, \langle\langle \mathrm{swt}, m\rangle\rangle), \langle\langle f_k\rangle\rangle), \langle\langle \mathrm{dup}, m\rangle\rangle), e, k-1).$$

By induction on the set of alternative descriptions we conclude that if S_1 is a stack with the real numbers $x_n, \ldots, x_1, y_j, \ldots, y_1$, and S_2 is a stack with the numbers $e, e_1, \ldots e_k$, where e codes the alternative description $\langle\langle \sigma_e^{m,n}\rangle\rangle$ with no infinite limits, then, by iterating $m_0^\dagger$, S_2 will eventually contain only $e_1, \ldots, e_k$ and then we will have m real numbers in the top of S_1, given by each component of $\sigma_e^{m,n}(x_1, \ldots, x_n)$, followed by $y_j, \ldots, y_1$. This is illustrated by Figure 3, where # marks the top of the stacks.

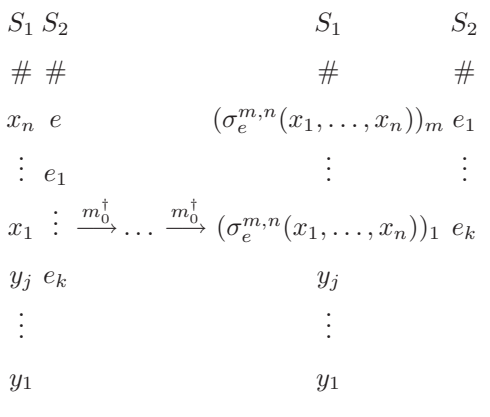

Fig. 3. Stack manipulation by $m_0^\dagger$

Now we set:

$$m_0(\langle\langle\sigma_e\rangle\rangle, x) = \lim_{z\to\infty} \mathrm{top}(I_1^2(m_1^{\dagger\lfloor z\rfloor}(\mathrm{psh}(0, x), \mathrm{psh}(0, \langle\langle\sigma_e\rangle\rangle))))).$$

$m_0 \in \mathrm{H}_4$ is a universal function for alternative descriptions without infinite limits.

Now, for every $k \geqslant 1$, we define $m_k^\dagger$ as

$$m_k^\dagger(S_1, S_2) = \begin{cases} (\mathrm{stlim}_k(S_1, \langle\langle h\rangle\rangle), \mathrm{pop}(S_2)) & \text{if top}(S_2) \text{ is } \langle\langle l(h)\rangle\rangle, \\ (\mathrm{stliminf}_k(S_1, \langle\langle h\rangle\rangle), \mathrm{pop}(S_2)) & \text{if top}(S_2) \text{ is } \langle\langle li(h)\rangle\rangle, \\ (\mathrm{stlimsup}_k(S_1, \langle\langle h\rangle\rangle), \mathrm{pop}(S_2)) & \text{if top}(S_2) \text{ is } \langle\langle ls(h)\rangle\rangle, \\ m_0(S_1, S_2) & \text{otherwise;} \end{cases}$$

and

$$m_k(\langle\langle\sigma_e\rangle\rangle, x) = \lim_{z\to\infty} \mathrm{top}(I_1^2(m_k^{\dagger\lfloor z\rfloor}(\mathrm{psh}(0, x), \mathrm{psh}(0, \langle\langle\sigma_e\rangle\rangle))))).$$

Let us explain the meaning of $\mathrm{stlim}_k, \mathrm{stliminf}_k$ and $\mathrm{stlimsup}_k$. Given a stack S and an alternative description $\langle\langle h\rangle\rangle$ of an n-ary function with m components, stlim_k works on the stack in the following way: for each component i of $\langle\langle h\rangle\rangle$, stlim_k stores on another stack the value

$$\lim_{y\to\infty}\lim_{z\to\infty} \mathrm{top}(I_1^2(m_{k-1}^{\dagger\lfloor z\rfloor}(\mathrm{psh}(S, y), \mathrm{psh}(0, \langle\langle c(I_i^m, h)\rangle\rangle))))).$$

This is similar to taking the limit of m_{k-1} as y goes to ∞, but starting with the stack S. After doing this for all components of h, stlim_k manipulates the top n elements of S and replaces them with the resulting m elements of the other stack. $\mathrm{stlimsup}_k$ and $\mathrm{stliminf}_k$ proceed in the same way, but for $\limsup$ and $\liminf$.

We conclude that stlim_k is in $\mathrm{H}_{E(m_{k-1}^\dagger)+2}$, and so $m_k^\dagger$ is also in $\mathrm{H}_{E(m_{k-1}^\dagger)+2}$, which results in $m_k \in \mathrm{H}_{E(m_k^\dagger)+1} = \mathrm{H}_{E(m_{k-1}^\dagger)+3} = \mathrm{H}_{E(m_{k-1})+2} = \mathrm{H}_{2k+5}$.

Given a description $\langle\phi_e\rangle$ with n nested limits we can, by Proposition 4.9, find an alternative description $\langle\langle\sigma_{e^\dagger}\rangle\rangle$, which describes the same function through the inductive structure of $\mathrm{REC}(\mathbb{R})$ shown in Definition

4.2, with $n+7$ nested limits. We assume by the Church–Turing thesis that some Diophantine function Δ does this, and so, by Proposition 2.13, there is a real recursive extension to Δ. We can finally define Ψ_n as

$$\Psi_n(\langle\phi_e\rangle, x) = m_{n+7}(\Delta(\langle\phi_e\rangle), x),$$

and conclude that Ψ_n is in $\mathrm{H}_{2(n+7)+5} = \mathrm{H}_{2n+19}$. □

7. Conclusions

We conclude that if we have a first order differential equation that gives us some real recursive function, we can always find an infinite limit that describes the same function, using a numerical approximation which asymptotically behaves in the intended manner. Nonetheless, given a function expressed by an infinite limit, we cannot always find a first order differential equation that results in the same function, because if we could, the η-hierarchy would collapse. We have also seen that $\mathrm{REC}(\mathbb{R})$ is closed for Riemann integration and function derivation.

Another result is that there is no universal real recursive Ψ function, but that there are universal real recursive Ψ_n functions for every level of the η-hierarchy. This last result assures that while we cannot have real recursive characteristics for the problems of domain and identity for every real recursive function, we can still have them for every real recursive function up to any level of the η-hierarchy.

One of the authors, Bruno Loff, will shortly submit a paper exposing the close relationship between real recursive functions and the analytical hierarchy. A new proof of the non-collapse of the analytical hierarchy will be shown, not done in the classical manner (see [27] for a classical proof), but as a corollary of the non-collapsing character of the η-hierarchy. The paper thus provides a clue on how analytical methods can be used to prove classical results of computability and complexity theories, even possibly addressing open problems.

However, the main result of this paper — namely the non-collapse of the η-hierarchy — could be of much interest to the analyst. It shows, using a somewhat constructible mathematics, that the taking of infinite limits, as an analytical operation, can always produce new functions which are not solutions of any differential equation of feasible real-valued functions.

It may be an interesting research work to do a systematical analysis of the first three or four levels of the η-hierarchy, with the aim of finding functions with new and interesting properties from the analytical point of view. Although probably not very relevant for our computability-oriented study, a recursive class of distributions (also known as generalised functions) could also be constructed using similar operators.

8. Conventions and notational preferences

The following conventions and notational preferences were used throughout this text.

- We treat infinity as undefinedness, i.e., $+\infty = -\infty = \bot$.
- Functions are partial, according to the conventions of recursion theory. Furthermore, if a function is given an undefined parameter, or results in an undefined component, it is completely undefined, i.e., undefinedness is strict (e.g., $\bot \times 0 = \bot$).
- Infinite limits, supremums, and other standard operators of analysis are given in the standard way. E.g., in symbols,

  $$z = (\mathfrak{sup} f)(\bar{x}, a, b) \iff \forall (y \in [a,b))z \geqslant f(\bar{x}, y) \wedge \forall (t < z)\exists u\ t < f(\bar{x}, u)$$

  So if $f(\bar{x}, y)$ is undefined for some y in $[a, b)$, then the supremum is undefined. Similarly, if $f(\bar{x}, y)$ is undefined for arbitrarily large y, then the limit of $f(\bar{x}, y)$ as $y \to \infty$ is also undefined.
- Real recursive operators can be partial.
- Functions of any arity in $\mathbb{N}$ are considered, where zero-ary functions are also called "numbers" or sometimes "values".
- All operations vectors are considered component-wise. For instance, if h is a vector-valued function with n components, then:

  $$\partial_x h(x) \equiv (\partial_x h_1(x), \partial_x h_2(x), \ldots, \partial_x h_n(x))$$

– Variables with bar accents, like $\bar{x}$, are vector-valued.
– Vector-valued arguments of functions and components of vectors are always expanded, in the sense that if $v(\bar{x}) = (v_1(\bar{x}), v_2(\bar{x}), \ldots, v_m(\bar{x}))$ defines a vector-valued function, then one has, for instance, that

$$f(\bar{x}, v(\bar{x})) \equiv f(x_1, \ldots, x_n, v_1(x_1, \ldots, x_n), v_2(x_1, \ldots, x_n), \ldots, v_m(x_1, \ldots, x_n)),$$

and

$$v(\bar{x}) \equiv (v_1(x_1, \ldots, x_n), v_2(x_1, \ldots, x_n), \ldots, v_m(x_1, \ldots, x_n)).$$

– The down-arrow and up-arrow signs, $\downarrow$ and $\uparrow$, have the usual converges / diverges or defined / undefined meaning.

Acknowledgements

Bruno would like to thank his mother, Isabel Serra, for putting up with his most quixotic ramblings about the universe and Turing machines.

References

[1] V. Bush and H. Hazen. The differential analyser. *Journal of the Franklin Institute*, 212(4):447–488, October 1931.
[2] C. Shannon. Mathematical theory of the differential analyser. *Journal Mathematical Physics*, 20:337–354, 1941.
[3] L. Blum, M. Shub, and S. Smale. On the theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, July 1989.
[4] H. T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*, volume 20 of *Prograss in Theoretical Computer Science*. Birkhäuser, 1997.
[5] C. Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44, August 1996.
[6] M. Campagnolo, C. Moore, and J. F. Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4):642–660, December 2000.
[7] M. Campagnolo, C. Moore, and J. F. Costa. An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 18(4):977–1000, December 2002.
[8] J. Mycka. Infinite limits and R-recursive functions. *Acta Cybernetica*, 16(1):83–91, January 2003.
[9] J. Mycka. μ-recursion and infinite limits. *Theoretical Computer Science*, 302:123–133, June 2003.
[10] J. D. Hamkins and A. Lewis. Infinite time Turing machines. *The Journal of Symbolic Logic*, 65(2):567–604, June 2000.
[11] T. Ord. The many forms of hypercomputation. *Applied mathematics and computation*, 178(1):143–153, July 2006.
[12] Z. Xia. The existence of noncollision singularities in Newtonian systems. *The Annals of Mathematics, Second Series*, 135(3):411–468, May 1992.
[13] W. D. Smith. Church's thesis meets the N-body problem. *Applied Mathematics and Computation*, 178(1):154–183, July 2006.
[14] J. Mycka and J. F. Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835–857, December 2004.
[15] J. Mycka and J. F. Costa. A new conceptual framework for analog computation. *Theoretical Computer Science*, 374:277–290, 2007.
[16] R. L. Pouso. On the cauchy problem for first order discontinuous ordinary differential equations. *Journal of Mathematical Analysis and Applications*, 264(1):230–252, December 2001.
[17] E. R. Hassan and W. Rzymowski. Extremal solutions of a discontinuous scalar differential equation. *Nonlinear Analysis*, 37(8):997–1017, September 1997.

[18] J. Mycka and J. F. Costa. The conjecture $P \neq NP$ presented by means of some classes of real functions. In A. Beckmann, U. Berger, B. Lwe, and J.V. Tucker, editors, *Logical Approaches to Computational Barriers, CiE 2006*, volume 3988 of *Lecture Notes in Computer Science*, pages 47–57, Berlin, July 2006. Springer.

[19] M. Davis. Hilbert's tenth problem is unsolvable. *The American Mathematical Monthly*, 80(3):233–269, March 1973.

[20] M. Campagnolo. The complexity of real recursive functions. In C.S. Calude, M.J. Dinneen, and F. Peper, editors, *Unconventional Models of Computation (UMC'02)*, volume 2509 of *Lecture Notes in Computer Science*, pages 1–14, Berlin, October 2002. Springer.

[21] J. Mycka and J. F. Costa. The $P \neq NP$ conjecture in the context of real and complex analysis. *Journal of Complexity*, 22(2):287–303, April 2006.

[22] D. Graça and J. F. Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664, October 2003.

[23] O. Bournez and E. Hainry. Real recursive functions and real extensions of recursive functions. In Maurice Margenstern, editor, *Machines, Computations and Universality (MCU'2004)*, volume 3354 of *Lecture Notes in Computer Science*, pages 116–127, Berlin, September 2004. Springer.

[24] O. Bournez and E. Hainry. Elementarily computable functions over the real numbers and $\mathbb{R}$-sub-recursive functions. *Theoretical Computer Science*, 348(2–3):130–147, December 2005.

[25] J. Mycka and J. F. Costa. Undecidability over continuous-time. *Logic Journal of the IGPL, Oxford University Press*, 14:649 – 658, October 2006.

[26] M. D. Gladstone. Simplifications of the recursion scheme. *Journal of Symbolic Logic*, 36(4):653–665, December 1971.

[27] P. Odifreddi. *Classical Recursion Theory*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. North Holland, 1989.