Algebraic Characterizations of Complexity-Theoretic Classes of Real Functions

Olivier Bournez, Walid Gomaa, Emmanuel Hainry

Algebraic Characterizations of Complexity-Theoretic Classes of Real Functions

Olivier Bournez[1], Walid Gomaa[2], and Emmanuel Hainry[3]

[1]Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France
[2]Egypt-Japan University of Science and Technology, Alexandria, Egypt
[2]Alexandria University, Faculty of Engineering, Alexandria, Egypt
[3]Loria, BP 239 - 54506 Vandœuvre-lès-Nancy Cedex, France
[3]Nancy Université, Université Henri Poincaré, Nancy, France

Recursive analysis is the most classical approach to model and discuss computations over the real numbers. Recently, it has been shown that computability classes of functions in the sense of recursive analysis can be defined (or characterized) in an algebraic machine independent way, without resorting to Turing machines. In particular nice connections between the class of computable functions (and some of its sub- and sup-classes) over the reals and algebraically defined (sub- and sup-) classes of $\mathbb{R}$-recursive functions à la Moore 96 have been obtained. However, until now, this has been done only at the computability level, and not at the complexity level. In this paper we provide a framework that allows us to dive into the complexity level of real functions. In particular we provide the first algebraic characterization of polynomial-time computable functions over the reals. This framework opens the field of implicit complexity of analog functions, and also provides a new reading of some of the existing characterizations at the computability level.

1 Introduction

Building a well founded theory of computation over the reals is a crucial task. However, computability over the reals is not as well understood as the corresponding notion over discrete objects where the Church-Turing thesis yields a clear equivalence between different computational models. When talking about continuous computation several approaches have been developed with various motivations but without so-clear relationships. Such approaches include the Blum-Shub-Smale (BSS) model [3, 4], Shannon's

General Purpose Analog Computer (GPAC) [26], algebraically defined classes of functions over the reals à la Moore 96 ($\mathbb{R}$-recursive functions) [24], as well as the recursive analysis approach.

Recursive analysis was introduced by Turing [27], Grzegorczyk [17], and Lacombe [22]. It can be considered as the most classical approach to talk about computability and complexity of functions over the real numbers, as its foundations are already present in Alan Turing's 1936 seminal paper. In recursive analysis, a function $f : \mathbb{R} \to \mathbb{R}$ is computable if there exists some computable functional, or Type 2 machine, that maps any sequence of rational numbers quickly converging to x to another sequence quickly converging to $f(x)$.

There is no hope to unify all approaches of continuous computations: for example the BSS models can not be conciliated with the recursive analysis viewpoint, as a non-continuous function can be computed in the BSS framework. However, if we put aside this latter model, which is more motivated by the algebraic complexity of problems rather than being a universal model, some recent works have shown strong connections between recursive analysis, Shannon's GPAC, and $\mathbb{R}$-recursive functions. These results basically state that all these paradigms are more or less equivalent: see [6, 7] or survey [5]. This can be considered somehow as yielding a kind of phenomenon for analog computations like the Church's thesis for discrete computations.

However, up till now discussions have mainly been restricted to the computability level, and not to the complexity level.

Connecting models, known to be related at the computability level, at the complexity level is an even more ambitious goal. An immediate deep problem is that of defining the traditional complexity notions for some of the models such as the GPAC. One reason is that there is no robust and well defined notion of time and space for these models, as shown by several attempts [24, 1, 25, 5].

We show in this paper that it is indeed possible to relate models at the complexity level when restricting to the recursive analysis and $\mathbb{R}$-recursive functions approaches. There is indeed an unambiguous, well developed, and rather well understood theory of complexity in recursive analysis [21]. We relate it to a subclass of $\mathbb{R}$-recursive functions, that is, to a machine-independent algebraically defined class of functions over the reals à la Moore 96 [24].

In particular this paper presents the first algebraic machine-independent characterization of polynomial-time computable functions in the sense of recursive analysis.

We provide as a side effect a whole framework for implicit complexity in recursive analysis that gives a way to relate computability and complexity over the reals to computability and complexity over the integers. We also extend [15], and prove that computable functions over the reals correspond to functions generable by Shannon's GPAC; we extend [7, 9, 6] and prove that computable functions and elementary-time computable functions correspond to natural subclasses of $\mathbb{R}$-recursive functions. In particular, unlike [15, 7, 9, 6], we provide characterizations that work even for non-Lipschitz functions (and that differ slightly for Lipschitz functions). This well founded framework may be a significant step towards a sane computability and complexity theory of functions over the reals.

Potential applications of polynomial-time characterizations include the possibility of proving whether a given function can be computed in polynomial time without resorting to efficiently program it, as well as the possibility of building methods to automatically derive computational properties of programs/systems, in the lines of [18, 19, 23] for discrete programs.

We also believe in the pedagogical value of our characterizations. They yield ways to define computability and complexity over the reals without resorting to any kind of machinery in the spirit of (Type 1 or Type 2) Turing machines. This is a very natural and intuitive paradigm that avoids discrete machinery when talking about continuous computation.

2 Related Work

We prove our results by relating the notion of (polynomial-time) computable functions over the reals to the corresponding notion over the integers. Our setting is actually proved to be robust to approximations: one does not need to be able to compute exactly the corresponding class over the integers, but only some defined approximation of it in order to be able to compute the corresponding class over the reals.

Hence, our framework gives a way to rely on algebraic machine-independent characterizations of computable functions over the integers. Several such characterizations are known [11]: in particular, Kleene's functions are well known to capture exactly the discrete functions computable by Turing machines. Cobham [12], and later Bellantoni and Cook [2], were among the first to propose algebraically defined characterizations of polynomial-time computable discrete functions. Our main theorem relies on Bellantoni and Cook's ideas in [2]. Other machine independent characterizations of classical computability and complexity classes (see survey [11]) over the integers could also be considered.

Notice that our framework is different from the one proposed by Campagnolo and Ojakian in [10]: in particular, it has the main advantage of allowing to talk not only about the computability level but also about the complexity level. It should also be noticed that our characterization relies exclusively on functions over the reals, hence it can not be compared with approaches such as [13] or [20] which explore complexity of type 2 functionals. Algebraic characterizations of functions over more general domains, including the reals, have been obtained in [8]. However, the obtained characterization in this latter paper is rather different to the ones discussed here: on one hand, a more abstract setting that is not restricted to real functions is considered there, but on the other hand the discussion is only restricted to the computability level, and less in the spirit of the above mentioned models of continuous computation.

In this paper, for ease of presentation, we only consider functions defined over compact domains. The constructions described here can indeed be extended to functions over arbitrary domains.

3 Essentials of Recursive Analysis

In this section, we recall some basic definitions from recursive analysis: see [28, 21] for a full detailed presentation. Let $\mathbb{D} = \{\frac{a}{2^b} : \ for\ integers\ a, b\ and\ b \geq 0\}$ be the set of dyadic rationals. These are the rationals with finite binary representation.

Definition 1 *Assume $x \in \mathbb{R}$. A* Cauchy sequence *representing x is a function $\varphi_x : \mathbb{N} \to \mathbb{D}$ that converges at a* binary rate*: $\forall n \in \mathbb{N}: |x - \varphi_x(n)| \leq 2^{-n}$. Given $x \in \mathbb{R}$, let CF_x denote the class of Cauchy functions that represent x.*

Definition 2 (Computability of real functions) *Let f be a function $f : D \subseteq \mathbb{R} \to \mathbb{R}$, where D has only one connected component (in the following discussion we deal almost exclusively with either $D = [0, 1]$ or $D = \mathbb{R}$). We say that f is* computable *if there exists a function-oracle Turing machine $M^{()}$ such that for every $x \in D$, for every $\varphi_x \in CF_x$, and for every $n \in \mathbb{N}$ the following holds: $|M^{\varphi_x}(n) - f(x)| \leq 2^{-n}$.*

If $D = [0, 1]$, then we say f is *polytime computable* if the computation time of $M^{\varphi_x}(n)$ is bounded by $p(n)$ for some polynomial p. In case $D = \mathbb{R}$, we say f is *polytime computable* if the computation time of $M^{\varphi_x}(n)$ is bounded by $p(k, n)$ for some polynomial p where $k = \min\{j : x \in [-2^j, 2^j]\}$.

It is well known that continuity is a necessary condition for real computation, though it is not sufficient. The following definition introduces the notion of 'modulus of continuity' which in some sense quantifies the concept of continuity and provides a useful tool in the investigation of real computation [14].

Definition 3 (Modulus of continuity) *Consider a function $f : \mathbb{R} \to \mathbb{R}$. Then f has a* modulus of continuity *if there exists a function $m : \mathbb{N}^2 \to \mathbb{N}$ such that for all $k, n \in \mathbb{N}$ and for all $x, y \in [-2^k, 2^k]$ the following holds: if $|x - y| \leq 2^{-m(k,n)}$, then $|f(x) - f(y)| \leq 2^{-n}$. If f is defined over $[0, 1]$ the same definition holds except that the parameter k is not necessary anymore, that is $m : \mathbb{N} \to \mathbb{N}$.*

Notice that the existence of a modulus of continuity for a function f implies that this function is continuous. In analogy with [21, corollary 2.14], computability over unbounded domains can be characterized as follows [14].

Proposition 4 *Let a function $f : \mathbb{R} \to \mathbb{R}$. Then f is computable iff there exist two computable functions $m : \mathbb{N}^2 \to \mathbb{N}$ and $\psi : \mathbb{D} \times \mathbb{N} \to \mathbb{D}$ such that*

1. *m is a modulus of continuity for f,*
2. *ψ is an approximation function for f, that is, for every $d \in \mathbb{D}$ and every $n \in \mathbb{N}$ the following holds: $|\psi(d, n) - f(d)| \leq 2^{-n}$.*

When restricting attention to polytime computability two additional requirements need to be added to the previous proposition: (1) the modulus m is a polynomial function, that is $m(k, n) = (k + n)^b$ for some $b \in \mathbb{N}$ and (2) $\psi(d, n)$ is computable in time $p(length(d) + n)$ for some polynomial p, where $length(d)$ is the length of the binary encoding of the number d.

4 Characterizing Polytime Real Complexity over Compact Domains

In this section, we prove that it is possible to relate computability over the reals to computability over the integers. We do it in two steps. In the first step, we consider the special case of Lipschitz functions. In the second step, we discuss how to avoid the Lipschitz hypothesis, and consider general functions.

Without loss of generality we assume that the compact domain is always the unit interval $[0,1]$. Let's first provide a preliminary first result to help explaining what we would like to get.

4.1 A preliminary first result

A real function over a compact interval can be characterized by the discrete projection of a function with domain $[0,1] \times \mathbb{R}$. The extra dimension can be viewed as representing the precision of the computed approximation.

Proposition 5 (Complexity over $[0,1]$ vs Complexity over $[0,1] \times \mathbb{R}$) *The following are equivalent:*

1. *a function $f : [0,1] \to \mathbb{R}$ is polytime computable,*
2. *there exists a polytime computable function $g : [0,1] \times \mathbb{R} \to \mathbb{R}$ such that:*

$$\forall x \in [0,1], \forall y \in \mathbb{N}\colon |g(x,y) - yf(x)| \leq 1. \tag{1}$$

We would like to talk about functions g with assertions like above but quantification is only done over the integers, that is to say about assertions like (1) but with something like $\forall x \in \mathbb{N}$ instead of $\forall x \in [0,1]$.

Moving to such a full integer characterization we are faced with the problem of how the notion of continuity, which is exclusive to real computable functions, can be transferred to the integer domain.

4.2 Lipschitz functions

For Lipschitz functions this is facilitated by the fact that such functions provide us with free information about their continuity properties. A real function $f\colon [0,1] \to \mathbb{R}$ is *Lipschitz* if there exists a constant $K \geq 0$ such that for all $x_1, x_2 \in [0,1]$ the following holds: $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$.

Proposition 6 (Complexity over $[0,1]$ vs Complexity over $\mathbb{R} \times \mathbb{R}$) *Fix an arbitrary constant $\epsilon \geq 0$. Let f be a Lipschitz function on $[0,1]$. Then the following are equivalent:*

1. *f is polytime computable,*

2. *there exists a polytime computable function* $g\colon \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ *such that:*

$$\forall x\in\mathbb{N}, \forall y\in\mathbb{N}^{\geq 1}, x\leq y\colon |g(x,y)-yf(\frac{x}{y})|\leq \epsilon \tag{2}$$

In order to interrelate with discrete complexity classes we suggest to employ the following notion of *approximation.*[1]

Definition 7 (Approximation) *Let* $\mathcal{C}$ *be a class of functions from* $\mathbb{R}^2$ *to* $\mathbb{R}$*. Let* $\mathcal{D}$ *be a class of functions from* $\mathbb{N}^2$ *to* $\mathbb{N}$*. Let* f *be a function defined on* $[0,1]$*.*

1. *We say that* $\mathcal{C}$ approximates $\mathcal{D}$ *if for any function* $g\in\mathcal{D}$*, there exists some function* $\tilde{g}\in\mathcal{C}$ *such that for all* $x,y\in\mathbb{N}$ *we have*

$$|\tilde{g}(x,y)-g(x,y)|\leq 1/4 \tag{3}$$

2. *We say that* f *is* $\mathcal{C}$-definable *if there exists a function* $\tilde{g}\in\mathcal{C}$ *such that the following holds*

$$\forall x\in\mathbb{N}, \forall y\in\mathbb{N}^{\geq 1}, x\leq y\colon |\tilde{g}(x,y)-yf(\frac{x}{y})|\leq 3 \tag{4}$$

We then have the following result:[2]

Theorem 8 (Complexity over $[0,1]$ **vs approximate complexity over** $\mathbb{N}^2$**)** *Consider a class* $\mathcal{C}$ *of polytime computable real functions that approximates the class of polytime computable discrete functions. Assume that* $f:[0,1]\to\mathbb{R}$ *is Lipschitz. Then* f *is polytime computable iff* f *is* $\mathcal{C}$*-definable.*

In the right-to-left direction of the previous, Eq. (4) implicitly provides a way to efficiently approximate f from $\tilde{g}\restriction\mathbb{N}^2$. Computability of f is possible, in particular at the limit points, from the fact that it is Lipschitz (hence continuous), and efficiency is possible by the fact that $\tilde{g}$ is polytime computable. The left-to-right direction relates polytime computability of real functions to the corresponding discrete notion.

4.3 Avoiding the Lipschitz hypothesis

The major obstacle to avoid the Lipschitz hypothesis is how to implicitly encode the continuity of f in discrete computations. This is done in two steps: (1) encoding the modulus of continuity which provides information at arbitrarily small rational intervals (however, it does not tell anything about the limit irrational points) and (2) bounding the behavior of the characterizing function g both at unit intervals and at its integer projection.

We need another notion of 'approximation' that is a kind of converse to that given in Definition 7.

[1] Notice that the choice of the constant $\frac{1}{4}$ in Definition 7 and of the constant 3 in the last theorem is arbitrary.

[2] Note that all these results still hold if we replace 'polytime computable' by just 'computable'.

Definition 9 (Polytime computable integer approximation) *A function $g : \mathbb{R}^d \to \mathbb{R}$ is said to have a* polytime computable integer approximation *if there exists some polytime computable function $h : \mathbb{N}^d \to \mathbb{N}$ with $|h(\bar{x}) - g(\bar{x})| \leq 1$ for all $\bar{x} \in \mathbb{N}^d$.*

A sufficient condition is that the restriction of function g to integers is polytime computable. The choice of the constant 1 is then due to the fact that this is the best estimated error when trying to compute the floor of a real function. Now we define a special class of functions that will be used to implicitly describe information about the smoothness of real functions; its role can be compared to that of the moduli of continuity.

Definition 10 *Consider a function $T\colon \mathbb{N} \to \mathbb{N}$ and define $\#_T : \mathbb{R}^{\geq 1} \to \mathbb{R}$ by $\#_T[x] = 2^{T(\lfloor \log_2 x \rfloor)}$. When T is a polynomial function with $T(x) = \Theta(x^k)$ we write $\#_k$ to simplify the notation, and we let $\#_k[x] = 2^{\lfloor \log_2 x \rfloor^k}$.*

The following proposition is then the non-Lipschitz version of Proposition 6.

Proposition 11 (Complexity over $[0,1]$ vs complexity over $\mathbb{R} \times \mathbb{R}$) *Fix an arbitrary constant $\epsilon \geq 0$. The following are equivalent:*

1. *a function $f : [0,1] \to \mathbb{R}$ is polytime computable,*
2. *there exists some function $g\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that*
   a) *g has a polytime computable integer approximation,*
   b) *for some integer k,*

$$\forall x \in [0,1], \forall y \in \mathbb{R}^{\geq 1}\colon |g(x.\#_k[y], y) - yf(x)| \leq \epsilon, \tag{5}$$

   c) *for some integer M,*

$$\forall x_1, x_2 \in \mathbb{R}^{\geq 0}, y \in \mathbb{R}^{\geq 1}\colon |x_1 - x_2| \leq 1 \Rightarrow |g(x_1, y) - g(x_2, y)| \leq M \tag{6}$$

Proof: (2) $\Rightarrow$ (1) : For simplicity let $\epsilon = 1$. Assume that there exists a function g that satisfies the above conditions. Assume some $x \in [0,1]$ and $n \in \mathbb{N}$. Let $y = 2^n$. From condition (2b) we have

$$\begin{aligned} &|g(2^{n^k}x, 2^n) - 2^n f(x)| \leq 1 \\ &|2^{-n} g(2^{n^k}x, 2^n) - f(x)| \leq 2^{-n} \end{aligned} \tag{7}$$

Let h be some polytime computable discrete function with $|h(x,y) - g(x,y)| \leq 1$ for all $x, y \in \mathbb{N}$ that exists by (2a).

Then

$$|g(\lfloor 2^{n^k}x \rfloor, 2^n) - h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq 1 \tag{8}$$

From (2c) we have

$$|g(\lfloor 2^{n^k}x \rfloor, 2^n) - g(2^{n^k}x, 2^n)| \leq M \tag{9}$$

From the previous two equations

$$|g(2^{n^k}x, 2^n) - h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq M + 1$$
$$|2^{-n}g(2^{n^k}x, 2^n) - 2^{-n}h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq 2^{-n}(M+1) \tag{10}$$

From Equations (7) and (10)

$$|f(x) - 2^{-n}h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq 2^{-n}(M+2) \tag{11}$$

This last equation *characterizes the computation of the real function f by the computation of the integer function h.* Furthermore, it provides information about the precision of the approximation. We can build a polytime oracle Turing machine that computes f as follows. Assume some $\varphi \in CF_x$. Consider a machine $M^{\varphi}(n)$ that does the following:

1. let $d = \varphi(n^k + 1)$,
2. let $j = h(\lfloor 2^{n^k}d \rfloor, 2^n)$,
3. output $2^{-n}j$.

Since h is polytime computable, $M^{\varphi}(n)$ operates in polynomial time with respect to the precision parameter n. Verifying the correctness of $M^{\varphi}(n)$ we have

$$|d - x| \leq 2^{-(n^k+1)}$$
$$|2^{n^k}d - 2^{n^k}x| \leq 1/2$$
$$|\lfloor 2^{n^k}d \rfloor - \lfloor 2^{n^k}x \rfloor| \leq 1$$

Then by 2c

$$|g(\lfloor 2^{n^k}d \rfloor, 2^n) - g(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq M \tag{12}$$

By 2a and the choice of h

$$|g(\lfloor 2^{n^k}d \rfloor, 2^n) - h(\lfloor 2^{n^k}d \rfloor, 2^n)| \leq 1 \tag{13}$$

From Equations 8, 12, and 13 we have

$$|h(\lfloor 2^{n^k}d \rfloor, 2^n) - h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq M + 2$$
$$|2^{-n}h(\lfloor 2^{n^k}d \rfloor, 2^n) - 2^{-n}h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq 2^{-n}(M+2) \tag{14}$$

From the last equation and Eq. 11 we have the required conclusion

$$|f(x) - 2^{-n}h(\lfloor 2^{n^k}d \rfloor, 2^n)| \leq 2^{-n}(2M+4) \tag{15}$$

(1) ⇒ (2) : Assume that $f\colon [0,1] \to \mathbb{R}$ is polytime computable. Hence f has a polynomial modulus $m(n) = n^k$ for some constant $k \in \mathbb{N}$. Define g as follows:

$$g(x,y) = \begin{cases} yf(\frac{x}{\#_k[y]}) & x \geq 0, y \geq 1, x \leq \#_k[y] \\ yf(1) & x \geq 0, y \geq 1, x \geq \#_k[y] \\ yf(0) & ow \end{cases} \tag{16}$$

Then for every $x \in [0,1]$ and $y \in \mathbb{R}^{\geq 1}$ we have

$$|g(x.\#_k[y], y) - yf(x)| = 0 \leq 1,$$

hence condition (2b) is satisfied. Now assume $x_1, x_2 \in \mathbb{R}^{\geq 0}, y \in \mathbb{R}^{\geq 1}$ such that $|x_1 - x_2| \leq 1$. There are three cases.
<u>case 1:</u> $x_1 \leq \#_k[y]$ and $x_2 \leq \#_k[y]$, then

$$\begin{aligned} |g(x_1,y) - g(x_2,y)| &= |yf(\frac{x_1}{\#_k[y]}) - yf(\frac{x_2}{\#_k[y]})| \\ &= y|f(\frac{x_1}{\#_k[y]}) - f(\frac{x_2}{\#_k[y]})| \end{aligned}$$

We have $|\frac{x_1}{\#_k[y]} - \frac{x_2}{\#_k[y]}| = \frac{1}{\#_k[y]}|x_1 - x_2| \leq \frac{1}{\#_k[y]} = 2^{-(\log_2 y)^k}$. Hence, using the modulus of continuity of f, $|f(\frac{x_1}{\#_k[y]}) - f(\frac{x_2}{\#_k[y]})| \leq 2^{-\log_2 y} = \frac{1}{y}$ implying $|g(x_1,y) - g(x_2,y)| \leq 1$ and condition (2c) is satisfied with $M = 1$.
<u>case 2:</u> $x_1 \geq \#_k[y]$ and $x_2 \geq \#_k[y]$, then

$$|g(x_1,y) - g(x_2,y)| = |yf(1) - yf(1)| = 0$$

and condition (2c) is also satisfied with $M = 1$.
<u>case 3:</u> $x_1 \leq \#_k[y]$ and $x_2 \geq \#_k[y]$, then

$$\begin{aligned} |g(x_1,y) - g(x_2,y)| &\leq |g(x_1,y) - g(\#_k[y],y)| + |g(\#_k[y],y) - g(x_2,y)| \\ &\leq 1 + 0 = 1 \end{aligned}$$

by the above two cases, and hence condition (2c) is also satisfied with $M = 1$.

Note that division, multiplication, $\#_k$, and f are all computable. Hence, g is computable. Assume $i, j \in \mathbb{N}$ such that $i \leq \#_k[j]$. Then $g(i,j) = jf(\frac{i}{\#_k[j]})$. The computation of $\lfloor g(i,j) \rfloor$ involves the following steps:

1. Let b be the length of the binary representation of j.
2. Shift right the binary representation of i by $(b-1)^k$ positions. The result would be a dyadic rational d.
3. Simulate the computation of $f(d)$ assuming large enough precision that is at least the length of d. When simulating the oracle, d is presented exactly as an answer to any oracle query.

4. Multiply the output of the previous step by j. Finally, truncate the result to extract the integer part.

All of these steps can be performed in polynomial time in terms of the lengths of i and j. Since the output from step 3 is an approximation there is a possibility that the floor is computed with an error that can be bounded by 1. The case when $i > \#_k[j]$ is similar. Hence, Condition (2a) is satisfied. □

We need to consider real functions that are well behaved relative to their restriction to $\mathbb{N}^2$. For ease of notation, we will use $[a, b]$ to denote $[a, b]$ or $[b, a]$, according to whether $a < b$ or the countrary.

Definition 12 (Peaceful functions) *A function $g : \mathbb{R}^2 \to \mathbb{R}$ is said to be* peaceful *if $\forall x \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{N}$: $g(x, y) \in [g(\lfloor x \rfloor, y), g(\lceil x \rceil, y)]$. We say that a class $\mathcal{C}$ of real functions* peacefully approximates *some class $\mathcal{D}$ of integer functions, if the subclass of peaceful functions of $\mathcal{C}$ approximates $\mathcal{D}$.*

Definition 13 *Let $\mathcal{C}$ be a class of functions from $\mathbb{R}^2$ to $\mathbb{R}$. Let us consider a function $f : [0, 1] \to \mathbb{R}$ and a function $T \colon \mathbb{N} \to \mathbb{N}$.*

1. *We say that f is T-$\mathcal{C}$-*definable *if there exists some peaceful function $g \in \mathcal{C}$ such that*
$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_T[y] \colon |g(x, y) - yf(\frac{x}{\#_T[y]})| \leq 2, \tag{17}$$
2. *We say that f is T-*smooth *if there exists some integer M such that*
$$\forall x, x' \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{R}^{\geq 1}, x, x' \leq \#_T[y] \colon \\ |x - x'| \leq 1 \Rightarrow y|f(\frac{x}{\#_T[y]}) - f(\frac{x'}{\#_T[y]})| \leq M \tag{18}$$

Notice the similarity in the role that $\#_T[y]$ plays in the previous definition and as a modulus of continuity for f. Now we can have the non-Lipschitz version of Theorem 8.

Theorem 14 *(Complexity over $[0, 1]$ vs approximate complexity over $\mathbb{N}^2$) Consider a class $\mathcal{C}$ of real functions that peacefully approximates polytime computable discrete functions, and whose functions have polytime computable integer approximations.[3] Then the following are equivalent:*

1. *a function $f \colon [0, 1] \to \mathbb{R}$ is polytime computable,*
2. *there exists some integer k such that*
   a) *f is n^k-$\mathcal{C}$-definable,*
   b) *f is n^k-smooth.*

[3] A sufficient condition for that is restrictions to integers of functions from $\mathcal{C}$ are polytime computable.

Proof: (1) $\Rightarrow$ (2) : Let $f\colon [0,1] \to \mathbb{R}$ be a polytime polytime computable function. By Proposition 11 for $\epsilon = 3/4$, there exists some function g with a polytime computable integer approximation h such that (5) holds. Now, by the hypothesis of this theorem, there exists some peaceful $\tilde{h} \in \mathcal{C}$ such that $\forall x, y \in \mathbb{N} : |\tilde{h}(x,y) - h(x,y)| \leq 1/4$. Hence $\forall x, y \in \mathbb{N}\colon |\tilde{h}(x,y) - g(x,y)| \leq 1 + \frac{1}{4} = \frac{5}{4}$.

Finally, we have[4] (through change of variables in Eq. (5) and restricting the domains of the variables to $\mathbb{N}$)

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_k[y]\colon |\tilde{h}(x,y) - yf(\frac{x}{\#_k[y]})| \leq \frac{5}{4} + \frac{3}{4} = 2 \tag{19}$$

Hence, condition 2a holds. Now, by (2c) of Proposition 11, we know that for all $x \in \mathbb{R}^{\geq 0}$, $y \in \mathbb{R}^{\geq 1}$, and $\delta \in [0,1]$: $|g(x+\delta, y) - g(x,y)| \leq M$ for some integer M. Then by using Eq. (5) (after variable change and renaming), condition (2b) is satisfied.

(2) $\Rightarrow$ (1) : Let $g \in \mathcal{C}$ be a peaceful function that n^k-$\mathcal{C}$-defines f. Proof is by applying Proposition 11 as follows. From the hypothesis of this theorem g has a polytime computable integer approximation, hence condition 2a of Proposition 11 is satisfied. Condition 2a of the current theorem is equivalent to condition 2b of Proposition 11 by: (1) letting $\epsilon = 2$, (2) renaming of the variables, and (3) observing that the proof of Proposition 11 can be easily adapted to a new version of condition 2b for which x and y take only integer values. Using the fact that g is peaceful (controlling the behavior of g between integer points) condition (2c) of Proposition 11 can be easily verified. □

The previous theorem can be generalized to any complexity class as indicated by the following corollary.

Corollary 15 *Let $\mathcal{D}$ be a class of time-constructive functions from $\mathbb{N}$ to $\mathbb{N}$ that includes polynomial functions and closed under composition. Consider a class $\mathcal{C}$ of functions that peacefully approximate the class of discrete functions computable in time $\mathcal{D}$; and whose functions have integer approximations computable in time $\mathcal{D}$.* [5] *Then the following are equivalent:*

1. *a function $f\colon [0,1] \to \mathbb{R}$ is computable in time $\mathcal{D}$,*
2. *there exists some $T \in \mathcal{D}$ such that*
   a) *f is T-$\mathcal{C}$-definable,*
   b) *f is T-smooth.*

Proof: The proof is similar to that of the previous theorem. It should be noted that if f is computable in time bounded by $\mathcal{D}$ then it has a modulus in $\mathcal{D}$. This is a direct consequence of [21, Theorem 2.19]. □

[4] Note that all these results still hold if we replace 'polytime computable' by just 'computable'.

[5] A sufficient condition for that is restrictions to integers of functions from $\mathcal{C}$ are computable in time $\mathcal{D}$.

5 Applications

In this section we apply the above results to algebraically characterize some computability and complexity classes of real functions. We first obtain some restatements and extensions of already known results, using our framework. We then provide new results, in particular, the main result given by theorems 27 and 28 which provide algebraic machine independent characterizations of polynomial time computable functions.

5.1 GPAC-generable functions

The General Purpose Analog Computer, introduced by Claude Shannon in [26] to model a mechanical device, can be seen in a modern perspective as what can be computed using analog electronics. It consists of circuits interconnecting basic blocks that can be constants, adders, multipliers, and integrators. GPAC-computable functions have been characterized in different ways since the introduction of the model. In the following we will use Graça and Costa's characterization by PIVP (Polynomial Initial Value Problems) [16]. A function is said to be PIVP if it is a component of the solution of a differential equation of the following form:

$$\begin{cases} y(t_0) &= y_0 \\ y'(t) &= p(t, y) \end{cases}$$

with $y : \mathbb{R}^n \to \mathbb{R}$ and p is a vector of polynomial functions. The next lemma follows from the constructions in [15]:

Lemma 16 *PIVP functions is a class of computable functions that peacefully approximate total (discrete) recursive functions.*

Then the following result follows directly from Theorem 8 (and Footnote 2), and from Corollary 15 (and Footnote 4).

Proposition 17 (Variation of [6]) *A Lipschitz function $f : [0, 1] \to \mathbb{R}$ is computable iff it is PIVP-definable.*

Proposition 18 (Extension of [6]) *Let $f : [0, 1] \to \mathbb{R}$ be some T-smooth function, for some total recursive function $T : \mathbb{N} \to \mathbb{N}$. Then f is computable iff it is T-PIVP-definable.*

5.2 Particular classes of $\mathbb{R}$-recursive functions

A function algebra $\mathcal{F} = [\mathcal{B}; \mathcal{O}]$ is the smallest class of functions containing a set of basic functions $\mathcal{B}$ and their closure under a set of operations $\mathcal{O}$.

5.2.1 Elementarily computable functions: class $\mathcal{L}$

Let us now consider the class $\mathcal{L}$ defined in [9]: $\mathcal{L} = [0, 1, -1, \pi, U, \theta_3; COMP, LI]$, where π is the mathematical constant $\pi = 3.14..$, U is the set of projection functions, $\theta_3(x) = \max\{0, x^3\}$, $COMP$ is the classical composition operation, LI is Linear Integration. From the constructions of [9], we know that this class captures the discrete elementary functions. In addition the following lemma follows from the constructions in [7].

Lemma 19 *$\mathcal{L}$ is a class of real functions computable in elementary time that peacefully approximates total discrete elementarily computable functions.*

Again using the above results we can obtain characterizations of the class of elementarily computable analysis functions:

Proposition 20 (Variation of [9]) *A Lipschitz function $f : [0, 1] \to \mathbb{R}$ is computable in elementary time iff it is $\mathcal{L}$-definable.*

Proposition 21 (Extension of [9]) *Let $f : [0, 1] \to \mathbb{R}$ be some T-smooth function, for some elementary function $T : \mathbb{N} \to \mathbb{N}$. Then f is computable in elementary time iff it is T-$\mathcal{L}$-definable.*

As in [9, 7], we can also characterize in a similar way the functions computable in time $\mathcal{E}_n$ for $n \geq 3$, where $\mathcal{E}_n$ represents the n-th level of the Grzegorczyk hierarchy.

5.2.2 Recursive functions: class $\mathcal{L}_\mu$

Let us now consider the class $\mathcal{L}_\mu$ defined in [7]: $\mathcal{L}_\mu = [0, 1, U, \theta_3; COMP, LI, UMU]$, where a zero-finding operator UMU has been added. This class is known (see [7]) to extend the class of total (discrete) recursive functions; from the constructions in this latter paper one can show:

Lemma 22 *$\mathcal{L}_\mu$ is a class of computable functions that peacefully approximate the class of total discrete recursive functions.*

And hence, as a consequence of Theorem 8 and Corollary 15, we obtain:

Proposition 23 (Variation of [7]) *A Lipschitz function $f : [0, 1] \to \mathbb{R}$ is computable iff it is $\mathcal{L}_\mu$-definable.*

Proposition 24 (Extension of [7]) *Let $f : [0, 1] \to \mathbb{R}$ be some T-smooth function, for some total recursive function $T : \mathbb{N} \to \mathbb{N}$. Then f is computable iff it is T-$\mathcal{L}_\mu$-definable.*

5.3 Main result: polytime computable functions

We are now ready to provide our main result: an algebraic characterization of polytime computable functions over the reals.

To do so, we define a class of real functions which are essentially extensions to $\mathbb{R}$ of the Bellantoni-Cook class [2]. This latter class was developed to exactly capture discrete polytime computability in an algebraic machine-independent way. In the next definition any function $f(x_1, \ldots, x_m; y_1, \ldots, y_n)$ has two types of arguments (see [2]): *normal* arguments which come first followed by *safe* arguments using ';' for separation. For any $n \in \mathbb{Z}$ we call $[2n, 2n+1]$ an even interval and $[2n+1, 2n+2]$ an odd interval.

Definition 25 *Define the function algebra*

$$\mathcal{W} = [0, 1, +, -, U, p, c, parity; SComp, SI]$$

1. *zero-ary functions for the constants* 0 *and* 1,
2. *a binary addition function:* $+(; x, y) = x + y$,
3. *a binary subtraction function:* $-(; x, y) = x - y$,
4. *a set of projection functions* $U = \{U_i^j : i, j \in \mathbb{N}, i \leq j\}$ *where:* $U_i^{m+n}(x_1, \ldots, x_m; x_{m+1}, \ldots, x_{m+n}) = x_i$,
5. *a polynomial conditional*[6] *function* c *defined by:* $c(; x, y, z) = xy + (1 - x)z$.
6. *a continuous parity function:* $parity(; x) = \max(0, 2/\pi sin(\pi x))$.

   Hence, $parity(; x)$ *is non-zero if and only if* x *lies inside an even interval. Furthermore, for any* $n \in \mathbb{Z}$ *the following holds:* $\int_{2n}^{2n+1} parity(; x)dx = 1$.
7. *a continuous predecessor function* p *defined by:* $p(; x) = \int_0^{x-1} parity(; t)dt$. *Note that when* x *belongs to an even interval* $p(; x)$ *acts exactly like* $\lfloor \frac{x}{2} \rfloor$. *On an odd interval* $[2n+1, 2n+2]$, *it grows continuously from* n *to* $n+1$.
8. *a safe composition operator SComp: given a vector of functions* $\bar{g}_1(\bar{x};) \in \mathcal{W}$, *a vector of functions* $\bar{g}_2(\bar{x}; \bar{y}) \in \mathcal{W}$, *and a function* $h \in \mathcal{W}$ *of arity* $len(\bar{g}_1) + len(\bar{g}_2)$ *(where len denotes the vector length). Define new function*

$$f(\bar{x}; \bar{y}) = h(\bar{g}_1(\bar{x};); \bar{g}_2(\bar{x}; \bar{y})) \tag{20}$$

   It is clear from the asymmetry in this definition that normal arguments can be repositioned in safe places whereas the opposite can not happen.

[6] If $x = 1$, the conditional is equal to y; if $x = 0$, it is equal to z. Between 0 and 1, it stays between y and z.

9. *safe integration operator*[7] *SI: given functions* $g, h_0, h_1 \in \mathcal{W}$. *Let* $p'(;x) = p(;x-1)+1$. *Define a new function solution of the ODE:*

$$\begin{aligned} f(0,\bar{y};\bar{z}) =& g(\bar{y};\bar{z}) \\ \partial_x f(x,\bar{y};\bar{z}) =& \, parity(x;)[h_1(p(x;),\bar{y};\bar{z}, f(p(x;),\bar{y};\bar{z})) \\ & - f(2p(x;),\bar{y};\bar{z})] \\ & + parity(x-1;)[h_0(p'(x;),\bar{y};\bar{z}, f(p'(x;),\bar{y};\bar{z})) \\ & - f(2p'(x;)-1,\bar{y};\bar{z})] \end{aligned} \tag{21}$$

This operator closely matches Bellantoni and Cook's predicative recursion on notations: if x *belongs to an even interval, we apply* h_0 *to its predecessor* $p(x;)$*; if* x *belongs to an odd interval, we apply* h_1 *to* $p'(x;) = \lfloor x/2 \rfloor$.

This class $\mathcal{W}$ is based on the Bellantoni-Cook's constructions and normal/safe arguments ideas in order to have the following properties, proved by induction.

Proposition 26 *1. Class* $\mathcal{W}$ *preserves the integers, that is for every* $f \in \mathcal{W}$, $f \restriction \mathbb{Z}\colon \mathbb{Z} \to \mathbb{Z}$.

2. Every polytime computable discrete function has a peaceful extension in $\mathcal{W}$.

3. Every function in $\mathcal{W}$ *is polytime computable.*

Proof: Part I: Proof is by induction on the construction of functions in $\mathcal{W}$. It is easy to see that the constant functions 0 and 1, addition, subtraction, and projections all preserve $\mathbb{Z}$. Given $n \in \mathbb{Z}$ we have $p(;n) = \lfloor \frac{n}{2} \rfloor$ which is an integer. Given $i, j, k \in \mathbb{Z}$ we have $c(;i,j,k)$ equals either j or k depending on the value of i, hence the conditional function preserves $\mathbb{Z}$. The parity function is always 0 at the integer points. Hence, all basic functions preserve the integers. Trivially composition preserves the integers. Let $g, h_0, h_1 \in \mathcal{W}$ be functions that preserve $\mathbb{Z}$ and consider the application of the safe integration operator to define a new function $f \in \mathcal{W}$. We use strong induction over the discrete values of the integration variable to show f preserves $\mathbb{N}$ (for simplicity we restrict to non-negative integers; also we neglect the arguments $\bar{y}$ and $\bar{z}$ and drop the ';' in case all arguments of the function are normal). The base case $f(0) = g$ holds by assumption on g. Let $n \in \mathbb{N}^{\geq 1}$ and assume $f(j) \in \mathbb{N}$ for every $j \leq 2n$, then

[7] Notice also that for simplicity we misuse the basic functions (and p') so that their arguments are now in normal positions (the alternative is to redefine a new set of basic functions with arguments in normal positions).

$$
\begin{aligned}
f(2n+1) &= f(2n) + \int_{2n}^{2n+1} parity(x)[h_1(p(x); f(p(x))) - f(2p(x))]dx \\
&= f(2n) + \int_{2n}^{2n+1} parity(x)[h_1(n; f(n)) - f(2n)]dx \\
&= f(2n) + [h_1(n; f(n)) - f(2n)] \int_{2n}^{2n+1} parity(x)dx \\
&= f(2n) + [h_1(n; f(n)) - f(2n)] \cdot 1 \\
&= h_1(n; f(n))
\end{aligned} \tag{22}
$$

which is an integer value by the assumption on h_1 and the induction hypothesis on f. Similarly, it can be shown that $f(2n+2) = h_0(n+1; f(n+1))$ which is also an integer value. This completes the proof of the first part of the proposition.

Part II: First we use induction to show that every function in the Bellantoni-Cook class (which captures discrete polytime computability) has an extension in $\mathcal{W}$. The Bellantoni-Cook class is defined by: $B = [0, U, s_0, s_1, pr, cond; SComp, SRec]$, see [2].

The functions $0, U \in \mathcal{W}$ are extensions of the corresponding functions in B. Define functions $\tilde{s}_i \in \mathcal{W}$ by $\tilde{s}_i(; x) = 2x + i$ where $i \in \{0, 1\}$. Then $\tilde{s}_i$ are extensions of the successor funtions s_i. We have $p \restriction \mathbb{N} = pr$, hence p is an extension of the predecessor function pr. Define a function $c_d \in \mathcal{W}$ as follows.

$$
c_d(; x, y, z) = c(; x - 2p(; x), z, y) \tag{23}
$$

Assume $x = 2n$ for $n \in \mathbb{N}$. Then $c_d(; 2n, y, z) = c(; 2n - 2p(; 2n), z, y) = c(; 0, z, y) = y$. Now assume $x = 2n+1$. Then $c_d(; 2n+1, y, z) = c(; 2n+1-2p(; 2n+1), z, y) = c(; 2n+1-2n, z, y) = c(; 1, z, y) = z$. So $c_d \restriction \mathbb{N}^3 = cond$, hence it is an extension of the conditional function $cond$. The case for safe composition $SComp$ is easy. Now assume $f \in B$ that is defined by safe recursion from $g, h_0, h_1 \in B$. Assume $\tilde{g}, \tilde{h}_0, \tilde{h}_1 \in \mathcal{W}$ are extensions of g, h_0, h_1. Define the function $\tilde{f} \in \mathcal{W}$ by safe integration from $\tilde{g}, \tilde{h}_0, \tilde{h}_1$. We claim that $\tilde{f}$ is an extension of f. Proof is by strong induction on the recursion/integration variable. At the base case we have $\tilde{f}(0) = \tilde{g} = g = f(0)$. Let $n \in \mathbb{N}$, then from the proof of the first part of the proposition we have:

$$
\begin{aligned}
\tilde{f}(2n+1) &= \tilde{h}_1(n; \tilde{f}(n)) = \tilde{h}_1(n; f(n)), \qquad \textit{from induction over } n \\
&= h_1(n; f(n)), \quad \textit{by assumption on } \tilde{h}_1 \\
&= f(2n+1), \quad \textit{by definition of safe recursion}
\end{aligned} \tag{24}
$$

Similarly, it can be shown that $\tilde{f}(2n+2) = f(2n+2)$, hence $\tilde{f}$ is an extension of f. We have shown that every function in B has an extension in $\mathcal{W}$. It now remains to show that we can find a peaceful extension inside $\mathcal{W}$. Since $\int_n^{n+1} parity(x)dx = 1$ for any $n \in \mathbb{N}$, it is easy to see that every function generated by the safe integration operator is

peaceful. Now consider an arbitrary function $f \in \mathcal{W}$. Define the following functions in $\mathcal{W}$:

$$\begin{aligned} \hat{g}() &= f(0) \\ \hat{h}_0(x;y) &= f(2x) \\ \hat{h}_1(x;y) &= f(2x+1) \end{aligned} \tag{25}$$

Now define a function $\hat{f} \in \mathcal{W}$ by safe integration using the functions $\hat{g}, \hat{h}_1$, and $\hat{h}_2$. It can be easily seen that $\hat{f}(n) = f(n)$ for every $n \in \mathbb{N}$. In addition $\hat{f}$ is peaceful. This completes the proof of the second part of the proposition.

Part III: Proof is by induction on the construction of functions in $\mathcal{W}$. It is easy to see that all the basic functions are polytime computable. Composition preserves polytime computability. Consider a function $f \in \mathcal{W}$ that is defined by safe integration from g, h_0, h_1 where these latter functions are polytime computable. At $x = 0$ we have $f(0) = g$ which is polytime computable by assumption on g. Assume $x \in [2n, 2n+1]$ for some $n \in \mathbb{N}$, then

$$\begin{aligned} f(x) &= g + \int_0^x parity(u)[h_1(p(u); f(p(u))) - f(2p(u))]du \\ &= f(2n) + \int_{2n}^x parity(u)[h_1(n; f(n)) - f(2n)]du \\ &= f(2n) + [h_1(n; f(n)) - f(2n)] \int_{2n}^x parity(u)du \\ &= f(2n) + [f(2n+1) - f(2n)] \int_{2n}^x parity(u)du, \quad \textit{from above discussion} \\ &= f(2n) + [f(2n+1) - f(2n)](p(x) - p(2n)) \\ &= f(2n) + [f(2n+1) - f(2n)](p(x) - \lfloor \frac{x}{2} \rfloor) \end{aligned} \tag{26}$$

Notice that $p(x)$ is a polytime computable function, and $\epsilon_x = p(x) - \lfloor \frac{x}{2} \rfloor$ belongs to $[0, 1]$.

$$f(x) = f(2n) + \epsilon_x(f(2n+1) - f(2n)) \tag{27}$$

Similarly, over odd intervals $[2n+1, 2n+2]$ we have

$$f(x) = f(2n+1) + \epsilon_x(f(2n+2) - f(2n+1)) \tag{28}$$

At the integer points the safe integration operator exactly simulates the behavior of the safe recursion operator, and given that h_0 and h_1 are polytime computable we can

reach the conclusion that f is too polytime computable. This completes the proof of the third part of the proposition. □

The proposition indicates that $\mathcal{W}$ is a class of polytime computable real functions that approximates polytime computable discrete functions. Hence, using Theorem 8 the following result is obtained.

Theorem 27 *A Lipschitz function $f : [0,1] \to \mathbb{R}$ is polytime computable iff it is $\mathcal{W}$-definable.*

Additionally, the previous proposition implies that any function in $\mathcal{W}$ has polytime computable integer approximation, hence using Corollary 15, we can get the following result.

Theorem 28 *Let $f : [0,1] \to \mathbb{R}$ be some n^k-smooth function for some k. Then f is polytime computable iff it is n^k-$\mathcal{W}$-definable.*

Notice that $\mathcal{C}$-definability of a function can be seen as a schema that builds a function f from a function $\tilde{g} \in \mathcal{C}$ (see definition of $\mathcal{C}$-definability). Hence, the class of polytime computable functions can be algebraically characterized in a machine-independent way as follows.

Corollary 29 *Let $Def[\mathcal{C}]$ stands for $\mathcal{C}$-definability. Then a function $f\colon [0,1] \to \mathbb{R}$ is polytime computable iff either (1) f is Lipschitz and belongs to*

$$Def[0, 1, +, -, U, p, c, parity; SComp, SI]$$

or (2) f is n^k-smooth and belongs to

$$n^k\text{-}Def[0, 1, +, -, U, p, c, parity; SComp, SI].$$

Remark 30 *It follows from our constructions that one could have put $*(;x,y) = xy$ as a basic function, from which $c(;x,y,z) = +(;*(;x,y),*(;-(;1,x),z))$ would be definable. In the same spirit adding π, $1/\pi$, $\sin(;x) = \sin(x)$, and $\max(;x,y) = max(x,y)$ would yield $parity(;x)$.*

Remark 31 *$parity(;x)$ can be replaced by any function with above mentioned properties.*

References

[1] Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoret. Comput. Sci.*, 138(1):35–65, 1995.

[2] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Comput. Complexity*, 2:97–110, 1992.

[3] Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. *Complexity and Real Computation*. Springer, 1998.

[4] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21(1):1–46, 1989.

[5] Olivier Bournez and Manuel L. Campagnolo. A survey on continuous time computations. In S. B. Cooper, B. Löwe, and A. Sorbi, editors, *New Computational Paradigms*, pages 383–423. Springer, New York, 2008.

[6] Olivier Bournez, Manuel L. Campagnolo, Daniel S. Graça, and Emmanuel Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *J. Complexity*, 23(3):317–335, 2007.

[7] Olivier Bournez and Emmanuel Hainry. Recursive analysis characterized as a class of real recursive functions. *Fund. Inform.*, 74(4):409–433, 2006.

[8] Vasco Brattka. Computability over topological structures. In S. B. Cooper and S. Goncharov, editors, *Computability and Models*, pages 93–136. Kluwer Academic Publishers, New York, 2003.

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

[10] Manuel L. Campagnolo and Kerry Ojakian. The methods of approximation and lifting in real computation. In *Computability and Complexity in Analysis (CCA 2006)*, volume 167 of *Electron. Notes Theor. Comput. Sci.*, pages 387–423, 2007.

[11] Peter Clote. Computational models and function algebras. In E. R. Griffor, editor, *Handbook of Computability Theory*, pages 589–681. North-Holland, Amsterdam, 1998.

[12] Alan Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1965.

[13] Robert Constable. Type two computational complexity. In *Proc. fifth annual ACM symposium on Theory of computing*, pages 108–121, 1973.

[14] Walid Gomaa. Characterizing polynomial time computability of rational and real functions. In S. B. Cooper and V. Danos, editors, *Proceedings of DCM 2009*, volume 9 of *Electron. Proc. Theor. Comput. Sci.*, pages 54–64, 2009.

[15] Daniel S. Graça, Manuel L. Campagnolo, and Jorge Buescu. Robust simulations of Turing machines with analytic maps and flows. In S. B. Cooper, B. Löwe, and L. Torenvliet, editors, *CiE 2005: New Computational Paradigms*, volume 3526 of *Lecture Notes in Comput. Sci.*, pages 169–179. Springer, 2005.

[16] Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. *J. Complexity*, 19(5):644–664, 2003.

[17] Andrzej Grzegorczyk. On the definitions of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.

[18] Martin Hofmann. Type systems for polynomial-time computation, 1999. Habilitation thesis.

[19] Neil D. Jones. The expressive power of higher-order types or, life without CONS. *J. Funct. Programming*, 11(1):5–94, 2001.

[20] Bruce M. Kapron and Stephen A. Cook. A new characterization of type-2 feasibility. *SIAM J. Comput.*, 25(1):117–132, 1996.

[21] Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991.

[22] Daniel Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. *C. R. Acad. Sci. Paris*, 241:151–153, 1955.

[23] Jean-Yves Marion and Jean-Yves Moyen. Efficient first order functional program interpreter with time bound certifications. In *LPAR*, volume 1955 of *Lecture Notes in Comput. Sci.*, pages 25–42. Springer, Nov 2000.

[24] Cristopher Moore. Recursion theory on the reals and continuous-time computation. *Theoret. Comput. Sci.*, 162(1):23–44, 1996.

[25] Keijo Ruohonen. Event detection for ODEs and nonrecursive hierarchies. In *Proceedings of the Colloquium in Honor of Arto Salomaa. Results and Trends in Theoretical Computer Science (Graz, Austria, June 10-11, 1994)*, volume 812 of *Lecture Notes in Comput. Sci.*, pages 358–371. Springer, Berlin, 1994.

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

[27] Alan. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936.

[28] Klaus Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.