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] AND WALID GOMAA [2,3] AND EMMANUEL HAINRY [2,4]

[1] Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France
E-mail address: Olivier.Bournez@lix.polytechnique.fr

[2] Loria, BP 239 - 54506 Vandœuvre-lès-Nancy Cedex, France

[3] Alexandria University, Faculty of Engineering, Alexandria, Egypt
E-mail address: walid.gomaa@loria.fr

[4] Nancy Université, Université Henri Poincaré, Nancy, France
E-mail address: Emmanuel.Hainry@loria.fr

Abstract. Recursive analysis is the most classical approach to model and discuss computations over the reals. It is usually presented using Type 2 or higher order Turing machines. Recently, it has been shown that computability classes of functions computable in recursive analysis can also 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 functions over the reals. 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 functions over the reals, and also provide 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. In particular, unlike what happens in the latter 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. Models include the Blum-Shub-Smale BSS model [3, 4], Shannon's General Purpose Analog Computer (GPAC) [25], algebraically defined classes of functions over the reals à la Moore 96 ($\mathbb{R}$-recursive functions) [23], as well as the recursive analysis approach.

1998 ACM Subject Classification: Computational and structural complexity.
Key words and phrases: Recursive Analysis, Polynomial Time, Algebraic Characterization, Real Computation, Oracle Turing Machines.

Recursive analysis was introduced by Turing [26], Grzegorczyk [16], and Lacombe [21]. 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 quickly converging to some x to a sequence quickly converging to $f(x)$, for all x. That means that this notion of computability requires *a priori* to deal with functionals, or higher order Turing machines.

There is no hope to unify all approaches of computations over the reals: for example the BSS approach can not be conciliated with the recursive analysis point of view, as a non-continuous function can be computed in the BSS approach. However, if we put aside this latter model, which is more motivated by discussing algebraic complexity of problems rather than being a universal model for computations over the reals, 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 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.

Relating models, known to be related at the computability level, at the complexity level is an even more ambitious goal. A first immediate deep problem, is that defining time and space complexity for some of the models, such as for the GPAC is very hard. One reason is that there is no robust and well defined notions of time and space for these models, as shown by several attempts [23, 1, 24, 5].

We prove in this paper that this 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 and well developed and rather well understood theory of complexity for real computations in recursive analysis [20]. We relate it in this paper to a subclass of $\mathbb{R}$-recursive functions, that is to say to machine-independent algebraically defined classes of functions over the reals à la Moore 96 [23].

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

We actually provide 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. As we said, we apply this framework to get a characterization of polynomial time computability over the reals. Besides, we also apply this framework to re-obtain or extend results at the computability level: We extend [14], and prove that computable functions over the reals correspond to functions generable by Shanon's GPAC; we extend [7, 9, 6] and prove that computable functions and elementarily time computable functions correspond to natural subclasses of $\mathbb{R}$-recursive functions. In particular, unlike [14, 7, 9, 6], we provide characterizations that work even for non-Lipschtiz functions (and that differ slightly for Lipschtiz 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 effectively program it, as well as the possibility of building methods to automatically derive computational properties of programs/systems, in the lines of [17, 18, 22] 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 resort to any kind of machinery in the spirit of (Type 1 or Type 2) Turing machines. This might be considered as very natural avoiding discrete machinery when talking about computation of real functions.

The paper is organized as follows. Section 2 discusses some related work. Section 3 introduces a preliminary background in recursive analysis. Section 4 gives some relationships between polynomial time real computability (and other computability/complexity classes) and approximate computability over the integers, starting from the special case of Lipschtiz functions moving on to the general case. Section 5 applies the results obtained in Section 4 to characterize different real computability and complexity classes by function algebras.

2. Related Work

We prove our results by relating the notion of (polynomial-time) computable functions over the integers to the corresponding notion over the reals. This gives a direct way to lift algebraic characterizations of computability and complexity classes over the integers to algebraic characterizations of the corresponding classes over the reals.

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. This can be seen as a way to reformulate/reprove/reread very nicely some constructions already used in [6, 7].

Hence, this framework gives a way to rely on algebraic machine-independent characterizations of computable functions over the integers. Several such characterization 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 perhaps 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 be also noted our characterization is machine-free and relies exclusively on functions over the reals, hence it can not be compared with approaches such as [19]. 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 in spirit 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 also restricted to the computability level, and less close in spirit to the above mentioned models of continuous computation.

In this paper, for ease of presentation, we are restricting to functions 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 [27, 20] 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 notations.

Definition 3.1. 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*:

$$\forall n \in \mathbb{N} : |x - \varphi_x(n)| \leq 2^{-n} \tag{3.1}$$

Given $x \in \mathbb{R}$, let CF_x denote the class of Cauchy functions that represent x.

Definition 3.2 (Computability of real functions)**.** Assume a function $f : D \subseteq \mathbb{R} \to \mathbb{R}$, where D has only one connected component (on 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:

$$|M^{\varphi_x}(n) - f(x)| \leq 2^{-n}. \tag{3.2}$$

If $D = [0, 1]$, then we say f is *polytime computable* (or *polynomial-time 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. A derived notion of continuity that plays an essential role in the investigation of real computation is the modulus of continuity [13].

Definition 3.3 (Modulus of continuity)**.** Assume 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 $m : \mathbb{N} \to \mathbb{N}$.

In analogy with [20, corollary 2.14], computability over unbounded domains can be characterized as indicated by the following proposition [13].

Proposition 3.4. *Assume 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.

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 Lipschtiz functions. In the second step, we discuss how to avoid the Lipschtiz hypothesis, and we consider general functions. Proofs can be found in the appendix.

Without loss of generality we will assume in this section that the compact domain is always the unit interval $[0,1]$. Actually, before all that, let's first provide a preliminary first result to help to explain 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 compensating for the precision of the computed approximation.

Proposition 4.1 (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:*

$$\forall x \in [0,1], \forall y \in \mathbb{N}\colon |g(x,y) - yf(x)| \leq 1 \tag{4.1}$$

We would like to talk of functions g with assertions like above but where quantifiers are only about integers, that is to say about assertions like (4.1) but with something like $\forall x \in \mathbb{N}$ instead of $\forall x \in [0,1]$.

To move 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. Lipschtiz functions

For Lipschtiz 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 *Lipschtiz* if there exists a constant $K \geq 0$ such that for all $x_1, x_2 \in [0,1]$ the following holds:

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2| \tag{4.2}$$

We can now state:

Proposition 4.2 (Complexity over $[0,1]$ vs Complexity over $\mathbb{R} \times \mathbb{R}$)**.** *Assume an arbitrary constant $\epsilon \geq 0$ is fixed. Assume a function $f : [0,1] \to \mathbb{R}$ that is Lipschtiz. 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:*

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y\colon |g(x,y) - yf(\frac{x}{y})| \leq \epsilon \tag{4.3}$$

In order to interrelate with discrete complexity classes we suggest to employ some notion of approximation.

Definition 4.3 (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 $\mathbb{N}$. 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

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

We then have the following result.

Theorem 4.4 (Complexity over $[0, 1]$ vs approximate complexity over $\mathbb{N}^2$)**.** *Consider a class $\mathcal{C}$ of polytime computable real functions that approximates polytime computable discrete functions. Assume that $f : [0, 1] \to \mathbb{R}$ is Lipschtiz. Then the following are equivalent:*

1. *f is polytime computable,*
2. *f is $\mathcal{C}$-definable:*
   that is to say, there exists a function $\tilde{g} \in \mathcal{C}$ such that the following holds

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y \colon |\tilde{g}(x, y) - yf(\frac{x}{y})| \leq 3 \tag{4.4}$$

For the purposes of this article the choice of the constant $\frac{1}{4}$ in the previous definition, and of the constant 3 in last theorem can be arbitrary.

In the direction (2) $\Rightarrow$ (1), Eq. (4.4) implicitly provides a way to efficiently approximate f from $g \restriction \mathbb{N}^2$. Computability of f is possible, in particular at the limit points, from the fact that it is Lipschtiz, and efficiency is possible by the fact that g is polytime computable. The direction (1) $\Rightarrow$ (2) relates polytime computability of real functions to the corresponding discrete notion.

Remark 4.5. Note that all the previous results still hold if we replace 'polytime computable' by just 'computable'.

4.3. Avoiding the Lipschtiz hypothesis

The major obstacle to avoid the Lipschtiz 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 small unit intervals and with respect to its integer projection.

Let's first define the following class of functions.

Definition 4.6 ($\#_k$)**.** Fix some $k \in \mathbb{N}$. Denote by $\#_k$ the function $\#_k : \mathbb{R}^{\geq 1} \to \mathbb{R}$ defined by $\#_k[x] = 2^{((\log_2 x)^k)}$.

We need another approximation notion that is a kind of converse to that given in Definition 4.3.

Definition 4.7 (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 polynomial time 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.

The following proposition is then the non-Lipschtiz version of Proposition 4.2.

Proposition 4.8 (Complexity over $[0,1]$ vs Complexity over $\mathbb{R}^2$)**.** *Assume an arbitrary constant $\epsilon \geq 0$ is fixed. 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,*

$$\forall x \in [0,1], \forall y \in \mathbb{R}^{\geq 1}\colon |g(x.\#_k[y], y) - yf(x)| \leq \epsilon, \tag{4.5}$$

   (c) *for some integer M,*

$$\forall x_1, x_2 \in \mathbb{R}^{\geq 0}, y \in \mathbb{R}^{\geq 1}\colon |x_1 - x_2| \leq 1 \Rightarrow |g(x_1, y) - g(x_2, y)| \leq M \tag{4.6}$$

We need to consider real functions that are well behaved with respect to their restriction to $\mathbb{N}^2$. For ease of notation, we will use $[a,b]$ to denote $[a,b]$, when $a < b$, and $[b,a]$ otherwise.

Definition 4.9 (Peaceful functions)**.** A function $g : \mathbb{R}^2 \to \mathbb{R}$ is said to be *peaceful* if

$$\forall x \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{N}^{\geq 1}, : g(x,y) \in [g(\lfloor x \rfloor, y), g(\lceil x \rceil, y)] \tag{4.7}$$

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 $\mathcal{D}$.

Now we can have the non-Lipschtiz version of Theorem 4.4.

Theorem 4.10. *(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.*[1] *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:*
      that is to say, there exists some peaceful function $g \in \mathcal{C}$ such that

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_k[y]\colon |g(x,y) - yf(\frac{x}{\#_k[y]})| \leq 2, \tag{4.8}$$

   (b) *f is n^k-smooth:*
      that is to say, there exists some integer M such that

$$\forall x, x' \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{R}^{\geq 1}, x, x' \leq \#_k[y], |x - x'| \leq 1\colon$$
$$y|f(\frac{x}{\#_k[y]}) - f(\frac{x'}{\#_k[y]})| \leq M \tag{4.9}$$

Remark 4.11. Condition (2b) is a property of function f, and hence is a necessary condition for the computability of f. This condition is also needed for the continuity of f at the irrational points.

Proof. (1) $\Rightarrow$ (2) : Assume a function $f\colon [0,1] \to \mathbb{R}$ that is polytime computable. By Proposition 4.8 for $\epsilon = 3/4$, there exists some function g with a polytime computable integer approximation h such that (4.5) holds. Now, by the hypothesis of this theorem, there exists some peaceful $\tilde{h} \in \mathcal{C}$ such that

$$\forall x, y \in \mathbb{N} : |\tilde{h}(x,y) - h(x,y)| \leq 1/4$$

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

Hence

$$\forall x, y \in \mathbb{N}\colon |\tilde{h}(x,y) - g(x,y)| \leq 1 + \frac{1}{4} = \frac{5}{4} \tag{4.10}$$

Finally, we have (through change of variables in Eq. (4.5) and restricting the domains of the variables to $\mathbb{N}$)

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_k[y]\colon |\tilde{h}(x,y) - yf(\frac{x}{\#_k[y]})| \leq \frac{5}{4} + \frac{3}{4} = 2 \tag{4.11}$$

Hence, condition 2a holds. Now, by (2c) of Proposition 4.8, we know that for all $x \in \mathbb{R}$, $y \in \mathbb{R}^{\geq 1}$, and $\delta \in [0,1]$: $|g(x+\delta, y) - g(x,y)| \leq M$ for some integer M. Then by using Eq. (4.5) (after variable change and renaming), condition (2b) is satisfied.

(2) $\Rightarrow$ (1) : This follows by applying Proposition 4.8 as follows. From the hypothesis of this theorem g has a polytime computable integer approximation, hence condition 2a of Proposition 4.8 is satisfied. Condition 2a of the current theorem is equivalent to condition 2b of Proposition 4.8 by: (1) letting $\epsilon = 2$, (2) renaming of the variables, and (3) observing that the proof of Proposition 4.8 can be adapted to a new version of condition 2b for which x and y assume only integer values. Using the fact that g is peaceful (controlling the behavior of g between integer points) condition (2c) of Proposition 4.8 can be easily verified. ∎

Remark 4.12. Note that all the previous results still hold if we replace 'polytime computable' by just 'computable'.

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

Corollary 4.13. *Let $\mathcal{D}$ be some class of time-constructive functions from $\mathbb{N}$ to $\mathbb{N}$ that includes polynomial functions and closed under composition. For a function $T \in \mathcal{D}$, define $\#_T : \mathbb{R}^{\geq 1} \to \mathbb{R}$ by $\#_T[x] = 2^{T(\log_2 x)}$. Consider a class $\mathcal{C}$ of functions that peacefully approximates discrete functions computable in time $\mathcal{D}$; and whose functions have integer approximations computable in time $\mathcal{D}$.*[2] *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:*
   that is to say, there exists some peaceful function $g \in \mathcal{C}$ such that

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_T[y]\colon |g(x,y) - yf(\frac{x}{\#_T[y]})| \leq 2, \tag{4.12}$$

   (b) *f is T-smooth:*
   that is to say, there exists some integer M such that

$$\forall x, x' \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{R}^{\geq 1}, x, x' \leq \#_T[y], |x - x'| \leq 1\colon$$
$$y|f(\frac{x}{\#_T[y]}) - f(\frac{x'}{\#_T[y]})| \leq M \tag{4.13}$$

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 [20, Theorem 2.19]. ∎

[2] 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 5.12 and 5.13 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 [25] 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 this model. In the following, we will use Graça and Costa's characterization by PIVP (Polynomial Initial Value Problems) [15]. 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.

Next lemma follows from the constructions in [14]:

Lemma 5.1. *PIVP functions is a class of computable functions that peacefully approximate total (discrete) recursive functions.*

We can then furthermore obtain the following resultas a direct application of Theorem 4.4 and Corollary 4.13 (and Remark 4.12).

Proposition 5.2 (Variation of [6])**.** *A Lipschtiz function $f : [0, 1] \to \mathbb{R}$ is computable iff it is PIVP-definable.*

Proposition 5.3 (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 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 matches discrete elementary functions.

Lemma 5.4. *$\mathcal{L}$ 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 5.5 (Variation of [9])**.** *A Lipschtiz function $f : [0,1] \to \mathbb{R}$ is computable in elementary time iff it is $\mathcal{L}$-definable.*

Proposition 5.6 (Extension of [9])**.** *Let $f : [0,1] \to \mathbb{R}$ be some T-smooth function, for some elementary function $T\colon \mathbb{N} \to \mathbb{N}$. 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 from the constructions of [7] to extend the class of total (discrete) recursive functions:

Lemma 5.7. *$\mathcal{L}_\mu$ is a class of computable functions that peacefully approximate total discrete recursive functions.*

And hence, as a consequence to Theorem 4.4 and Corollary 4.13, we obtain:

Proposition 5.8 (Variation of [7])**.** *A Lipschtiz function $f : [0,1] \to \mathbb{R}$ is computable iff it is $\mathcal{L}_\mu$-definable.*

Proposition 5.9 (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 polynomial time 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: *normal* arguments which come first followed by *safe* arguments using ';' for separation. For any $n \in \mathbb{N}$ we call $[2n, 2n+1]$ an even interval and $[2n+1, 2n+2]$ an odd interval.

Definition 5.10. Define the following class of real functions

$$\mathcal{W} = [0, U, s_0, s_1, pr_0, pr_1, \theta_1, e, o; SComp, SI, Lin] \tag{5.1}$$

(1) a zero-ary function for the constant 0: $0(;) = 0$,
(2) a set of projection functions $U_i^{m+n}(x_1, \ldots, x_m; x_{m+1}, \ldots, x_{m+n}) = x_i$,
(3) successor functions, $s_i(;x) = 2x + i$ for $i \in \{0, 1\}$,
(4) two predecessor functions

$$pr_0(;x) = \begin{cases} n & 2n \leq x \leq 2n+1, n \in \mathbb{Z} \\ n + (\epsilon - 1) & 2n+1 \leq x \leq 2n+\epsilon, 1 \leq \epsilon \leq 2 \end{cases}$$

$$pr_1(;x) = \begin{cases} n + \epsilon & 2n \leq x \leq 2n+\epsilon, 0 \leq \epsilon \leq 1 \\ n+1 & 2n+1 \leq x \leq 2n+2 \end{cases}$$

So the function pr_0 acts as $\lfloor \frac{x}{2} \rfloor$ over even intervals and piecewise linear otherwise; whereas the function pr_1 acts as $\lceil \frac{x}{2} \rceil$ over odd intervals and piecewise linear otherwise.

(5) a continuous function to sense inequalities, $\theta_1(;x) = \max\{0, x\}$,

(6) parity distinguishing functions

$$e(;x) = \frac{\pi}{2}\theta_1(sin\pi x)$$
$$o(;x) = \frac{\pi}{2}\theta_1(-sin\pi x)$$

The function e is non-zero only over even intervals and conversely o is non-zero only over odd intervals. On the integer values $e(;k) = o(;k) = 0$. Note that $\int_{2n}^{2n+2} e(;x)dx = \int_{2n}^{2n+2} o(;x)dx = 1$.

(7) safe composition operator $SComp$: assume 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

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

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

(8) safe integration operator SI: assume functions $g, h_0, h_1 \in \mathcal{W}$, define a new function f satisfying[3]

$$\begin{aligned} f(0,\bar{y};\bar{z}) =& g(\bar{y};\bar{z}) \\ \partial_x f(x,\bar{y};\bar{z}) =& e(x;)[h_1(pr_0(x;),\bar{y};\bar{z}, f(pr_0(x;),\bar{y};\bar{z})) \\ & - h_0(pr_0(x;),\bar{y};\bar{z}, f(pr_0(x;),\bar{y};\bar{z}))] \\ & + o(x;)[h_0(pr_1(x;),\bar{y};\bar{z}, f(pr_1(x;),\bar{y};\bar{z})) \\ & - h_1(pr_1(x;)-1,\bar{y};\bar{z}, f(pr_1(x;)-1,\bar{y};\bar{z}))] \end{aligned}$$

Note that at the integer points this definition reduces to some sort of recursion on the binary notation of the input.

(9) A linearization operator Lin: given functions $g, h\mathcal{W}$, define a new function f by

$$f(x,\bar{y};\bar{z}) = \begin{cases} \delta h(2pr_0(x;)+1,\bar{y};\bar{z}) + (1-\delta)g(2pr_0(x;),\bar{y};\bar{z}) & e(;x) \geq o(;x) \\ \delta' g(2pr_1(x;),\bar{y};\bar{z}) + (1-\delta')h(2pr_1(x;)-1,\bar{y};\bar{z}) & o(;x) \geq e(;x) \end{cases}$$

where $\delta = x - 2pr_0(x;)$, $\delta' = x + 1 - 2pr_1(x;)$. Note that at even integers f reduces to g whereas at odd integers it reduces to h.

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

Proposition 5.11.

(1) *Class $\mathcal{W}$ preserves the integers, that is for every $f \in \mathcal{W}$, $f \restriction \mathbb{N}: \mathbb{N} \to \mathbb{N}$.*

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

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

[3]For simplicity we misused the basic functions 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).

The proof of above proposition (in appendix) follows from induction. The existence of a peaceful extension in part 2 of the proposition is due to the possible application of the linearization operator. The proposition indicates that $\mathcal{W}$ is a class of polytime computable real functions that approximates polytime computable discrete functions. Hence, using Theorem 4.4 the following result is obtained.

Theorem 5.12. *A Lipschtiz 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 4.13, we get to the following result.

Theorem 5.13. *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 class $\mathcal{C}$ (see definition of $\mathcal{C}$-definability). In other words,

Corollary 5.14. *The class of polynomial time computable functions can be characterized algebraically in a machine-independent way:*

(1) *A Lipschtiz function $f : [0, 1] \to \mathbb{R}$ is polytime computable iff it belongs to*
$$Def[0, U, s_0, s_1, pr_0, pr_1, \theta_1, e, o; SComp, SI, Lin]$$
(2) *A n^k-smooth function $f : [0, 1] \to \mathbb{R}$ for some k is polytime computable iff it belongs to*
$$Def[0, U, s_0, s_1, pr_0, pr_1, \theta_1, e, o; SComp, SI, Lin]$$

where $Def[\mathcal{C}]$ stands for $\mathcal{C}$-definability.

References

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

[2] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Computational 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. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

[5] Olivier Bournez and Manuel L. Campagnolo. *New Computational Paradigms. Changing Conceptions of What is Computable*, chapter A Survey on Continuous Time Computations, 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. *Journal of Complexity*, 23(3):317–335, June 2007.

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

[8] Vasco Brattka. Computability over topological structures. In S. Barry Cooper and Sergey 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. *Journal of 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 *Electronic Notes in Theoretical Computer Science*, pages 387–423, 2007.
[11] Peter Clote. Computational models and function algebras. In Edward 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] Walid Gomaa. Characterizing polynomial time computability of rational and real functions. *To appear in EPTCS*.
[14] 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 Computer Science*, pages 169–179. Springer, 2005.
[15] Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. 19(5):644–664, 2003.
[16] A. Grzegorczyk. Computable functionals. *Fundamenta Mathematicae*, 42:168–202, 1955.
[17] M. Hofmann. Type systems for polynomial-time computation, 1999. Habilitation.
[18] Neil D. Jones. The expressive power of higher-order types or, life without CONS. *Journal of Functionnal Programming*, 11(1):5–94, 2001.
[19] Bruce M. Kapron and Stephen A. Cook. A new characterization of type-2 feasibility. *SIAM Journal on Computing*, 25(1):117–132, 1996.
[20] Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991.
[21] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. *Comptes Rendus de l'Académie des sciences Paris*, 241:151–153, 1955.
[22] 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 Computer Science*, pages 25–42. Springer, Nov 2000.
[23] Cristopher Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44, 1996.
[24] 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 Computer Science*, pages 358–371. Springer, Berlin, 1994.
[25] Claude E. Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337–354, 1941.
[26] Alan. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.
[27] Klaus Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.

Appendix A. Proofs

A.1. Proof of Proposition 4.1

Proof. (1) ⇒ (2) : is obtained directly by letting $g(x, y) = yf(x)$. Note that multiplication can be done in polynomial time.
(2) ⇒ (1) : Since g is polytime computable, there exists an oracle machine $N^{()}$ which efficiently computes g. Assume an input $x \in [0, 1]$ and a Cauchy sequence $\varphi_x \in CF_x$. Assume $n \in \mathbb{N}$ and consider an oracle machine $M^{\varphi_x}(n)$ that does the following:

(1) Simulate the operation of $N^{\varphi_x,\varphi_y}(0)$ (for any oracle φ_y),
(2) Whenever $N^{()}$ queries $\varphi_x(i)$, $M^{()}$ queries its own oracle and returns $d = \varphi_x(i)$. Whenever $N^{()}$ queries $\varphi_y(j)$, $M^{()}$ returns 2^{n+1},
(3) Repeat the last step as long as $N^{()}$ keeps querying,
(4) Let e be the output of $N^{()}$. Output $2^{-(n+1)}e$.

First note that $M^{\varphi_x}(n)$ operates in polytime. Next we need to verify its correctness. We have

$$\begin{aligned} &|e - g(x, 2^{n+1})| \leq 1 \\ &|2^{-(n+1)}e - 2^{-(n+1)}g(x, 2^{n+1})| \leq 2^{-(n+1)} \end{aligned} \tag{A.1}$$

From the proposition hypothesis:

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

Then

$$\begin{aligned} &|M^{\varphi_x}(n) - f(x)| \leq |M^{\varphi_x}(n) - 2^{-(n+1)}g(x, 2^{n+1})| + |2^{-(n+1)}g(x, 2^{n+1}) - f(x)| \\ &= |2^{-(n+1)}e - 2^{-(n+1)}g(x, 2^{n+1})| + |2^{-(n+1)}g(x, 2^{n+1}) - f(x)| \\ &\leq 2^{-(n+1)} + |2^{-(n+1)}g(x, 2^{n+1}) - f(x)| \qquad \textit{from Inequality A.1} \\ &\leq 2^{-(n+1)} + 2^{-(n+1)} \qquad \textit{from Inequality A.2} \\ &\leq 2^{-n} \end{aligned}$$

■

A.2. Proof of Proposition 4.2

Proof. (1) ⇒ (2) : Define g on the integer points as follows:

$$g(x, y) = \begin{cases} 0 & y = 0 \\ yf(\frac{x}{y}) & x \leq y, y \geq 1 \\ yf(1) & otherwise \end{cases} \tag{A.3}$$

and piecewise linear for non-integer values. Clearly, g is polytime computable and satisfies (4.3).

(2) ⇒ (1) : Without loss of generality assume $\epsilon = 1$. Since f is Lipschtiz there exists a computable non-negative constant M such that for all $x, y \in [0, 1]$ the following holds:

$|f(x) - f(y)| \leq M|x - y|$. Let $a \in \mathbb{N}$ such that $M \leq 2^a$. Hence, for all $x, y \in [0,1]$ the following holds: $|f(x) - f(y)| \leq 2^a|x - y|$. Since g is polytime computable, there exists an oracle machine $N^{()}$ which efficiently computes g. Assume an input $x \in [0,1]$ and a Cauchy function $\varphi_x \in CF_x$. Assume $n \in \mathbb{N}$ and consider an oracle machine $M^{\varphi_x}(n)$ that does the following:

(1) Let $n' = n + 2 + a$ and let $d = \varphi_x(n')$,
(2) Then $|d - x| \leq 2^{-n'}$, hence, it can be assumed without loss of generality that (for example by truncating extra bits), $d = \frac{k_1}{2^{n'}}$ for some $k_1 \in \mathbb{N}$,
(3) Simulate the operation of $N^{\varphi_x,\varphi_y}(0)$ (for any oracle φ_y),
(4) Whenever $N^{()}(0)$ queries $\varphi_x(i)$, $M^{()}$ returns k_1. Whenever $N^{()}$ queries $\varphi_y(j)$, $M^{()}$ returns $2^{n'}$,
(5) Repeat the last step as long as $N^{()}$ keeps querying,
(6) Let e be the output of $N^{()}$. Output $2^{-n'}e$.

It is clear that $M^{\varphi_x}(n)$ operates in polynomial time with respect to n. Now verifying the correctness of the above procedure. We have

$$\begin{aligned} &|e - g(k_1, 2^{n'})| \leq 1 \\ &|2^{-n'}e - 2^{-n'}g(k_1, 2^{n'})| \leq 2^{-n'} \end{aligned} \tag{A.4}$$

From Eq. (4.3) with $\epsilon = 1$

$$\begin{aligned} &|g(k_1, 2^{n'}) - 2^{n'}f(\frac{k_1}{2^{n'}})| \leq 1 \\ &|2^{-n'}g(k_1, 2^{n'}) - f(\frac{k_1}{2^{n'}})| \leq 2^{-n'} \end{aligned} \tag{A.5}$$

From Inequalities (A.4) and (A.5) we have

$$|2^{-n'}e - f(\frac{k_1}{2^{n'}})| \leq 2^{-(n'-1)} \tag{A.6}$$

From the fact that f is Lipschtiz we have

$$|f(\frac{k_1}{2^{n'}}) - f(x)| \leq 2^a 2^{-n'} = 2^{-(n+2)} \tag{A.7}$$

From Inequalities (A.6) and (A.7) we have

$$|2^{-n'}e - f(x)| \leq 2^{-(n'-1)} + 2^{-(n+2)} \leq 2^{-n} \tag{A.8}$$

This completes the proof that f is polytime computable.

■

A.3. Proof of Theorem 4.4

Proof. (1) ⇒ (2) : Assume an Lipschtiz function $f\colon [0,1] \to \mathbb{R}$ that is polytime computable, then by Proposition 4.2, there exists some polytime computable g such that (4.3) holds with $\epsilon = \frac{3}{4}$. Computing g with precision $1/2$, one can easily build some polytime computable

discrete function $h : \mathbb{N}^2 \to \mathbb{N}$ such that $|h(x,y) - \lfloor g(x,y) \rfloor| \leq 1$. From the theorem hypothesis there exists some $\tilde{g} \in \mathcal{C}$ such that

$$\forall x, y \in \mathbb{N} : |\tilde{g}(x,y) - h(x,y)| \leq 1/4$$

Hence

$$\forall x, y \in \mathbb{N} \colon |\tilde{g}(x,y) - \lfloor g(x,y) \rfloor| \leq 1 + \frac{1}{4} = \frac{5}{4} \tag{A.9}$$

We have $|g(x,y) - \lfloor g(x,y) \rfloor| \leq 1$, then $|\tilde{g}(x,y) - g(x,y)| \leq \frac{9}{4}$. Finally, we have the desired result

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y \colon |\tilde{g}(x,y) - yf(\frac{x}{y})| \leq \frac{9}{4} + \frac{3}{4} = 3 \tag{A.10}$$

(2) $\Rightarrow$ (1) : This follows from Proposition 4.2 with $\epsilon = 3$, observing that functions from $\mathcal{C}$ are assumed polytime computable.

■

A.4. Proof of Proposition 4.8

Proof. (2) $\Rightarrow$ (1) : For simplicity, assume $\epsilon = 1$. Assume 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

$$\begin{aligned} &|g(2^{n^k}x, 2^n) - 2^n f(x)| \leq 1 \\ &|2^{-n} g(2^{n^k}x, 2^n) - f(x)| \leq 2^{-n} \end{aligned} \tag{A.11}$$

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

$$|g(\lfloor 2^{n^k}x \rfloor, 2^n) - h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq 1 \tag{A.12}$$

From (2c) we have

$$|g(\lfloor 2^{n^k}x \rfloor, 2^n) - g(2^{n^k}x, 2^n)| \leq M \tag{A.13}$$

From the previous two equations

$$\begin{aligned} &|g(2^{n^k}x, 2^n) - h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq M + 1 \\ &|2^{-n} g(2^{n^k}x, 2^n) - 2^{-n} h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq 2^{-n}(M+1) \end{aligned} \tag{A.14}$$

From Equations (A.11) and (A.14)

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

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. 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

$$|g(\lfloor 2^{n^k}d \rfloor, 2^n) - g(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq M \tag{A.16}$$

By 2a and the choice of h

$$|g(\lfloor 2^{n^k}d \rfloor, 2^n) - h(\lfloor 2^{n^k}d \rfloor, 2^n)| \leq 1 \tag{A.17}$$

From Equations A.12, A.16, and A.17 we have

$$|h(\lfloor 2^{n^k}d \rfloor, 2^n) - h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq M + 2$$
$$|2^{-n}h(\lfloor 2^{n^k}d \rfloor, 2^n) - 2^{-n}h(\lfloor 2^{n^k}x \rfloor, 2^n)| \leq 2^{-n}(M+2) \tag{A.18}$$

From the last equation and Eq. A.15 we have the required conclusion

$$|f(x) - 2^{-n}h(\lfloor 2^{n^k}d \rfloor, 2^n)| \leq 2^{-n}(2M+4) \tag{A.19}$$

(1) $\Rightarrow$ (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:

$$g(x,y) = \begin{cases} yf(\frac{x}{\#_k[y]}) & x \leq \#_k[y] \\ yf(1) & otherwise \end{cases} \tag{A.20}$$

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 for $M = 1$.

<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 satisfied for $M = 1$.
case 3: $x_1 \leq \#_k[y]$ and $x_2 > \#_k[y]$, 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 satisfied for $M = 1$.

Note that division, multiplication, 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:

(1) Shift right the binary representation of i by $|j|^k$ positions. The result would be a dyadic rational d.
(2) 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.
(3) 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 of step 2 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. ∎

A.5. Proof of Proposition 5.11

Proof. Proof is by induction on the construction of functions in $\mathcal{W}$. Note that for integer values the parity functions e and o are always 0. It is easy to see that the other basic functions preserve the integers and that composition preserves this property. Assume two functions $g, h \in \mathcal{W}$ that preserve $\mathbb{N}$ and consider the application of linearization. Given an input $x = 2n$ any of the two cases of the definition of f can be applied (with $\delta = 0$ and $\delta' = 1$) to give $f(2n, \bar{y}; \bar{z}) = g(2n, \bar{y}; \bar{z})$ which is integer. Alternatively, given $x = 2n+1$ then again any of the two cases can be applied (with $\delta = 1$ and $\delta' = 0$) to give $f(2n + 1, \bar{y}; \bar{z}) = h(2n + 1, \bar{y}; \bar{z})$ which is again an integer by the induction hypothesis. Assume functions $g, h_0, h_1 \in \mathcal{W}$ that preserve $\mathbb{N}$ and consider the application of the safe integration operator to define a new function $f \in \mathcal{W}$. We then use strong induction over the integration variable to show f preserves $\mathbb{N}$. The base case $f(0, \bar{y}; \bar{z}) = g(\bar{y}; \bar{z})$ holds by assumption on g. Assume $f(j, \bar{y}; \bar{z}) \in \mathbb{N}$ for every integer $j \leq 2n$ and consider an input $x \in [2n, 2n + 1]$. Then $o(; x) = 0$, $e(; x) \neq 0$, and

$$\begin{aligned}&h_1(pr_0(x;), \bar{y}; \bar{z}, f(pr_0(x;), \bar{y}; \bar{z})) - h_0(pr_0(x;), \bar{y}; \bar{z}, f(pr_0(x;), \bar{y}; \bar{z})) = \\ &h_1(n, \bar{y}; \bar{z}, f(n, \bar{y}; \bar{z})) - h_0(n, \bar{y}; \bar{z}, f(n, \bar{y}; \bar{z}))\end{aligned}$$

This latter difference is independent of the integration variable x. Furthermore, by the hypotheses of the main and secondary inductions it is an integer value. Notice also that

$$\begin{aligned}\int_{2n}^{2n+1} e(; u)du &= \frac{\pi}{2} \int_{2n}^{2n+1} \theta_1(; \sin \pi u)du \\ &= \frac{\pi}{2} \int_{2n}^{2n+1} \sin \pi u \; du = 1\end{aligned}$$

This implies that

$$f(2n+1,\bar{y};\bar{z}) = f(2n,\bar{y};\bar{z}) + h_1(n,\bar{y};\bar{z},f(n,\bar{y};\bar{z})) - h_0(n,\bar{y};\bar{z},f(n,\bar{y};\bar{z}))$$

which is an integer value. Similarly for doing induction over odd intervals. Hence, the safe integration operator preserves $\mathbb{N}$. This completes the proof of the first part of the proposition.

For the second part we prove that $dp(\mathcal{W}) = \{f : \mathbb{N} \to \mathbb{N} : \exists \tilde{f} \in \mathcal{W}\ such\ that \tilde{f} \restriction \mathbb{N} = f\}$ captures polytime discrete computability by showing that this class coincides with the Bellantoni-Cook class. Then by the linearization operator we obtain the desired result. The Bellantoni-Cook class is defined as: $B = [0, U, s_0, s_1, pr, cond; SComp, SRec]$, where $cond$ is the conditional function $cond(;x,y,z)$ outputs y if x is even and z otherwise and $SRec$ is the safe recursion operator [2].

$B \subseteq dp(\mathcal{W})$: Proof is by induction on the construction of functions in B. It is obvious that the basic functions $0, U, s_0, s_1$ exist in $dp(\mathcal{W})$. The function pr_0 from $\mathcal{W}$ acts exactly like pr when restricted to $\mathbb{N}$. Using U we can define the identity function inside $\mathcal{W}$. Then using linearization we can define the function:

$$f(;x,y,z) = \begin{cases} \delta z + (1-\delta)y & e(;x) \geq o(;x) \\ \delta' y + (1-\delta')z & o(;x) \geq e(;x) \end{cases} \tag{A.21}$$

where $\delta = x - 2pr_0(;x)$, $\delta' = x + 1 - 2pr_1(;x)$. It can be easily verified that $f(;2n,y,z) = y$ and $f(;2n+1,y,z) = z$ for every $n \in \mathbb{N}$, hence $f \restriction \mathbb{N} = cond$. The case for safe composition is easy. Now assume $f \in B$ that is defined from g, h_0, h_1 by safe recursion. Then by the induction hypothesis there exist $\tilde{g}, \tilde{h}_0, \tilde{h}_1 \in \mathcal{W}$ such that $\tilde{g} \restriction \mathbb{N} = g$, $\tilde{h}_0 \restriction \mathbb{N} = h_0$, and $\tilde{h}_1 \restriction \mathbb{N} = h_1$. Define the function $\tilde{f} \in \mathcal{W}$ using safe integration from $\tilde{g}, \tilde{h}_0$, and $\tilde{h}_1$. We claim that $\tilde{f} \restriction \mathbb{N} = f$. Proof is by strong induction over the recursion variable. For readability we will exclude the $\bar{y}$ and $\bar{z}$ arguments. At the base case we have $\tilde{f}(0;) = \tilde{g}(;) = g(;) = f(0;)$. From the proof of the first part of the proposition we have

$$\begin{aligned}
\tilde{f}(2n+1;) &= \tilde{f}(2n;) + \tilde{h}_1(n;\tilde{f}(n;)) - \tilde{h}_0(n;\tilde{f}(n;)) \\
&= f(2n;) + \tilde{h}_1(n;f(n;)) - \tilde{h}_0(n;f(n;)), \quad \textit{by induction hypothesis} \\
&= f(2n;) + h_1(n;f(n;)) - h_0(n;f(n;)), \quad \textit{by assumption} \\
&= h_0(n;f(n;)) + h_1(n;f(n;)) - h_0(n;f(n;)), \quad \textit{by safe recursion} \\
&= h_1(n;f(n;)) \\
&= f(2n+1;), \quad \textit{by safe recursion}
\end{aligned}$$

Similarly, it can be shown that $\tilde{f}(2n+2;) = f(2n+2;)$.

$dp(\mathcal{W}) \subseteq B$: Proof is by induction on the construction of functions in $\mathcal{W}$. The case for the basic functions $0, U, s_i, pr_0$ is obvious. For $x \in \mathbb{N}$ we have $(pr_1 \restriction \mathbb{N})(;x) = \lceil \frac{x}{2} \rceil$ which is polynomial time computable. $\theta_1 \restriction \mathbb{N}$ is the identity function. For all $x \in \mathbb{N}$ we have $e(;x) = o(;x) = 0$. Safe composition sustains polynomial time computability. Assume a function $f \in \mathcal{W}$ defined by linearization from g and h. At integer values f reduces either to h or g, hence by the induction hypothesis $f \restriction \mathbb{N}$ is polynomial time computable. Finally, assume a function $f \in \mathcal{W}$ defined by safe integration from g, h_0, and h_1. Let $\hat{g} = g \restriction \mathbb{N}$, $\hat{h}_0 = h_0 \restriction \mathbb{N}$, and $\hat{h}_1 = h_1 \restriction \mathbb{N}$. Then by the induction hypothesis we have $\hat{g}, \hat{h}_0, \hat{h}_1 \in B$. Define $\hat{f} \in B$ by safe recursion from $\hat{g}, \hat{h}_0, \hat{h}_1$. We claim that $f \restriction \mathbb{N} = \hat{f}$; proof is by

strong induction on the integration variable and is similar to that given in the proof that $B \subseteq dp(\mathcal{W})$. This completes the proof of the second part of the proposition.

It is easy to see that the basic functions $0, U, s_i, pr_i$, and θ_1 are all polynomial time computable. The constant π is polynomial time computable. The trigonometric sine and cosine functions are computable in polynomial time using, for example, Taylor series expansion as an approximation. Hence, the parity functions e and o are polynomial time computable. Composition preserves polynomial time computability. Given polynomial time computable functions g and h, then clearly their linearization is polynomial time computable. Assume a function $f \in \mathcal{W}$ that is defined by safe integration from g, h_0, h_1 where these latter functions are polynomial time computable. Then from part 2 of the proposition we have $f \restriction \mathbb{N}$ is polynomial time computable. As can be seen from the proof of part 1 of the proposition the function f is piecewise trigonometric with breakpoints at $\mathbb{N}$, hence it is also polynomial time computable at non-integer points. This completes the proof of the proposition.

■