Implicit complexity in recursive analysis

Olivier Bournez, Walid Gomaa, Emmanuel Hainry

Implicit complexity in recursive analysis

Olivier Bournez[1], Walid Gomaa[2,3], and Emmanuel Hainry[2,4]

[1] LIX, Ecole Polytechnique, 91128 Palaiseau Cedex, France
Olivier.Bournez@lix.polytechnique.fr
[2] Loria, BP 239 - 54506 Vandœuvre-lès-Nancy Cedex, France
[3] Alexandria University, Faculty of Engineering, Alexandria, Egypt
walid.gomaa@loria.fr
[4] Nancy Université, Université Henri Poincaré, Nancy, France
Emmanuel.Hainry@loria.fr

Abstract. Recursive analysis is a model of analog computation which is based on type 2 Turing machines. Various classes of functions computable in recursive analysis have recently been characterized in a machine independent and algebraical context. 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 have been obtained.
We provide in this paper a framework that allows to dive into complexity for functions over the reals. It indeed relates classical computability and complexity classes with the corresponding classes in recursive analysis. This framework opens the field of implicit complexity of functions over the reals.
While our setting provides a new reading of some of the existing characterizations, it also provides new results: inspired by Bellantoni and Cook's characterization of polynomial time computable functions, we provide the first algebraic characterization of polynomial time computable functions over the reals.

1 Introduction

Building a well founded theory of computations over the reals is a crucial task. However, computation theory over the reals is not as well understood as classical discrete computation theory. In particular, unlike what happens for discrete computations where the Church-Turing thesis yields a clear equivalence between models, when talking about real or analog computations, several approaches have been developed, with various motivations, but with not so-clear relations. This includes the Blum-Shub-Smale model [3,4], Shannon's General Purpose Analog Computer (GPAC) [24], algebraically defined classes of functions over the reals à la Moore 96 [22] ($\mathbb{R}$-recursive functions), as well as the recursive analysis approach. Recursive analysis has been introduced by Turing [25], Grzegorczyk [15], Lacombe [20]. In this framework, a function $f : \mathbb{R} \to \mathbb{R}$ over the reals is considered as computable, if there is 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.

While there is no hope to unify the Blum-Shub-Smale approach with recursive analysis point of view, some recent works have shown strong connections between recursive analysis, Shannon's General Purpose Analog Computer model, and $\mathbb{R}$-recursive functions. These results basically say that all these paradigms of computations are more or less equivalent: see [6,7] or survey [5]. These results also provide machine independent descriptions of recursive analysis and of the GPAC which are both deeply rooted on machinery.

However, up till now discussions have mainly been restricted to the computability level, and not to complexity level. While there is a well developed and rather well understood theory of complexity for real computations (over compact domains) [19], talking about complexity for analog models of computations is often very problematic. One reason is that there is no robust and well defined notions of time and space for these models [22,1,23,5].

In this paper, we aim at providing a framework for implicit complexity in recursive analysis. We claim that this framework gives relations between recursive analysis and algebraical classes of functions over the reals at the complexity level. We also apply this framework to re-obtain results at the computability level. This framework is then both well founded and a step towards a sane complexity theory of functions over the reals.

To do so, we actually relate (polynomial-time) computable functions over the reals to (polynomial-time) computable functions over the integers, and provide some statements that allow to state that a class of functions over the reals that approximates in some defined sense computable functions over the integers yields naturally an algebraic characterization of the corresponding class over the reals.

First, our results provide a new understanding of some of the recent constructions relating recursive analysis to Shannon's General Purpose Analog Computer, and to $\mathbb{R}$-recursive functions [6,7].

Second, our results provide, to our knowledge, the first algebraic machine-independent characterization of polynomial time computable functions over the reals. Potential applications include the possibility of proving whether a given function can be computed in polynomial time without resorting to effectively program it.

More concretely, we first express how recursive analysis relates the notion of (polynomial-time) computable functions over the integers to the corresponding notion over the reals. While the obtained results can be seen as rather basic, they yield a direct understanding of how algebraic characterizations of computability and complexity classes over the integers can be lifted to algebraic characterizations of corresponding classes over the reals.

Indeed when talking about computability and complexity over the integers, several machine-independent characterizations of computable functions are known [11]: in particular, Kleene's functions are well known to be exactly the 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 functions. See survey [11] for some other machine independent characterizations of classical computability and complexity classes 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 to be able to compute the corresponding class over the reals. This provides a way to reformulate/reprove/reread very nicely some constructions already used in [6,7].

Notice that our framework is definitively different from the one proposed by Campagnolo and Ojakian in [10], and has the main advantage to allow to talk not only about the computability level but also about the complexity level. It should be also noted that although recursive analysis can be seen as a specific type 2 computation, our characterisation relies exclusively on functions on the reals and hence cannot be compared with approaches such as [18]. 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 above mentioned models of analog computations.

Potential applications 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 [16,17,21] for discrete programs.

2 Basic Definitions from Recursive Analysis

We first recall some definitions from recursive analysis [26,19].

Let $\mathbb{D} = \{r \in \mathbb{Q} \colon r = \frac{a}{2^b} \ for\ integers\ a, b\}$ (these are the rationals with finite binary representation). We adopt the following representation of real numbers.

Definition 1. *Assume $x \in \mathbb{R}$. Then x can be represented by a Cauchy sequence $\varphi_x \colon \mathbb{N} \to \mathbb{D}$ that converges at a binary rate:*

$$\forall n \in \mathbb{N} \colon |x - \varphi_x(n)| \leq 2^{-n}. \tag{1}$$

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

Definition 2. *(Modulus of continuity)*

1. *Assume a function $f \colon [0,1] \to \mathbb{R}$. Then f has a* modulus of continuity *if there exists a function $m \colon \mathbb{N} \to \mathbb{N}$ such that for all $x, y \in [0,1]$ and for all $n \in \mathbb{N}$: if $|x - y| \leq 2^{-m(n)}$, then $|f(x) - f(y)| \leq 2^{-n}$.*

2. *Assume a function* $f\colon [0,\infty) \to \mathbb{R}$. *Then* f *has a* modulus of continuity *if there exists a function* $m\colon \mathbb{N}^2 \to \mathbb{N}$ *such that for all* $k, n \in \mathbb{N}$ *and for all* $x, y \in [0, 2^k]$ *the following holds: if* $|x - y| \leq 2^{-m(k,n)}$, *then* $|f(x) - f(y)| \leq 2^{-n}$.

The notions of computability and complexity of real functions have been characterized as will be shown below.

Definition 3. *Assume a function* $f\colon I \to \mathbb{R}$ *where* I *is either* $[0,1]$ *or* $[0,\infty)$. *We say that* f *is computable if there exists a function-oracle Turing machine* $M^{()}$ *such that for every* $x \in I$, *for every* $\varphi_x \in CF_x$, *and for every* $n \in \mathbb{N}$ *the following holds:*

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

Notice that we need to talk about functions defined over unbounded (non-compact) domains, unlike what is done for example in [19]: Computability over compact domains is characterized by the existence of a computable modulus and a computable approximation function (see corollary 2.14 in [19]). Computability over unbounded domains can be characterized in a similar way as indicated by the following proposition.

Proposition 1. *Assume a function* $f\colon [0,\infty) \to \mathbb{R}$. *Then* f *is* computable *iff there exist two computable functions* $m : \mathbb{N}^2 \to \mathbb{N}$ *and* $\psi : \mathbb{D} \cap [0,\infty) \times \mathbb{N} \to \mathbb{D}$ *such that*

1. m *is a modulus of continuity for* f,
2. ψ *is an* approximation function *for* f, *that is, for every* $d \in \mathbb{D} \cap [0,\infty)$ *and every* $n \in \mathbb{N}$ *the following holds:* $|\psi(d,n) - f(d)| \leq 2^{-n}$.

Proof. The proof is a simple extension of the proof of Corollary 2.14 in [19].

Definition 4. *(Polynomial time complexity of real functions)*

1. *Assume a function* $f\colon [0,1] \to \mathbb{R}$. *Then* f *is polynomial time computable if it is computable in the sense of Definition 3, and there exists an oracle machine* $N^{()}(n)$ *that computes* f *whose computation time is bounded by* $p(n)$ *for some polynomial function* p.
2. *Assume a function* $f\colon [0,\infty) \to \mathbb{R}$. *Then* f *is polynomial time computable if it is computable in the sense of Definition 3, and there exists an oracle machine* $N^{()}(n)$ *that computes* $f(x)$ *whose computation time is bounded by* $q(k,n)$ *for some polynomial* q *where* $k = \min\{j\colon x \in [0, 2^j]\}$.

For a domain that is a combination of compact and unbounded components computability/complexity can be defined and characterized in the obvious way.

The following proposition characterizes polynomial time computability of real functions over unbounded domains. Using an extension of Theorem 2.19 in [19] the proof is similar to that of Proposition 1.

Proposition 2. *Assume a function* $f\colon [0,\infty) \to \mathbb{R}$. *Then* f *is* polynomial time computable *iff there exist two functions* $m : \mathbb{N}^2 \to \mathbb{N}$ *and* $\psi : \mathbb{D} \cap [0,\infty) \times \mathbb{N} \to \mathbb{D}$ *such that*

1. m *is a modulus function for* f *and it is polynomial with respect to both the extension parameter* k *and the precision parameter* n, *that is,* $m(k,n) = O((k+n)^a)$ *for some* $a \in \mathbb{N}$.
2. ψ *is an* approximation function *for* f, *that is, for every* $d \in \mathbb{D} \cap [0,\infty)$ *and for every* $n \in \mathbb{N}$ *the following holds:*

$$|\psi(d,n) - f(d)| \leq 2^{-n} \tag{3}$$

3. $\psi(d,n)$ *is computable in time* $p(|d|+n)$ *for some polynomial* p.

3 Transferring Computability and Complexity over The Integers to the Reals

Proofs of some of the results in this section can be found in the appendix.

3.1 General Case

We now state some results that yield ways to transfer notions of computability/complexity over the integers to corresponding notions over the reals.

We first give a way to relate computability over some compact domain to computability over this domain times reals.

Proposition 3 (Computability over I vs Computability over $I \times \mathbb{R}$). *The following are equivalent.*

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

$$\forall x \in [0,1],\ \forall y \in \mathbb{R}^{\geq 0}\colon |g(x,y) - yf(x)| \leq 1 \tag{4}$$

3. *there exists a computable function* $g : [0,1] \times \mathbb{R}^{\geq 0} \to \mathbb{R}$ *such that,*

$$\forall x \in [0,1], \forall y \in \mathbb{N}\colon |g(x,y) - yf(x)| \leq 1 \tag{5}$$

These statements works fine also at the complexity level[5]:

Proposition 4 (Complexity over I vs Complexity over $I \times \mathbb{R}$). *The following are equivalent:*

1. *a function* $f : [0,1] \to \mathbb{R}$ *is polynomial time computable,*
2. *there exists a polynomial time computable function* $g : [0,1] \times \mathbb{R}^{\geq 0} \to \mathbb{R}$ *such that* (4) *holds.*
3. *there exists a polynomial time computable function* $g : [0,1] \times \mathbb{R}^{\geq 0} \to \mathbb{R}$ *such that* (5) *holds.*

[5] Or at any level above polynomial time. That is to say, polynomial time computable can be replaced by computable in elementary time in following proposition for example.

3.2 Lispchitz Functions

For Lipschitz functions, it is possible to relate computability over compact domains to computability over $\mathbb{R}^2$.

Proposition 5 (Computability over I vs Computability over $\mathbb{R}^2$). *Fix an arbitrary constant ϵ. Assume a function $f : [0,1] \to \mathbb{R}$ that is Lipschitz. Then the following are equivalent:*

1. *f is computable*
2. *there exists some computable function $g\colon \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0} \to \mathbb{R}$ such that*

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y\colon |g(x,y) - yf(\frac{x}{y})| \leq \epsilon \tag{6}$$

This can still be transfered at the complexity level[1]:

Proposition 6 (Complexity over I vs Complexity over $\mathbb{R}^2$). *Fix any arbitrary constant ϵ. Assume a function $f : [0,1] \to \mathbb{R}$ that is Lipschitz. Then the following are equivalent:*

1. *f is polynomial time computable,*
2. *there exists a polynomial time computable function $g\colon \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0} \to \mathbb{R}$ such that (6) holds.*

We propose then to consider the following notion of approximation.

Definition 5 (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*

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

This yields the following result.

Theorem 1 (Computability over I vs Approximate Computability over $\mathbb{N}^2$). *Consider a class $\mathcal{C}$ of computable real functions that approximates total (discrete) recursive functions. Assume that $f : [0,1] \to \mathbb{R}$ is Lipschitz. Then the following are equivalent:*

1. *f is computable,*
2. *there exists some function $g \in \mathcal{C}$ such that*

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y\colon |g(x,y) - yf(\frac{x}{y})| \leq 3 \tag{7}$$

This can also be stated at the complexity level:

Theorem 2 (Complexity over I vs Approximate Complexity over $\mathbb{N}^2$). *Consider a class $\mathcal{C}$ of polynomial time computable functions that approximates polynomial time computable discrete functions. Assume that $f : [0,1] \to \mathbb{R}$ is Lipschitz. Then the following are equivalent:*

1. *f is polynomial time computable,*
2. *there exists a function $g \in \mathcal{C}$ such that (7) holds.*

3.3 Avoiding the Lispchitz Hypothesis

This is indeed possible to avoid the Lipschitz hypothesis, and to yet talk about complexity, at the price of a little bit of complications.

Definition 6 ($\#_k$). *Fix some integer k. Denote by $\#_k$ the function $\#_k : \mathbb{R}^{\geq 1} \to \mathbb{R}^{\geq 0}$ defined by $\#_k[x] = 2^{((\log_2 x)^k)}$.*

Definition 7 (Polynomial Time Computable Integer Approximation). *A function $g : \mathbb{R}^d \to \mathbb{R}$ is said to have a polynomial time computable integer approximation if there exists some polynomial time computable function $h : \mathbb{N}^d \to \mathbb{N}$ with $|h(\bar{x}) - \lfloor g(\bar{x}) \rfloor| \leq 1$ for all $\bar{x} \in \mathbb{N}^d$.*

We then have.

Proposition 7 (Complexity over I vs Complexity over $\mathbb{R}^2$). *Fix an arbitrary constant ϵ. The following are equivalent.*

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

$$\forall x \in [0,1], \forall y \in \mathbb{R}^{\geq 1}\colon |g(x.\#_k[y], y) - yf(x)| \leq \epsilon, \tag{8}$$

   (c) *for some integer M,*

$$\forall x_1, x_2 \in \mathbb{R}^{\geq 0}, y \in \mathbb{R}^{\geq 1}\colon |x_1 - x_2| \leq 1 \Rightarrow |g(x_1, y) - g(x_2, y)| \leq M \tag{9}$$

Notice that (8) is equivalent to (6) for $k = 1$, by some obvious change of variable.

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

$$\begin{aligned} &|g(2^{n^k}x, 2^n) - 2^n f(x)| \leq 1 \\ &|2^{-n} g(2^{n^k}x, 2^n) - f(x)| \leq 2^{-n} \end{aligned} \tag{10}$$

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

Then

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

From (2c) we have

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

From the previous two equations

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

From Equations (10) and (13)

$$|f(x) - 2^{-n}h(\lfloor 2^{n^k}x\rfloor, 2^n)| \leq 2^{-n}(M+3) \tag{14}$$

This gives an approximation to $f(x)$ to within the desired precision 2^{-n}. The computation time of $h(\lfloor 2^{n^k}x\rfloor, 2^n)$ is bounded by $p(n)$ for some polynomial p, hence, $f(x)$ is polynomial time computable.

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

$$g(x,y) = \begin{cases} yf(\frac{x}{\#_k[y]}) & x \leq \#_k[y], y \geq \epsilon \\ yf(1) & otherwise \end{cases} \tag{15}$$

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> $x_1 > \#_k[y]$ and $x_2 > \#_k[y]$, then

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

and condition (2c) is satisfied for $M = 1$.

<u>case 3:</u> $x_1 \leq \#_k[y]$ 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 above two cases, and hence condition (2c) is satisfied for $M = 1$.

Note that all 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 the floor of the last function involves the following:

1. Shift left 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 though fixed precision. 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. The fixed precision of step 2 can be calculated from the modulus of f. 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.

For ease of notation, let's consider that interval $[a, b]$ is $[b, a]$ if $b < a$.

Definition 8 (Peaceful Functions).

– *A function* $g : \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ *is said to be peaceful if*

$$\forall x \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{N}^{\geq 1}, : g(x,y) \in [g(\lfloor x \rfloor, y), g(\lceil x \rceil, y)]. \tag{16}$$

– *We will say that a class* $\mathcal{C}$ *of functions peacefully approximates some class* $\mathcal{D}$ *of functions, if the class of peaceful functions from* $\mathcal{C}$ *approximates* $\mathcal{D}$.

Theorem 3 (Complexity over I vs Approximate Complexity over $\mathbb{N}^2$). *Consider a class* $\mathcal{C}$ *of functions that peacefully approximates polynomial time computable discrete functions, and whose functions have polynomial time computable integer approximation*[6]*.*

Then the following are equivalent.

1. *a function* $f\colon [0,1] \to \mathbb{R}$ *is polynomial time computable,*
2. *there exists some peaceful function* $g \in \mathcal{C}$ *such that*
   (a) for some integer k*,*

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_k[y]\colon |g(x,y) - yf(\frac{x}{\#_k[y]})| \leq 3, \tag{17}$$

   (b) for some integer M*,*

$$\forall x, x' \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{R}^{\geq 1}, x, x' \leq \#_k[y], |x - x'| \leq 1\colon \\ y|f(\frac{x}{\#_k[y]}) - f(\frac{x'}{\#_k[y]})| \leq M \tag{18}$$

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

Remark 1. Condition (2b) is a property of function f, and hence is a necessary condition to get computability of f.

Proof. (1) $\Rightarrow$ (2) : by Proposition 7 for $\epsilon = 3/4$, there exists some function g with a polynomial time computable integer approximation h such that (8) holds.

Now, by hypothesis, 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

$$\forall x, y \in \mathbb{N} \colon |\tilde{h}(x, y) - \lfloor g(x, y) \rfloor| \leq 1 + \frac{1}{4} = \frac{5}{4} \tag{19}$$

We have $|g(x, y) - \lfloor g(x, y) \rfloor| \leq 1$, then $|\tilde{h}(x, y) - g(x, y)| \leq \frac{9}{4}$, for all $x, y \in \mathbb{N}$.

Finally, we have (through change of variables in Eq. (8))

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_k[y] \colon |\tilde{h}(x, y) - yf(\frac{x}{\#_k[y]})| \leq \frac{9}{4} + \frac{3}{4} = 3 \tag{20}$$

Now, by (2c) of Proposition 7, we know that for all $x, y \in \mathbb{N}$, $\delta \in [0, 1]$, $|g(x + \delta, y) - g(x, y)| \leq M$ for some integer M. From Eq. (8), clearly condition (2b) must hold.

(2) $\Rightarrow$ (1) : This follows from Proposition 7 with $\epsilon = 3$, observing that the peacefulness of g and condition (2b) implies condition (2c) of Proposition 7.

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

Corollary 1. *Let $\mathcal{D}$ be some class of functions from $\mathbb{N}$ to $\mathbb{N}$ above the class of polynomial functions and closed under composition. For a function $T \in \mathcal{D}$, consider $\#_T$ as the function $\#_T : \mathbb{R}^{\geq 1} \to \mathbb{R}^{\geq 0}$ defined 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 $\mathcal{D}$[7]*.*

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 peaceful function $g \in \mathcal{C}$ such that*
   (a) for some $T \in \mathcal{D}$

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_T[y] \colon |g(x, y) - yf(\frac{x}{\#_T[y]})| \leq 3, \tag{21}$$

   (b) for some integer M,

$$\forall x, x' \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{R}^{\geq 1}, x, x' \leq \#_T[y], |x - x'| \leq 1 \colon \\ y|f(\frac{x}{\#_T[y]}) - f(\frac{x'}{\#_T[y]})| \leq M \tag{22}$$

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

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 Theorem 2.19 in [19].

4 Applications

We now present applications of our previous theorems and corollaries to characterize algebraically some computability and complexity classes of functions over the reals. The main new results of this sections are theorems 4 and 5 which give characterizations of polynomial-time computable functions on the reals as the functions described by a specific class $\mathcal{W}$.

To do so, we propose the following terminology.

Definition 9 (Definability). *Let $\mathcal{C}$ be a class of functions from $\mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$. A function $f : [0,1] \to \mathbb{R}$ is said to be $\mathcal{C}$-definable if there exists some total function $g \in \mathcal{C}$ such that:*

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y \colon |g(x,y) - yf(\frac{x}{y})| \leq 3 \tag{23}$$

Let T be a function from $\mathbb{N}$ to $\mathbb{N}$.

Definition 10 (T, M-Bounded). *A function $f : [0,1] \to \mathbb{R}$ is said to be T, M-bounded if*

$$\begin{aligned} \forall x, x' \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{R}^{\geq 1}, x, x' \leq \#_T[y], |x - x'| \leq 1 \colon \\ y|f(\frac{x}{\#_T[y]}) - f(\frac{x'}{\#_T[y]})| \leq M \end{aligned} \tag{24}$$

Definition 11 (T-Definability). *Let $\mathcal{C}$ be a class of functions from $\mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$. A function $f : [0,1] \to \mathbb{R}$ is said $\mathcal{C}$-T-definable if there exists some total function $g \in \mathcal{C}$ such that*

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_T[y] \colon |g(x,y) - yf(\frac{x}{\#_T[y]})| \leq 3 \tag{25}$$

4.1 GPAC, Polynomial IVP, and computable functions

The General Purpose Analog Computer, introduced by Claude Shannon in [24] to model a mechanical device, 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 creation of this model. In the following, we will use Graça and Costa's characterization by PIVP (Polynomial Initial Value Problems) [14].

Definition 12. *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 polynomials.

Lemma 1 ([14]). *A function is generated by a GPAC iff it is PIVP.*

Lemma 2 ([13]). *PIVP functions is a class of computable functions that peacefully approximates total (discrete) recursive functions.*

Proof. It is proved in [13] than Turing machines can be robustly simulated by PIVP functions. Robust means that even with noise (that does not exceed 1/4) the computation is conducted satisfactorily. This lemma implies that discrete functions that are computable can be generated by GPACs with precision $\frac{1}{4}$. Hence the class of PIVP functions approximates the discrete recursive functions.

We get the following corollaries (that can be seen as a new reading of [6]) as a direct application of Theorem 1 and Corollary 1.

Proposition 8. *A Lipschitz function $f : [0, 1] \to \mathbb{R}$ is computable iff it is PIVP-definable.*

Proposition 9. *Let $f : [0, 1] \to \mathbb{R}$ be some T, M-bounded function, for some total recursive function T. Then f is computable iff it is PIVP-T-definable.*

4.2 Real recursive functions revisited

Some characterizations of computable functions in the context of real recursive functions [22] are already known, see for example [7,10]. They make use of limits that have restrictive constraints. We can dispense with those limits by using theorem 1 with a simpler class of functions.

Definition 13. *Let $\mathcal{L}_\mu$ be the smallest set of functions containing 0, 1, projections, $\theta_3 : \max(0, x^3)$ and closed under the operations of composition, controlled linear integration (CLI) and unique minimization (UMU).*

CLI is defined for functions g, h and c, with the norm of the first partial derivative of h bounded by c, as a solution to the following differential equation $\partial_y f(x, y) = h(x, y)f(x, y)$ with initial condition $f(x, 0) = g(x)$.

Given a differentiable function $f : \mathcal{D} \times \mathcal{I} \subseteq \mathbb{R}^{k+1} \to \mathbb{R}$ where $\mathcal{D} \times \mathcal{I}$ is a product of closed intervals, if for all $\mathbf{x} \in \mathcal{D}$, $y \mapsto f(\mathbf{x}, y)$ is a non-decreasing function with a unique root y_0 that lies in the interior of $\mathcal{I}$ and $\partial_y f|_{(\mathbf{x},y_0)} > 0$, then UMU(f) *is defined as* $\begin{cases} \mathcal{D} \longrightarrow \mathbb{R} \\ \mathbf{x} \mapsto y_0 \end{cases}$

Lemma 3 ([7]). *Functions in $\mathcal{L}_\mu$ is a class of computable functions that peacefully approximate total recursive functions.*

We get from Theorem 1 and Corollary 1.

Proposition 10. *A Lipschitz function $f : [0,1] \to \mathbb{R}$ is computable iff it is $\mathcal{L}_\mu$ -definable.*

Proposition 11. *Let $f : [0,1] \to \mathbb{R}$ be some T, M-bounded function, for some total recursive function T. Then f is computable iff it is $\mathcal{L}_\mu$-T-definable.*

Similarly, we know that this class without the UMU operator closely matches discrete elementary functions.

Definition 14. *Let $\mathcal{L}$ be the smallest set of functions containing 0, 1, -1, π, projections and θ_3 and closed under composition and controlled linear integration.*

Lemma 4 ([9]). *$\mathcal{L}$ is a class of total functions computable in elementary time that peacefully approximates total discrete elementary computable functions.*

We get from a generalization of Theorem 2 and Corollary 1.

Proposition 12. *A Lipschitz function $f : [0,1] \to \mathbb{R}$ is computable in elementary time iff it is $\mathcal{L}$-definable.*

Proposition 13. *Let $f : [0,1] \to \mathbb{R}$ be some T, M-bounded function, for some elementary function T. Then f is computable in elementary time iff it is $\mathcal{L}$-T-definable.*

4.3 Polynomial Time Computable Functions

We define a class of polynomial time computable real functions. These functions are essentially extensions to $\mathbb{R}$ of the Bellantoni and Cook class [2].

Definition 15. *Define functions algebra*

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

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*

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

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

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

6. *parity distinguishing functions*

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

7. *safe composition operator SComp: assume a vector of functions* $\bar{g}_1(\bar{x};)$*, a vector of functions* $\bar{g}_2(\bar{x};\bar{y})$*, and a function* h *of arity* $|\bar{g}_1| + |\bar{g}_2|$*. Define new function* f

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

8. *safe integration operator SI: assume functions* g, h_0, h_1*, define a new function* f *satisfying*

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

Notice that 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).

9. *A linearization operator Lin: given functions* g, h*, 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 $\delta = x - 2pr_0(x;)$*,* $\delta' = x + 1 - 2pr_1(x;)$*.*

By induction the following can easily be shown.

Proposition 14. *Every function in* $\mathcal{W}$ *is polynomial time computable.*

From the fact that the built functions contain piecewise linear extensions to $\mathbb{R}$ of the Bellantoni and Cook class [2], we have.

Proposition 15. *Every polynomial time computable discrete function has a peaceful extension in* $\mathcal{W}$*.*

We get from Theorem 2 and and Corollary 1.

Theorem 4. *A Lipschitz function* $f : [0,1] \to \mathbb{R}$ *is polynomial-time computable iff it is* $\mathcal{W}$*-definable.*

Theorem 5. *Let* $f : [0,1] \to \mathbb{R}$ *be some* n^k, M*-bounded function for some* k*. Then* f *is polynomial-time computable iff it is* $\mathcal{W}$*-*n^k*-definable.*

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. 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.
14. Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. 19(5):644–664, 2003.
15. Andrzej Grzegorczyk. On the definitions of computable real continuous functions. 44:61–71, 1957.
16. M. Hofmann. Type systems for polynomial-time computation, 1999. Habilitation.
17. Neil D. Jones. The expressive power of higher-order types or, life without CONS. *Journal of Functionnal Programming*, 11(1):5–94, 2001.
18. Bruce M. Kapron and Stephen A. Cook. A new characterization of type-2 feasibility. *SIAM Journal on Computing*, 25(1):117–132, 1996.

19. Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991.
20. 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.
21. 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.
22. Cristopher Moore. Dynamical recognizers: real-time language recognition by analog computers. *Theoretical Computer Science*, 201(1–2):99–136, 6 July 1998.
23. 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.
24. Claude E. Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337–354, 1941.
25. Alan. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.
26. Klaus Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.

A Proofs

A.1 Proof of Proposition 3

Proof. (1) $\Rightarrow$ (2) : is obtained directly by letting $g(x, y) = yf(x)$.
(2) $\Rightarrow$ (1) : Since g is computable, there exists an oracle machine $N^{()}$ which 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 $2^{-(n+1)}e$.

From this procedure we have

$$\begin{aligned} &|e - g(x, 2^{n+1})| \leq 1 \\ &|2^{-(n+1)}e - 2^{-(n+1)}g(x, 2^{n+1})| \leq 2^{-(n+1)} \end{aligned} \tag{26}$$

From the proposition hypothesis:

$$\begin{aligned} &|g(x, 2^{n+1}) - 2^{n+1}f(x)| \leq 1 \\ &|2^{-(n+1)}g(x, 2^{n+1}) - f(x)| \leq 2^{-(n+1)} \end{aligned} \tag{27}$$

Then

$$\begin{aligned} &|M^{\varphi_x}(n) - f(x)| \leq |M^{\varphi_x}(n) - 2^{-(n+1)}g(x, 2^{n+1})| + |2^{-(n+1)}g(x, 2^{n+1}) - f(x)| \\ &= |2^{-(n+1)}e - 2^{-(n+1)}g(x, 2^{n+1})| + |2^{-(n+1)}g(x, 2^{n+1}) - f(x)| \\ &\leq 2^{-(n+1)} + |2^{-(n+1)}g(x, 2^{n+1}) - f(x)| \qquad \textit{from Inequality 26} \\ &\leq 2^{-(n+1)} + 2^{-(n+1)} \qquad \textit{from Inequality 27} \\ &\leq 2^{-n} \end{aligned}$$

(2) $\Rightarrow$ (3) : Obvious.
(3) $\Rightarrow$ (1) : Same as the proof (2) $\Rightarrow$ (1) since the value of y used in this proof, namely 2^{n+1}, is a natural number.

A.2 Proof of Proposition 4

Proof. The proof is similar to that of Proposition 3 only replacing computability by polynomial time computability and observing: (1) multiplication can be done in polynomial time and (2) the procedure given in Proposition 3 is actually polynomial time computable in terms of the precision parameter n.

A.3 Proof of Proposition 5

Proof. (1) $\Rightarrow$ (2) : Define g on integer points as follows,

$$g(x,y) = \begin{cases} 0 & y = 0 \\ yf(\frac{x}{y}) & x \leq y, y \geq 1 \\ yf(1) & otherwise \end{cases} \tag{28}$$

and piecewise linear for non-integer values. Clearly, g is computable and satisfies (6).

(2) $\Rightarrow$ (1) : For simplicity assume $\epsilon = 1$. Since f is Lipschitz there exists a 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 computable, there exists an oracle machine $N^{()}$ which 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 $N^{()}(0)$,
4. Whenever $N^{()}$ 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 $2^{-n'}e$.

Then

$$\begin{aligned} &|e - g(k_1, 2^{n'})| \leq 1 \\ &|2^{-n'}e - 2^{-n'}g(k_1, 2^{n'})| \leq 2^{-n'} \end{aligned} \tag{29}$$

From (6) with $\epsilon = 1$

$$\begin{aligned} &|g(k_1, 2^{n'}) - 2^{n'}f(\frac{k_1}{2^{n'}})| \leq 1 \\ &|2^{-n'}g(k_1, 2^{n'}) - f(\frac{k_1}{2^{n'}})| \leq 2^{-n'} \end{aligned} \tag{30}$$

From Inequalities (29) and (30) we have

$$|2^{-n'}e - f(\frac{k_1}{2^{n'}})| \leq 2^{-(n'-1)} \tag{31}$$

From the fact that f is Lipschitz we have

$$|f(\frac{k_1}{2^{n'}}) - f(x)| \leq 2^a 2^{-n'} = 2^{-(n+2)} \tag{32}$$

From Inequalities (31) and (32) we have

$$|2^{-n'}e - f(x)| \leq 2^{-(n'-1)} + 2^{-(n+2)} \leq 2^{-n} \tag{33}$$

This completes the proof that f is computable.

A.4 Proof of Proposition 6

Proof. Similar to the proof of Proposition 5 replacing computability by polynomial time computability.

A.5 Proof of Theorem 1

Proof. (1) $\Rightarrow$ (2) : by Proposition 5, there exists some computable g such that (6) holds with $\epsilon = \frac{3}{4}$. Computing g with precision $1/2$, one can easily build some total recursive function $h : \mathbb{N}^2 \to \mathbb{N}$ such that $|h(x,y) - \lfloor g(x,y) \rfloor| \leq 1$. Then h is total recursive, and hence by 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

$$\forall x, y \in \mathbb{N} \colon |\tilde{g}(x,y) - \lfloor g(x,y) \rfloor| \leq 1 + \frac{1}{4} = \frac{5}{4} \tag{34}$$

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

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y \colon |\tilde{g}(x,y) - yf(\frac{x}{y})| \leq \frac{9}{4} + \frac{3}{4} = 3 \tag{35}$$

(2) $\Rightarrow$ (1) : This follows from Proposition 5 with $\epsilon = 3$, observing that functions from $\mathcal{C}$ are assumed computable.

A.6 Proof of Theorem 2

Proof. Similar to the proof of Theorem 1 replacing computability by polynomial time computability.