A Survey of Recursive Analysis and Moore's Notion of Real Computation

Walid Gomaa
INRIA Nancy Grand-Est Research Centre, France,
Faculty of Engineering, Alexandria University, Egypt
walid.gomaa@loria.fr

Abstract

The theory of analog computation aims at modeling computational systems that evolve in a continuous manner. Unlike the situation with the discrete setting there is no unified theory of analog computation. There are several proposed theories, some of them seem quite orthogonal. Some theories can be considered as generalizations of the Turing machine theory and classical recursion theory. Among such are recursive analysis and Moore's class of recursive real functions. Recursive analysis was introduced by A. Turing [1936], A. Grzegorczyk [1955], and D. Lacombe [1955]. Real computation in this context is viewed as effective (in the sense of Turing machine theory) convergence of sequences of rational numbers. In 1996 Moore introduced a function algebra that captures his notion of real computation; it consists of some basic functions and their closure under composition, integration and zero-finding. Though this class is inherently unphysical, much work have been directed at stratifying, restricting, and comparing it with other theories of real computation such as recursive analysis and the $GPAC$. In this article we give a detailed exposition of recursive analysis and Moore's class and the relationships between them.

1 Introduction

Analog computation is a computational paradigm that attempts to model systems whose internal states are continuous rather than discrete. Unlike the case of discrete computation, which has enjoyed a kind of uniformity and conceptual universality, there have been several approaches to analog computations some of which are not even comparable. A survey of the wide spectrum of analog models is written by P. Orponen in 1997 [30]. An up-to-date version of Orponen's has been written by O. Bournez and M. Campagnolo in 2008 [29]. V. Blondel and J. Tsitsiklis wrote a more specialized survey about the role of classical computational complexity in systems and control theory [1]. A dedicated survey about the class of Blum-Shub-Smale (BSS) models was written by K. Meer and C. Michaux [24]; this survey essentially focuses on the complexity-theoretic aspects of these models. A less detailed survey of the main BSS model is given in [13]; this article focuses on extending the Grzegorczyk hierarchy to the reals through that model.

So much of the current research in this area have been directed at developing a unified theory of continuous computation through discovering relationships among the different models. Some recent developments towards that direction can be found in [15, 14, 12, 2]. There have also been work relating analog models to discrete recursive classes (see [5, 6, 8, 7]). In this article we give a detailed survey on two theories of continuous computation: recursive analysis and Moore's function algebra, and the relationships between them.

One of the earliest theoretical models of continuous-time computation is the $GPAC$ (General Purpose Analog Computer). It was introduced by C. Shannon in 1941 [33] as a mathematical model of the differential analyzer which was a kind of a universal analog machine built at MIT in 1931. Since then the model has been refined in a series of papers by Pour-El 1974 [31], Lipshitz and Rubel 1987 [22], Graça and Costa 2003 [15], and Graça 2004 [14]. One of the major extensions of the $GPAC$ was proposed by C. Moore in 1996 [25] where he defined recursiveness of real functions in terms of a function algebra. This function algebra is constructed as a direct analogy of classical recursion theory, for example, the scheme of integration is introduced as a replacement of primitive recursion and a version of the minimalization scheme is given for real functions. Moore's class is inherently unphysical essentially due to the unrealizability of the minimalization operator.

Recursive analysis was introduced by A. Turing [34], A. Grzegorczyk [17], and D. Lacombe [21]. It is a model of analog computation based on classical computability theory that, unlike Moore's recursive class, takes more direct and realistic approach by appealing to the notion of *mechanism* and hence physical realizability. The computation model is essentially an *oracle Turing machine* which is a normal machine equipped with function oracles.

This article gives a detailed exposition of the two aforementioned theories of real computation and the relationships between them. Section 1 is an introduction. Section 2 introduces Moore's notion of recursiveness along with examples and basic results. Section 3 defines the notions of computability and complexity of real functions in the context of

recursive analysis along with examples and basic results. Section 4 shows how these two approaches are interrelated. Section 5 concludes the paper.

2 Moore's Recursive Class of Real Functions

As an extension to the $GPAC$, C. Moore proposed in 1996 [25] a class of functions defined in analogy with classical recursion theory and corresponds to a conceptual analog computer operating in continuous time [29]. We will call this class *Moore's recursive functions* or $\mathcal{MR}ec$ for short. A general discussion about the motivations behind Moore's notion of real computation can be found in [28]. The following version of $\mathcal{MR}ec$ is based on that given in [5].

Definition 1 ($\mathcal{MR}ec$ functions)**.**

$$\mathcal{MR}ec = [0, 1, -1, U; Comp, \int, \mu] \tag{1}$$

where

1. $0, 1, -1$ are constants which can be represented by zero-ary constant functions.
2. U is the set of projection functions.
3. $Comp$ is the *composition* operator.
4. $\int$ is the *integration operator*. Assume functions $g, h \in \mathcal{MR}ec$. Assume the differential equation

$$\begin{aligned} f(0, \bar{y}) &= g(\bar{y}) \\ \partial_x f(x, \bar{y}) &= h(x, \bar{y}, f(x, \bar{y})) \end{aligned} \tag{2}$$

   has a unique solution over a domain that includes 0. Then define a new function $f\colon \mathcal{D} \subseteq \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ as that unique solution over the maximal domain that includes 0.

5. μ is the *μ-recursion* or *zero-finding* operator. Given a function $g \in \mathcal{MR}ec$ with $g\colon \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$, define a new function $f\colon \mathbb{R}^n \to \mathbb{R}$ as follows

$$f(\bar{y}) = \mu_x g(x, \bar{y}) \triangleq \begin{cases} x^- = sup\{x \in \mathbb{R}^{\leq 0}\colon g(x^-, \bar{y}) = 0\} & -x^- < x^+ \\ x^+ = inf\{x \in \mathbb{R}^{\geq 0}\colon g(x^+, \bar{y}) = 0\} & -x^- \geq x^+ \end{cases}$$

   Note that the μ-operator returns a limit point which itself might not be a root of g, however in this case, it is the limit of an accumulating sequence of roots.

The analog schemas of integration and zero-finding are meant to correspond to primitive recursion and zero-finding over $\mathbb{N}$. The original definition of $\mathcal{MR}ec$ [25] was somewhat ambiguous. For example, the integration operator was not defined precisely, the zero-finding operator was too general and un-intuitive, and composing with undefined functions or values is allowed. With excluding the zero-finding operator, some authors have adopted a restricted version of integration. For example, M. Campagnolo, C. Moore, and J. Costa [7] used some sort of integration that generates solutions of differential equations where the solutions and their derivatives are continuous everywhere (no singularities). The resulting class captures $GPAC$-computable functions. They then extended this class by adding a k-continuously differentiable extension of the Heaviside step function. This gives a differentiable way to test inequalities without introducing discontinuities. They also redefined the integration operator so as to generate functions that are solutions of quasi-linear differential equations. The new class is a proper extension of $GPAC$; its functions are actually piecewise $GPAC$-computable, closed under iteration, and contain the primitive recursive discrete functions. B. Loff, J. Costa, and J. Mycka [23] replaced the zero-finding by an *infinite limit* operator. The resulting class is stratified based on the number of nested limits required to define a function. Such a hierarchy has been proven to be proper; its first level contains the discrete primitive recursive functions, the second level contains the partial recursive functions, and the following level contains the solution to the halting problem. Recently, A. Kawamura [19] tried to redefine $\mathcal{MR}ec$ exploring several ways to remove the ambiguities in the original definition, in particular the behavior of the operators on partial functions. Further work on $\mathcal{MR}ec$ is mentioned below when discussing the relationships with recursive analysis.

$\mathcal{MR}ec$ is a very rich class. As a start off it can generate simple arithmetic functions such as multiplication which can be defined as the solution of the differential equation

$$\begin{aligned} f(0, y) &= 0 \\ \partial_x f(x, y) &= y \end{aligned}$$

An important observation about the initial condition of this definition is that, using the composition schema, 0 can be obtained as the result of multiplying 0 by an undefined value. Trigonometric functions are in $\mathcal{MR}ec$, for example, $\tan x$ can be generated by

$$f(0) = 0$$
$$\partial_x f(x) = f^2(x) + 1$$

Using the μ operator, non-smooth functions are introduced into the class. For example, $|x|$, which has discontinuous derivative, can be defined by

$$|x| \triangleq \mu_y(x^2 - y^2) \tag{3}$$

It can also generate discontinuous functions such as the *Kronecker δ-function* which is 0 everywhere except at the origin.

$$\delta(x) \triangleq 1 - \mu_y[(x^2 + y^2)(y - 1)] \tag{4}$$

One of the interesting and surprising properties of the $\mathcal{MR}ec$ class is the use of what Moore called the *compression trick* which can be used to transform potentially indefinite search over the whole real line into a search over a finite interval that is guaranteed to eventually have a root at either or both of its boundaries. However, this trick is only possible when multiplication is defined as given above. This is exemplified as follows.

Example 1. $\mathbb{Z} \in \mathcal{MR}ec$ as its characteristic function can be defined by

$$\chi_{\mathbb{Z}}(x) \triangleq \delta(\sin \pi x) \tag{5}$$

Using this we try to show that $\mathbb{Q} \in \mathcal{MR}ec$. The first attempt is as follows

$$\tilde{\chi}_{\mathbb{Q}}(x) \triangleq \chi_{\mathbb{Z}}(\mu_y[(1 - \delta(y))\chi_{\mathbb{Z}}(y)\chi_{\mathbb{Z}}(xy) - 1]) \tag{6}$$

If x is rational, then $x = \frac{i}{j}$ for $i, j \in \mathbb{Z}$ with $j \neq 0$. Then $y = j$ would be a root of the expression $(1-\delta(y))\chi_{\mathbb{Z}}(y)\chi_{\mathbb{Z}}(xy)-1$ and $\mu_y[\cdots]$ returns j. Hence, $\tilde{\chi}_{\mathbb{Q}}$ outputs 1 which is the supposed value in this case. Note that the term $(1 - \delta(y))$ is used to exclude the case $y = 0$ from the search process. However, if x is not rational the search would go on forever and $\tilde{\chi}_{\mathbb{Q}}$ would be undefined. The compression trick can be used to have a total version of the characteristic function as follows. Consider the function $f(x) = \tan(x)$, which maps $(-\frac{\pi}{2}, \frac{\pi}{2})$ to $(-\infty, \infty)$. Consider the following definition

$$\chi_{\mathbb{Q}}(x) \triangleq 1 - \delta(\frac{\pi}{2} - \mu_y[((1 - \delta(\tan y))\chi_{\mathbb{Z}}(\tan y)\chi_{\mathbb{Z}}(x \tan y) - 1)(\frac{\pi}{2} - y)]) \tag{7}$$

Again the term $(1 - \delta(\tan y))$ is used to exclude the case $\tan y = 0$ from the search process, otherwise $y = 0$ is always a correct answer whether or not $x \in \mathbb{Q}$. If $x \in \mathbb{Q}$, then $x = \frac{i}{j}$. Then $y = \tan^{-1} j$ with $|y| < \frac{\pi}{2}$ will be returned by the search process and $\chi_{\mathbb{Q}}$ will output 1 in this case. If $x \notin \mathbb{Q}$, then the search will continue until y reaches $\frac{\pi}{2} = \tan^{-1} \infty$ which is a root of the μ-expression due to the existence of the factor $(\frac{\pi}{2} - y)$ and eventually $\chi_{\mathbb{Q}}$ will output 0.

The trick done in the previous example to limit the search process to within a finite interval can be generalized to yield the following proposition.

Proposition 1. Every partial function in $\mathcal{MR}ec$ can be converted into a total function that is also in $\mathcal{MR}ec$.

Proof. Assume a function $f(x, \bar{y}) \in \mathcal{MR}ec$. Define an operator η as follows:

$$\eta_x[f(x, \bar{y})] = \begin{cases} 1 & \exists x f(x, \bar{y}) = 0 \\ 0 & ow \end{cases} \tag{8}$$

$\mathcal{MR}ec$ is closed under η for it can be simulated by the following expression

$$1 - \delta(\frac{\pi}{2} - \mu_x[f(\tan x, \bar{y})(\frac{\pi}{2} - x)]) \tag{9}$$

Hence, any partial function $g(\bar{y}) \in \mathcal{MR}ec$ defined by $\mu_x[f(x, \bar{y})]$ can be converted to a total function defined by $\eta_x[f(x, \bar{y})] \cdot \mu_x[f(x, \bar{y})]$. Two subtle points still to be clarified in this latter definition: (1) in case $\mu_x[f(x, \bar{y})]$ is undefined the value of the expression would be 0 as is indicated by the definition of multiplication above and (2) if the value of the expression is 0, then either f does not have a zero or have a zero at $x = 0$, these can be distinguished by testing $f(x, \bar{y}) = 0?$. □

So in an obvious sense every function in $\mathcal{MR}ec$ is total. Moore's class is actually too powerful, it contains non-computable discrete functions, the whole of the arithmetical hierarchy, and the whole of the analytical hierarchy.

Based on the minimum number of nested applications of the μ operator necessary to generate $\mathcal{MR}ec$ functions the class can be stratified. This induces what is called the *μ-hierarchy* which is believed to be proper (it is proper in the discrete recursive case). For example, $\mathbb{Q} \in \mathcal{MR}ec_2 \backslash \mathcal{MR}ec_1$ [25] ($\mathcal{MR}ec_i$ indicates the i^{th} level of this hierarchy). The total recursive functions over $\mathbb{N}$ have extensions in $\mathcal{MR}ec_2$ and the partial ones have extensions in $\mathcal{MR}ec_3$. For larger values of i, $\mathcal{MR}ec_i$ contains various levels of the analytical hierarchy but no upper bounds for these classes are known [5]. The integration schema can also be restricted and stratified for finer investigation of $\mathcal{MR}ec$. We will look at that below when discussing the relationship between Moore's class and recursive analysis.

To the best of our knowledge there is no well-developed notion of complexity (at least from the physical perspective) for $\mathcal{MR}ec$. This is in contrast with recursive analysis as will be seen below. Note that there are finite number of basic functions in $\mathcal{MR}ec$ and the operations are finitary, hence $\mathcal{MR}ec$ is a countable class. So actually most real functions are uncomputable even with respect to this powerful notion of recursiveness.

3 Recursive Analysis as an Approach to Real Computation

Recursive analysis was introduced by A. Turing [34], A. Grzegorczyk [17], and D. Lacombe [21]. The machine associated with this model of computation is essentially an extension of the traditionally known Turing machine that computes functions over $\mathbb{N}$. Like Moore's notion of real computation, recursive analysis is an offspring of classical computability theory. However, it takes more direct and realistic approach by appealing to the notion of *mechanism* and hence physical realizability.

3.1 Representation of $\mathbb{R}$

In order to perform a mechanical computation over the real numbers we must first have a *representation* of such numbers. Almost all real numbers are infinite objects, whereas Turing machines are inherently discrete structures, so machines can only have indirect access to real numbers through successively accurate finite approximations. Given $x \in \mathbb{R}$, there are several representations of x that can provide such approximations, among which are the following.

1. Binary expansion: x is represented by a function $\psi_x \colon \mathbb{N} \cup \{-1\} \to \mathbb{N} \cup \{-1\}$, where $\psi_x(-1) \in \{-1, 1\}$, $\psi_x(0) \in \mathbb{N}$, and $\psi_x(k) \in \{0, 1\}$ for every $k \geq 1$. Then
$$x = \psi_x(-1) \cdot \left(\psi_x(0) + \sum_{k \geq 1} \psi_x(k) \cdot 2^{-k} \right) \tag{10}$$
2. Left cut: x is represented by the set $L_x = \{r \in \mathbb{Q} \colon r < x\}$.
3. Cauchy sequence: x is represented by a function $\varphi_x \colon \mathbb{N} \to \mathbb{Q}$, such that $\{\varphi_x(n)\} \rightsquigarrow x$.

It should be noted that all representations induce the same computability concept, that is, they give the same class of computable real numbers. Examples of such numbers include the rationals, algebraic numbers such as $\sqrt{2}$, and transcendental numbers such as π and e. However, on the sub-computable and complexity-theoretic levels they induce different classes. For the remaining part of this article the *Cauchy representation*. However, two refinements are added that are in accordance with using binary alphabet for encoding numbers.

1. Instead of $\mathbb{Q}$, the codomain of φ_x will be the set of *dyadic rationals* $\mathbb{D} = \{\frac{a}{2^b} \colon a \in \mathbb{Z}, b \in \mathbb{N}\}$. These are the rational numbers with *finite binary encoding*.
2. φ_x has a *binary convergence rate*, that is for every $n \in \mathbb{N}$,
$$|x - \varphi_x(n)| \leq 2^{-n} \tag{11}$$

For any $x \in \mathbb{R}$ let CF_x denote the set of Cauchy functions for x that satisfy the above conditions.

3.2 Computability and complexity of real numbers

Assuming the Cauchy representation, real numbers are then first order objects. Hence, the notions of computability and complexity over $\mathbb{N}$ can be naturally applied to real numbers.

Definition 2 (Computability of real numbers)**.** Let $x \in \mathbb{R}$. We say that x is *computable* if CF_x contains a computable Cauchy function for x. Here computability is taken in the traditional discrete sense since the domain and codomain of Cauchy functions are sets of objects with finite representation.

The set of computable real numbers is countable. It can easily be shown that this class forms a *real closed field*.

Example 2. This shows how to compute $\sqrt{2}$. Define a function $g\colon \mathbb{N} \to \mathbb{N}$ by $g(n) = k$, where k is the least number satisfying

$$k^2 \leq 2^{2n+1} \leq (k+1)^2$$

Define the following function $\varphi_{\sqrt{2}}\colon \mathbb{N} \to \mathbb{D}$

$$\varphi_{\sqrt{2}}(n) = g(n)2^{-n} \tag{12}$$

Then

$$\begin{aligned}
&(\varphi_{\sqrt{2}}(n))^2 = k^2 2^{-2n} \leq 2^{2n+1}2^{-2n} \leq (k+1)^2 2^{-2n} \\
&k^2 2^{-2n} \leq 2 \leq (k+1)^2 2^{-2n} \\
&k2^{-n} \leq \sqrt{2} \leq (k+1)2^{-n}
\end{aligned}$$

Hence $\sqrt{2}$ lies in the interval $[k2^{-n}, (k+1)2^{-n}]$. This interval has length 2^{-n}, hence $\varphi_{\sqrt{2}}$ is a Cauchy sequence for $\sqrt{2}$.

Next the complexity notion over $\mathbb{R}$ is defined which is again not very different from the corresponding notion over $\mathbb{N}$.

Definition 3 (Complexity of real numbers)**.** Let $x \in \mathbb{R}$ and let $\mathcal{C}$ be a complexity class of $\mathbb{N}$-functions. We say that x is *computable in time bounded by* $\mathcal{C}$ or has *time complexity* $\mathcal{C}$ if CF_x contains a Cauchy function whose computation time is bounded by $\mathcal{C}$.

As mentioned above different representations of $\mathbb{R}$ may not induce the same complexity concept (with the Cauchy representation giving the largest classes [37, 36]). In particular we have the following.

Theorem 1 (Theorem 2.8 in [20])**.** Let P_X denote the class of real numbers that are computable in polynomial time assuming representation X. Then

$$P_{LC} = P_{BE} \subsetneq P_{CF} \tag{13}$$

3.3 Computability and complexity of real functions

In this subsection we discuss the more interesting notion of computation of $\mathbb{R}$-functions. For simplicity we only consider scalar real valued functions $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$. From the above discussion it is clear that $\mathbb{R}$-functions are higher-order objects, hence we need a higher order Turing machine to model their computation. The idea is that a real function is computable by a Turing machine if it can be approximated to the appropriate precision given approximations to the inputs of the function [10]. The following gives a more formal definition, for the sake of simplicity we restrict to functions over compact domains.

Definition 4 (Recursive analysis-1)**.** Assume a function $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$, where $\mathcal{D}$ is compact. We say that f is computable if there exists a function-oracle Turing machine $M^{()}$ such that for every $(x_1, \ldots, x_k) \in \mathcal{D}$, for every $\varphi_1 \in CF_{x_1}, \ldots, \varphi_k \in CF_{x_k}$, and for every $n \in \mathbb{N}$, the following holds:

$$|M^{\varphi_1,\ldots,\varphi_k}(n) - f(x_1, \ldots, x_k)| \leq 2^{-n} \tag{14}$$

For inputs outside the domain of the function, the behavior of $M^{()}$ is undefined.

Using proper encodings of of the dyadic rationals, Cauchy functions can be viewed as having codomain $\mathbb{N}$. Hence, oracle machines can be considered as Type-2 functionals over $\mathbb{N}$. This gives rise to the following equivalent machine-free definition of recursive analysis.

Definition 5 (Recursive analysis-2)**.** Assume a function $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$, where $\mathcal{D}$ is compact. We say that f is *computable* if there exists a *recursive functional* $F\colon (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N} \to \mathbb{N}$ that satisfies the following: for all $x_1, \ldots, x_k \in \mathbb{R}$ and for all $\varphi_1, \ldots, \varphi_k \in \mathbb{N}^{\mathbb{N}}$, if $\{\langle \varphi_1(n), \ldots, \varphi_k(n)\rangle\}_n \rightsquigarrow \langle x_1, \ldots, x_k\rangle$, then $\{F(\varphi_1, \ldots, \varphi_k, n)\}_n \rightsquigarrow f(x_1, \ldots, x_k)$.

Based on the computability power of the functional F, the class of computable real functions can be stratified. Let $\mathcal{E}(\mathbb{R})$ denote the set of *elementary computable* real functions over compact domains. So in terms of Definition 4, a function f belongs to $\mathcal{E}(\mathbb{R})$ if it can be computed by an oracle Turing machine $M^{()}$ whose time bound is elementary, that is the computation time of $M^{()}$ is bounded by $\lambda x.2^{[j]}(x)$. Likewise, let $\mathcal{PR}(\mathbb{R})$ denote the class of *primitive recursive* real functions, and $\mathcal{R}ec(\mathbb{R})$ denote the class of *recursive* real functions.

A derived notion of continuity that plays an essential role in the investigation of real computation is the *modulus of continuity*.

Definition 6 (Modulus of continuity)**.** Assume a function $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$ such that $\mathcal{D}$ is compact. We say that f has a *modulus of continuity* if there exists a monotonically increasing function $m\colon \mathbb{N} \to \mathbb{N}$ such that for all $\bar{x}, \bar{y} \in \mathcal{D}$ and for all $n \in \mathbb{N}$: if $\|\bar{x} - \bar{y}\| \leq 2^{-m(n)}$, then $|f(\bar{x}) - f(\bar{y})| \leq 2^{-n}$.

The modulus of continuity is a precise algebraic way to measure the *smoothness* of a function and is very strongly related to computability over $\mathbb{R}$ as indicated by the two following theorems.

Theorem 2 (Characterization of computable real functions over compact domains)**.** Assume a function $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$ such that $\mathcal{D}$ is compact. Then f is *computable* if and only if there exist two *computable* functions m and ψ such that

1. $m\colon \mathbb{N} \to \mathbb{N}$ is a modulus of continuity for f,
2. $\psi\colon \mathcal{D} \cap \mathbb{D}^k \times \mathbb{N} \to \mathbb{D}$ is an *approximation function* for f, that is for all $\bar{d} \in \mathcal{D} \cap \mathbb{D}^k$ and all $n \in \mathbb{N}$, $|\psi(\bar{d}, n) - f(\bar{d})| \leq 2^{-n}$.

Proof. For simplicity assume f is unary. Assume f has computable functions m and ψ. Let $M^{()}$ be a function-oracle Turing machine. Assume $x \in \mathcal{D}$, and let $\varphi_x \in CF_x$. Assume $n \in \mathbb{N}$, then $M^{\varphi_x}(n)$ can be programmed to do the following:

1. M^{φ_x} computes $m(n+1)$ and writes it on the oracle tape,
2. the oracle responds with $d = \varphi_x(m(n+1))$,
3. M^{φ_x} computes and outputs $e = \psi(d, n+1)$.

Then

$$\begin{aligned} |e - f(x)| &\leq |e - f(d)| + |f(d) - f(x)| \\ &\leq 2^{-(n+1)} + |f(d) - f(x)|, \qquad \text{by definition of } \psi \\ &\leq 2^{-(n+1)} + 2^{-(n+1)}, \qquad \text{definition of } m \text{ and } |d - x| \leq 2^{-m(n+1)} \\ &\leq 2^{-n} \end{aligned}$$

Hence, $M^{()}$ computes f. Proof in the other direction is a little bit more complex and depends on using the Heine-Borel Theorem to have a finite covering of all bounded neighborhoods of the points in $\mathcal{D}$. The interested reader can refer to the proof of Theorem 2.13 in [20]. □

It is evident from the last proof that the modulus of continuity can be used in two interrelated aspects: (1) it gives machine $M^{()}$ a way to access the real input x by asking the oracle the proper questions and (2) it gives an estimate of how good the approximating value $f(d)$ is to the desired value $f(x)$ and consequently an estimate of how good the computed approximation $M^{\varphi_x}(n)$ is to the desired value $f(x)$. The second theorem that relates real computability to continuity is the following.

Theorem 3. Assume $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$. If f is computable, then f must be continuous.

Turing machines are finite objects, hence the class of computable functions in the sense of recursive analysis, $\mathcal{R}ec(\mathbb{R})$, is countable. In some sense this class is much smaller than Moore's. However, it is much more amenable to physical realization.

There have been much work done investigating the complexity-theoretic properties of computable $\mathbb{R}$-functions in the context of recursive analysis. There are subtle issues here as compared with the complexity of $\mathbb{N}$-functions. As is typical in discrete complexity theory, time and space complexity measures are calculated in terms of the length of the *binary encoding* of the input. However, for the computation model introduced in Definition 4 complexity measures make sense only if calculated in terms of the *unary representation* of the machine input n. This is justified by the fact that n actually represents the required *precision* of the approximating algorithm so it is in a sense a lower bound on the length

of the output that will be produced by the machine. Another subtle issue is whether the space used on the oracle tape be accounted for when calculating the space complexity of the machine. As far as we know the general consensus is that such space should not be counted because this is where an approximation to the actual relevant real input is written so in a strong sense this is the space taken by the input itself.

Polynomial time computability is the most investigated complexity-theoretic notion for real functions over compact domains.

Definition 7. (Polynomial time computability) Assume a function $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$, where $\mathcal{D}$ is compact. We say that f is *polynomial time computable* if there exists a time-bounded function-oracle Turing machine $M^{()}$ that computes f in the sense of Definition 4. Furthermore, the computation time of $M^{()}(n)$ is bounded by $p(n)$ for some polynomial function p.

Similar to Theorem 2, polynomial time computability can be characterized in an algebraic way as indicated in the following theorem.

Theorem 4 (Characterization of poly-time computable real functions over compact domains)**.** Assume a function $f\colon \mathcal{D} \subseteq \mathbb{R}^k \to \mathbb{R}$ such that $\mathcal{D}$ is compact. Then f is *polynomial time computable* if and only if there exist two functions m and ψ such that

1. $m\colon \mathbb{N} \to \mathbb{N}$, is a polynomial function and is a modulus of continuity for f,
2. $\psi(\bar{d}, n)$ is an approximation function for f, its computation time is bounded by $p(|\bar{d}|, n)$ for some polynomial function p.

As far as we know there has not been so much progress in the complexity-theoretic investigation of arbitrary real functions (as compared with computability, for example). Some research has been done that investigate open problems in classical complexity in the context of real computation. For example, Mycka and Costa [26] propose two classes of real computable functions such that their inequality implies $P \neq NP$. More generally, a part of Costa and Mycka's program [27, 28] is to use recursion theory on the reals to establish connections between classical discrete computability/complexity and mathematical analysis [29].

4 Tieing $\mathcal{MR}ec$ with $\mathcal{R}ec(\mathbb{R})$ through Function Algebras

In this section $\mathcal{MR}ec$ is weakened in such a way that makes it feasible to compare with recursive analysis and classical recursion theory. First, the unphysical μ-operator is excluded. Then the integration operator is restricted as follows.

Definition 8. (Linear integration) Let LI be the operator that takes three real functions g,h, and r and generates a new function $f = LI(g, h, r)$ such that f is the maximal solution (over maximal domain that includes 0) of the *linear partial differential equation*

$$\begin{aligned} f(0, \bar{y}) &= g(\bar{y}) \\ \partial_x f(x, \bar{y}) &= h(x, \bar{y}) f(x, \bar{y}) + r(x, \bar{y}) \end{aligned}$$

Notice that such a system has always a unique solution.

Definition 9. (Elementary computable functions over $\mathbb{N}$) Let $\mathcal{E}$ denote the class of *elementary time computable discrete functions*, that is, the class of functions whose computation times are bounded by a fixed iteration of the exponential function. It can be characterized by the following function algebra (introduced by Kalmár [18]).

$$\mathcal{E} \equiv [0, s, \ominus, U; Comp, BSum, BProd] \tag{15}$$

where s is the successor function and $x \ominus y = \max\{x - y, 0\}$ is the *cutoff subtraction*. $BSum$ is the *bounded sum operator*, it takes a function $g\colon \mathbb{N} \times \mathbb{N}^k \to \mathbb{N}$ and generates a function $f\colon \mathbb{N} \times \mathbb{N}^k \to \mathbb{N}$ such that

$$f(x, \bar{y}) = \sum_{z=0}^{x} g(z, \bar{y})$$

$BProd$ is the *bounded product operator* and is defined similarly.

The class $\mathcal{E}$ coincides with the third level of the *Grzegorczyk hierarchy* which is a proper hierarchy of sub-primitive recursive functions whose limit is the primitive recursive class itself [16, 11]. From the programmer's perspective at most two nested FOR-NEXT loops are required to compute a function of class $\mathcal{E}$ [4], hence it is not closed under iteration.

In analogy with the function algebra capturing $\mathcal{E}$, Campagnolo, Moore, and Costa proposed following class of real functions [6, 8, 5].

Definition 10. Define the following function algebra

$$\mathcal{L} \triangleq [0, 1, -1, \pi, U, \theta_3; Comp, LI] \tag{16}$$

where θ_3 is defined as follows

$$\theta_3(x) = \begin{cases} x^3 & x \geq 0 \\ 0 & ow \end{cases} \tag{17}$$

θ_3 is a smooth extension of the *Heaviside function*; it gives a differentiable way to *sense inequalities* without introducing discontinuities.

It can be easily seen that all functions in $\mathcal{L}$ are C^2 (two-times continuously differentiable). Here are some examples.

Example 3.

1. $f(x) = e^x$: it is the solution of the linear differential equation
$$f(0) = 1$$
$$f' = f$$
2. Fixed iteration of exponentiation, $f_j(x) = f^{[j]}(x)$: obtained by composing f j times with itself.
3. The trigonometric functions $\sin x$ and $\cos x$ which are solutions to the following system of linear differential equations
$$h_1(0) = 0, \qquad h_2(0) = 1$$
$$h_1' = h_2, \qquad h_2' = -h_1$$

Let $dp(\mathcal{L}) = \{f \colon \mathbb{N}^k \to \mathbb{N} \colon \exists \tilde{f} \in \mathcal{L} \textit{ such that } \tilde{f} \upharpoonright \mathbb{N}^k = f\}$. That is $dp(\mathcal{L})$ is the subclass of functions in $\mathcal{L}$ that preserve $\mathbb{N}$. Remember that $\mathcal{E}(\mathbb{R})$ denotes the class of elementary computable real functions. $\mathcal{L}$ is partially related to this class as indicated by the following proposition.

Proposition 2 ([8, 5])**.** $\mathcal{L} \subsetneq \mathcal{E}(\mathbb{R})$

Hence $\mathcal{L}$ is too weak to capture the whole of $\mathcal{E}(\mathbb{R})$. However, any function in $\mathcal{E}(\mathbb{R})$ can be approximated inside $\mathcal{L}$ within any arbitrary small error [10]. So in a natural sense $\mathcal{L}$ is dense in $\mathcal{E}(\mathbb{R})$ and what is missing of $\mathcal{L}$ is just the ability to reach the limits as will be seen in the next subsection. For now $\mathcal{L}$ can be related exactly to the discrete setting as indicated by the following theorem.

Theorem 5 ([8, 10])**.** $dp(\mathcal{L}) = \mathcal{E}$

Proof. The proof of $\mathcal{E} \subseteq dp(\mathcal{L})$ can be done by induction on the construction tree of functions in $\mathcal{E}$ using the operations of $\mathcal{L}$ at each step. For the other direction $dp(\mathcal{L}) \subseteq \mathcal{E}$ the proof given in [8] applies induction on the construction tree of functions in $\mathcal{L}$. However, instead of the intuitive appeal of using the operations of the function algebra $\mathcal{E}$ at each step the fact that $\mathcal{L}$ is elementarily computable is used to build Turing machines that would output appropriately close approximations. The tricky part here is that given a function $f \in \mathcal{L}$ that preserves $\mathbb{N}$, it is not necessary that the functions used to construct f also preserve $\mathbb{N}$. Another proof for this latter direction is given in [10] which is purely algebraic. It uses a notion of interpreting a function algebra into another through an *approximation relation*. Using a function algebra over $\mathbb{Q}$ as an intermediary $\mathcal{L}$ can be interpreted inside $\mathcal{E}$ within a close enough approximation and the theorem follows almost immediately. □

4.1 Enriching $\mathcal{L}$

From the computability perspective $\mathcal{L}$ provided a cornerstone of research that on one hand investigates the relationships among different models of continuous computation and on the other hand investigates the relationships of these models to the discrete setting. On the finer complexity level though $\mathcal{L}$ might be too strong for such an investigation. In this subsection we will enrich $\mathcal{L}$ with different operators and see the effect on the computability power of the resulting classes.

Definition 11. (Limit schema [3, 4]) Assume a function $g\colon \mathbb{R} \times \mathcal{D} \to \mathbb{R}$ such that $\mathcal{D} \subseteq \mathbb{R}^k$ is a product of compact intervals. Assume g satisfies the following:

1. g is continuously differentiable,
2. there exist two functions $\alpha\colon \mathcal{D} \to \mathbb{R}^{\geq 0}$ and $\beta\colon \mathcal{D} \to \mathbb{R}^{>0}$ such that for all x with $|x| \geq \|\bar{y}\|$, the following holds

$$|\partial_x g(x, \bar{y})| \leq \alpha(\bar{y}) e^{-x\beta(\bar{y})} \tag{18}$$

This means that for any fixed $\bar{y}$, the function $g_{\bar{y}}(x) = g(x, \bar{y})$ is bounded. Hence, the $\lim_{x\to\infty} g_{\bar{y}}(x) < \infty$.

Define a new function $f = Lim(g, \alpha, \beta)\colon \mathcal{D} \to \mathbb{R}$ as follows

$$f(\bar{y}) = \lim_{x\to\infty} g(x, \bar{y}) \tag{19}$$

Definition 12. Extend the function algebra $\mathcal{L}$:

$$\widetilde{\mathcal{L}} \triangleq [0, 1, -1, U, \theta_3; Comp, LI, Lim] \tag{20}$$

$\widetilde{\mathcal{L}}$ is a proper extension of $\mathcal{L}$ as indicated by the following lemma.

Lemma 1. $\mathcal{L} \subsetneq \widetilde{\mathcal{L}}$

Proof. Consider the function $f\colon x \mapsto \frac{1}{x}$. f can not be generated as the solution of a linear differential equation, hence $f \notin \mathcal{L}$. It can be generated in $\widetilde{\mathcal{L}}$ as follows. Consider the following linear differential equation.

$$\begin{aligned} g(0, y) &= 1 \\ \partial_x g(x, y) &= -y g(x, y) + 1 \end{aligned}$$

Solving this results in $g(x, y) = \frac{1-e^{-xy}}{y}$. To obtain f in $\widetilde{\mathcal{L}}$ apply the limit schema

$$f(y) = \lim_{x\to\infty} g(x, y) = \lim_{x\to\infty} \frac{1 - e^{-xy}}{y} = \frac{1}{y} \tag{21}$$

Hence, $f \in \widetilde{\mathcal{L}}$. Next we show that $\pi \in \widetilde{\mathcal{L}}$. Using addition, multiplication, and composition the function $x \mapsto \frac{1}{1+x^2}$ is in $\widetilde{\mathcal{L}}$. By the following linear differential equation

$$\begin{aligned} h(0) &= 0 \\ \partial_x h(x) &= \frac{1}{1 + x^2} \end{aligned}$$

$\arctan(x) \in \widetilde{\mathcal{L}}$. Then $\pi = 4 \arctan(1)$, hence π can be generated inside $\widetilde{\mathcal{L}}$. This completes the proof. □

Theorem 6. (Characterization of $\mathcal{E}(\mathbb{R})$ [3]) Assume a function $f\colon \mathcal{D} \to \mathbb{R}$ such that $\mathcal{D}$ is the product of compact intervals with rational endpoints and $f \in C^2$. Then $f \in \mathcal{E}(\mathbb{R})$ if and only if $f \in \widetilde{\mathcal{L}}$.

The restriction to C^2 functions in the previous theorem is removed in the work of Campagnolo and Ojakian [10] using their developed techniques of approximation and lifting.

Another extension of $\mathcal{L}$ can be obtained by adding a restricted physically computable version of the μ-operator.

Definition 13. (Computable μ-operator [4]) Assume a function $g\colon \mathcal{I} \times \mathcal{D} \to \mathbb{R}$ where $\mathcal{I} \times \mathcal{D} \subseteq \mathbb{R} \times \mathbb{R}^k$ is a product of closed intervals. Assume g satisfies the following conditions:

1. g is differentiable,

2. $g(x, \bar{y})$ is non-decreasing over $\mathcal{I}$,

3. $g(x, \bar{y})$ has a unique root x^* in the interior of $\mathcal{I}$,

4. $\partial_x g(x, \bar{y})|_{x^*} > 0$.

Define a new function $f\colon \mathcal{D} \to \mathbb{R}$ as follows

$$f(\bar{y}) = !\mu_x[g(x, \bar{y})] = x^* \tag{22}$$

Lemma 2. ([35]) If the function g in Definition 13 is computable (in the sense of recursive analysis), then $f = !\mu[g]$ is computable.

Definition 14. Extend the function algebra $\mathcal{L}$:

$$\hat{\mathcal{L}} \triangleq [0, 1, U, \theta_3; Comp, LI, !\mu] \tag{23}$$

This new function algebra is a proper extension of $\mathcal{L}$ as indicated by the following lemma.

Lemma 3. $\mathcal{L} \subsetneq \hat{\mathcal{L}}$

Proof. The constant -1 can be generated by $!\mu_x[x + 1]$. π can be generated as follows. First, generate the function $g\colon x \mapsto \frac{1}{1+x^2}$ by $g(x) = !\mu_y[y(1+x^2)-1]$. Then, generate $\arctan(x)$ as the integration of $g(x)$. Finally, $\pi = 4\arctan(1)$. □

Furthermore, $\hat{\mathcal{L}}$ is computable. By its very definition the functions generated by $!\mu$ are total. It is obvious that this operator is computable using any root finding algorithm such as the *bisection method*.

Assume a total recursive function $f\colon \mathbb{N}^k \to \mathbb{N}$. It is known from classical recursion theory [18, 32] that there exist two elementary functions $g, h \in \mathcal{E}$ such that: (1) $g\colon \mathbb{N} \to \mathbb{N}$, (2) $h\colon \mathbb{N} \times \mathbb{N}^k \to \mathbb{N}$, (3) for every $\bar{y} \in \mathbb{N}^k$, there exists at least one $x \in \mathbb{N}$ such that $h(x, \bar{y}) = 0$, and (4) f can be defined as $g \circ !\mu_x[h]$. By theorem 5, these functions g and h have real extensions $\tilde{g}, \tilde{h} \in \hat{\mathcal{L}}$. Then the minimalization operator of this latter algebra can be applied, along with composition, over slightly modified variants of $\tilde{g}$ and $\tilde{h}$ to simulate the discrete case and thus obtaining a function $\tilde{f} \in \hat{\mathcal{L}}$ such that $\tilde{f} \restriction \mathbb{N}^k = f$. This shows that any total recursive $\mathbb{N}$-function has an extension in $\hat{\mathcal{L}}$. The converse can similarly by proven observing that $\mathcal{L}$ exactly captures the discrete elementary class. Hence we have the following.

Theorem 7. ([4]) Assume a function $f\colon \mathbb{N}^k \to \mathbb{N}$. Then f is total recursive if and only if $f \in dp(\hat{\mathcal{L}})$.

It is an open question how far $\hat{\mathcal{L}}$ goes beyond $\mathcal{L}$ with respect to computable analysis. Another interesting question would be whether $!\mu$ can be replaced by a more natural intuitive operation derived from the mathematical analysis repository. It seems that generally to reach computable analysis the limit operator is necessary so by combining the above schemas together we obtain the following function algebra that is able to capture a restricted subclass of the whole set of computable analysis functions.

Definition 15. Define the function algebra

$$\mathcal{L}^* \triangleq [0, 1, U, \theta_3; Comp, LI, Lim, !\mu] \tag{24}$$

From the above discussion, in particular Theorem 7 and the observation about the necessity of the limit operator to complete any discrete class to the corresponding computable analysis one, $\mathcal{L}^*$ can characterize computable analysis as follows.

Theorem 8. ([4]) Let f be a C^2 function defined over a compact domain, then $f \in \mathcal{L}^*$ if and only if $f \in \mathcal{R}ec(\mathbb{R})$.

The use of minimalization was just an explicit imitation of the corresponding operator over the natural numbers, however, it does not seem very natural when applied to real computation, especially from the perspective of the mathematical analysis community. M. Campagnolo and K. Ojakian have excluded minimalization by strengthening the integration operator by removing the linearity restriction [9].

Definition 16 (Ordinary Differential Equation operator)**.** Assume a sequence of functions $g_1(\bar{y}), \ldots, g_m(\bar{y})$ and a sequence of functions $h_1(x, \bar{y}, \bar{z}), \ldots, h_m(x, \bar{y}, \bar{z})$. The functions $\bar{g}$ and $\bar{h}$ may be partial. Consider the following initial value problem:

$$
\begin{aligned}
f_1(0,\bar{y}) &= g_1(\bar{y}) \\
&\vdots \\
f_m(0,\bar{y}) &= g_m(\bar{y}) \\
\partial_x f_1(x,\bar{y}) &= h_1(x,\bar{y},f_1,\ldots,f_m) \\
&\vdots \\
\partial_x f_m(x,\bar{y}) &= h_m(x,\bar{y},f_1,\ldots,f_m)
\end{aligned}
\tag{25}
$$

Then given $\bar{y}$, either $\bar{f}(x,\bar{y})$ is not defined anywhere, or is defined over a maximal open interval containing 0. Let ODE denote the operator that takes $\bar{g}$ and $\bar{h}$ and returns $f_1 = ODE(\bar{g},\bar{h})$ which is the first component of the solution to the above initial value problem.

With this new operator define the following stronger variation of $\mathcal{L}$.

Definition 17. Define the function algebra:

$$\mathscr{D} = [0, 1, -1, U, \theta_3; Comp, ODE, Lim] \tag{26}$$

Assume an operator that takes one function argument and generates its inverse. This latter operator was used by Campagnolo and Ojakian as an intermediary between unique minimalization and ordinary differential equations (along with Theorem 8) to give a *differential recursion characterization* of computable analysis over compact domains.

Theorem 9 ([9])**.** Let f be a C^2 function defined over a compact domain, then $f \in \mathscr{D}$ if and only if $f \in \mathcal{R}ec(\mathbb{R})$.

To the best of our knowledge this is so far the best that has been done in algebraically characterizing computable analysis functions. Several research directions need to be pursued.

1. Removing the restriction of C^2 functions. There are simple computable analysis functions such as $|x|$ which are not even C^1.
2. Characterizing real functions over arbitrary domains. This requires first a well-defined notion of a *partial real function*. Take for example the function $f\colon \subseteq \mathbb{R} \to \mathbb{R}$ that maps x to $\frac{1}{x}$. Then f is partial with domain $\mathbb{R}\backslash\{0\}$, which has two connected components: $(-\infty, 0)$ and $(0, \infty)$. Note that f is continuous on both components and hence does not violate the continuity condition necessary for computable analysis and is therefore computable in that context. Operators such as composition, integration, and juxtaposition (forming of vector functions from scalar ones), should then be carefully defined over partial functions.
3. Given the existence of the limit operator, it might be possible to simplify the function algebras by removing θ_3 from the set of basic functions.

5 Conclusion

In this article we have surveyed two approaches to real computation: computable analysis and Moore's $\mathbb{R}$-recursion theory, and how they interrelate through the use of function algebras. Generally, the idea of using function algebras to characterize different subclasses of computable analysis functions has been inspired both by the corresponding characterizations of discrete computability classes and the introduction of Moore's $\mathbb{R}$-recursion theory. Algebraic characterizations have provided useful tools to: (1) get a deeper understanding of the power of real computation, (2) give machine-independent characterizations of real computability that is more natural and intuitive than the machine viewpoint; this is specially appealing to the mathematical analysis and numerical analysis communities, (3) construct a mathematical framework within which a unified theory of continuous computation can be developed, and (4) view discrete computability from a different perspective, for example Mycka and Costa [26] proposed two classes of real computable functions such that their inequality implies $P \neq NP$.

Still much need to be done regarding the algebraic characterization of the sub-elementary, in particular the lower complexity-theoretic, classes of computable analysis. In general we are still missing a deep understanding of the complexity of computable analysis (especially over arbitrary non-compact domains), for example there is not a unified definition of the space complexity of real functions.

Also we are even still lacking a definition of real computation over arbitrary domains. For example, what happens when the domain has several connected components? What about the computational nature of the boundaries of these components? What happens when the number of these components is finite, countable, and uncountable? What about the computational properties of the derivatives and integrations of such functions?

References

[1] V. Blondel and J. Tsitsiklis. A Survey of Computational Complexity Results in Systems and Control. *Automatica*, 36(9):1249–1274, 2000.

[2] O. Bournez, M. Campagnolo, D. Graça, and E. Hainry. Polynomial Differential Equations Compute All Real Computable Functions on Computable Compact Intervals. *Journal of Complexity*, 23(3):317–335, 2007.

[3] O. Bournez and E. Hainry. An Analog Characterization of Elementarily Computable Functions Over the Real Numbers. In *2nd APPSEM II Workshop (APPSEM'04)*, Tallinn, Estonia, 2004.

[4] O. Bournez and E. Hainry. Recursive Analysis Characterized as a Class of Real Recursive Functions. *Fundamenta Informaticae*, 74(4):409–433, 2006.

[5] M. Campagnolo. *Computational Complexity of Real Valued Recursive Functions and Analog Circuits*. PhD thesis, Instituto Superior Técnico, 2001.

[6] M. Campagnolo, C. Moore, and J. Costa. An Analog Characterization of the Subrecursive Functions. In *4th Conference on Real Numbers and Computers*, pages 91–109.

[7] M. Campagnolo, C. Moore, and J. Costa. Iteration, Inequalities, and Differentiability in Analog Computers. *Journal of Complexity*, 16(4):642–660, 2000.

[8] M. Campagnolo, C. Moore, and J. Costa. An Analog Characterization of the Grzegorczyk Hierarchy. *Journal of Complexity*, 18(4):977–1000, 2002.

[9] M. Campagnolo and K. Ojakian. Characterizing Computable Analysis with Differential Equations. *Electronic Notes in Theoretical Computer Science*, 221:23–35, 2008.

[10] M. Campagnolo and K. Ojakian. The Elementary Computable Functions over the Real Numbers: Applying Two New Techniques. *Archives for Mathematical Logic*, 46(7-8):593–627, 2008.

[11] P. Clote. *Handbook of Computability Theory*, chapter Computational Models and Function Algebras, pages 589–681. Elsevier, 1999.

[12] M. C. Daniel Graça and J. Buescu. Robust Simulations of Turing Machines with Analytic Maps and Flows. *Lecture Notes in Computer Science*, 3526:169–179, 2005.

[13] J. Gakwaya. A Survey of the Grzegorczyk Hierarchy and its Extension through the BSS Model of Computability. Technical report, Royal Holloway, University of London, 1997. NeuroCOLT Technical Report Series.

[14] D. S. Graça. Some Recent Developments on Shannon's General Purpose Analog Computer. *Mathematical Logic Quarterly*, 50:473–485, 2004.

[15] D. S. Graça and J. F. Costa. Analog Computers and Recursive Functions over the Reals. *Journal of Complexity*, 19(5):644–664, 2003.

[16] A. Grzegorczyk. Some Classes of Recursive Functions. *Rosprawy Matematyzne*, 4, 1953.

[17] A. Grzegorczyk. Computable functionals. *Fundamenta Mathematicae*, 42:168–202, 1955.

[18] L. Kalmár. Egyzzerü Példa Eldönthetetlen Aritmetikai Problémára. *Mate és Fizikai Lapok*, 50:1–23, 1943.

[19] A. Kawamura. Differential Recursion. *ACM Transactions on Computational Logic*, 10(3):1–20, 2009.

[20] K. 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] L. Lipshitz and L. A. Rubel. A Differentially Algebraic Replacement Theorem, and Analog Computability. *Proceedings of the American Mathematical Society*, 99(2):367–372, 1987.

[23] B. Loff, J. Costa, and J. Mycka. Computability on Reals, Infinite Limits and Differential Equations. *Applied Mathematics and Computation*, 191(2):353–371, 2007.

[24] K. Meer and C. Michaux. A Survey on Real Structural Complexity Theory. *Bulletin of the Belgian Mathematical Society*, 4(1):113–148, 1997.

[25] C. Moore. Recursion Theory on the Reals and Continuous-Time Computation. *Theoretical Computer Science*, 162(1):23–44, 1996.

[26] J. Mycka and J. Costa. The $P \neq NP$ Conjecture in the Context of Real and Complex Analysis. *Journal of Complexity*, 22(2):287–303, 2006.

[27] J. Mycka and J. Costa. A New Conceptual Framework for Analog Computation. *Theoretical Computer Science*, 374:277–290, 2007.

[28] J. Mycka and J. F. Costa. What Lies beyond the Mountains? Computational Systems beyond the Turing Limit. *European Association for Theoretical Computer Science Bulletin*, 85:181–189, 2005.

[29] Olivier Bournez and Manuel L. Campagnolo. A Survey on Continuous Time Computations. In S.B. Cooper and B. Löwe and A. Sorbi, editor, *New Computational Paradigms. Changing Conceptions of What is Computable*, pages 383–423. Springer-Verlag, New York, 2008.

[30] P. Orponen. A Survey of Continous-Time Computation Theory. In *Advances in Algorithms, Languages, and Complexity*, pages 209–224, 1997.

[31] Pour-El. Abstract Computability and its Relation to the General Purpose Analog Computer. *Transactions of the American Mathematical Society*, 199:1–28, 1974.

[32] H. E. Rose. *Subrecursion: Functions and Hierarchies*. Clarendon Press, 1984.

[33] C. E. Shannon. Mathematical Theory of the Differential Analyzer. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.

[34] A. Turing. On Computable Numbers, With an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936. (correction ibid. 43, pp 544-546, 1937).

[35] K. Weihrauch. *Computable Analysis: An Introduction*. Springer, 1 edition, 2000.

[36] Q. Zhou. *Computing and Combinatorics*, volume 1276 of *Lecture Notes in Computer Science*, chapter Subclasses of Computable Real Valued Functions, pages 156–165. Springer Berlin / Heidelberg, 1997.

[37] J. Zimmerman. *Classes of Grzegorczyk's-Computable Real Numbers*. PhD thesis, University of Minnesota, 1990.