Chapter 4

On the Links Between Several Models

In this chapter, we present some of our results of comparisons between several continuous time models. We first focus on the General Purpose Analog Computer from Shannon [Shannon, 1941], and on polynomial Cauchy problems. Later on, we will focus on subclasses of $\mathbb{R}$-recursive functions. $\mathbb{R}$-recursive functions were introduced by [Moore, 1998]. We relate them to computable functions in the sense of recursive analysis.

All the results of this chapter have been obtained in collaborations. The results about the GPAC are the fruit of a collaboration with Manuel Campagnolo, Daniel Graça and Emmanuel Hainry (our PhD student). The results on $\mathbb{R}$-recursive functions also belong to the PhD thesis of Emmanuel Hainry.

4.1 A Church-Turing Thesis for Analog Computations?

According to Church-Turing thesis, all sufficiently powerful "reasonable" models of digital computations are computationally equivalent to Turing machines.

No similar result is known when considering analog computations. Many analog models have been studied, including the BSS model [Blum et al., 1989], Moore's $\mathbb{R}$-recursive functions [Moore, 1998], neural networks [Siegelmann, 1999], or computable analysis [Pour-El and Richards, 1989], [Ko, 1991], [Weihrauch, 2000a], but none is able to affirm itself as "universal". In part, this is due to the fact that few relations between them are known. Moreover some of the known results assert that these models are not equivalent, making the idea of a Church-Turing thesis for analog models an apparently unreachable goal. For example the BSS model allows discontinuous functions while only continuous functions can be computed in the framework of computable analysis [Weihrauch, 2000b].

In this chapter, we will show that some of our results prove that this goal may not be as far as those results suggest.

4.2 GPAC, Polynomial Cauchy Problems and Recursive Analysis: three equivalent paradigms

First, we proved the equivalence of two models of analog computations that were previously considered non-equivalent: computable analysis and *General Purpose Analog Computer* (GPAC) from Claude Shannon.

4.2.1 Introduction

The GPAC was introduced in 1941 by Shannon [Shannon, 1941] as a mathematical model of an analog device: the Differential Analyzer [Bush, 1931]. The Differential Analyzer was used from the 1930s to the early 60s to solve numerical problems. For example, Differential Analysers were used to solve ballistics problems. These devices were first built with mechanical components and later evolved to electronic versions. A GPAC may be seen as a circuit built of interconnected black boxes, whose behavior is given by Figure 4.1, where inputs

are functions of an independent variable called the *time* (in an electronic Differential Analyzer, inputs usually correspond to electronic voltages). These black boxes add or multiply two inputs, generate a constant, or solve a particular kind of Initial Value Problem defined with Ordinary Differential Equations (ODE for short).

While many of the usual real functions are known to be generated by a GPAC, a notable exception is the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ [Shannon, 1941]. If we have in mind that this function is known to be computable under the computable analysis framework [Pour-El and Richards, 1989], the previous result has long be interpreted as evidence that the GPAC is a somewhat weaker model than computable analysis.

However, we believe that this limitation is due to the notion of GPAC-computability rather than the model itself.

The GPAC usually computes in "real time" - a very restrictive form of computation. But if we change this notion of computability to the kind of "converging computation" used in recursive analysis, then it has been shown recently that the Γ function becomes computable [Graça, 2004].

In the rest of this section, we will reinforce strongly this result by proving that actually *any computable*[1] *function* over a compact domain can be computed by a GPAC in this sense. Reciprocally, we show that under some reasonable hypothesis, the converse is also true.

In other words, the computational power of GPAC coincides with that of recursive analysis.

Observe that it has been shown in [Graça et al., 2005] that Turing machines can be simulated by GPACs. We showed in some way that this can be extended to type 2 Turing machines, or to oracle Turing machines that are used in recursive analysis: see [Weihrauch, 2000a].

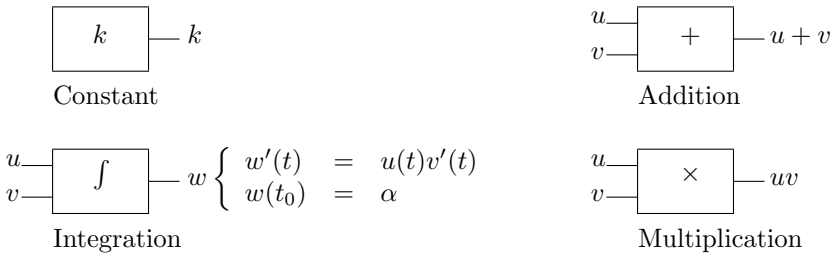


Figure 4.1: The basic units of a GPAC.

4.2.2 The GPAC

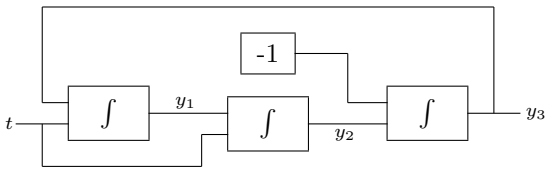
The GPAC was originally introduced by Shannon in [Shannon, 1941], and further refined in [Pour-El, 1974], [Lipshitz and Rubel, 1987], [Graça and Costa, 2003], and [Graça, 2004]. The model basically consists of families of circuits built with the basic units presented in Figure 4.1. Not all kinds of interconnections are allowed since this may lead to undesirable behavior (*e.g.* non-unique outputs). For further details, refer to [Graça and Costa, 2003].

Shannon, in his original paper, already mentions that the GPAC generates polynomials, the exponential function, the usual trigonometric functions, and their inverses. More generally, Shannon claims that all functions generated by a GPAC are differentially algebraic, *i. e.* they satisfy the condition of the following definition:

Definition 1 *The unary function y is differentially algebraic (d.a.) on the interval I if there exists a nonzero polynomial p with real coefficients such that*

$$p\left(t, y, y', ..., y^{(n)}\right) = 0, \qquad on\ I. \tag{4.1}$$

[1] If not otherwise stated, the expression "computable function" is interpreted in the computable analysis sense.



$$\begin{cases} y_1' = y_3 & \& \quad y_1(0) = 1 \\ y_2' = y_1 & \& \quad y_2(0) = 0 \\ y_3' = -y_1 & \& \quad y_3(0) = 0 \end{cases}$$

Figure 4.2: Generating cos and sin via a GPAC: circuit version on the left and ODE version on the right. One has $y_1 = \cos$, $y_2 = \sin$, $y_3 = -\sin$.

As a corollary, and noting that the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ is not d.a. [Rubel, 1989], we get that

Proposition 1 *The Gamma function cannot be generated by a GPAC.*

However, Shannon's proof relating functions generated by GPACs with d.a. functions was incomplete (as pointed out and partially corrected in [Pour-El, 1974], [Lipshitz and Rubel, 1987]). Actually, as pointed out in [Graça and Costa, 2003], the original GPAC model suffers from several problems of ill-definitions.

4.2.3 Polynomial Cauchy Problems

However, for the more robust class of GPACs defined in [Graça and Costa, 2003], the following stronger property holds:

Proposition 2 *A scalar function $f : \mathbb{R} \to \mathbb{R}$ is generated by a GPAC iff it is a component of the solution of a system*

$$y' = p(y, t), \tag{4.2}$$

where p is a vector of polynomials. That is to say, iff it is a projection of a solution of a polynomial Cauchy problem, according to the the terminology of Chapter 1. A function $f : \mathbb{R} \to \mathbb{R}^k$ is generated by a GPAC iff all of its components are.

From now on, we will mostly talk about GPACs as being systems of ODEs of the type (4.2), that is to say as polynomial Cauchy problems according to the terminology of Chapter 1.

For a concrete example of the previous proposition, see Figure 4.2. GPAC generable functions (in the sense of [Graça and Costa, 2003]) are obviously d.a.. Another interesting consequence is the following (recall that solutions of analytic ODEs are always analytic – *cf.* [Arnold, 1992]):

Corollary 1 *If f is a function generated by a GPAC, then it is analytic.*

As we have seen in Proposition 1, the Gamma function is not generated by a GPAC. However, as we said, it has been recently proved that it can be computed by a GPAC if we use a notion of GPAC computability inspired by the notion used in recursive analysis [Graça, 2004].

Notice that in Shannon's original definition of the GPAC, nothing is assumed about the constants and initial conditions of the ODE (4.2). In particular, there can be non-computable reals. This kind of GPAC can trivially lead to super-Turing computations. To avoid this, the model of [Graça, 2004] can actually be reinforced as follows:

Definition 2 *A function $f : [a, b] \to \mathbb{R}$ is GPAC-computable* [2] *iff there exists some computable polynomial $p : \mathbb{R}^{n+1} \to \mathbb{R}^n$ and $n - 1$ computable values $\alpha_1, ..., \alpha_{n-1}$ such that:*

1. $(y_1, ..., y_n)$ is the solution of ODE $y' = p(y, t)$ with initial condition $(\alpha_1, ..., \alpha_{n-1}, x)$ set at time $t_0 = 0$

[2] Note that in this paper, the term GPAC-computability refers to this particular notion. The expression "generated by a GPAC" corresponds to Shannon's notion of computability.

2. There are $i, j \in \{1, ..., n\}$ such that $\lim_{t\to\infty} y_j(t) = 0$ and $|f(x) - y_i(t)| \leq y_j(t)$.[3]

We remark that $\alpha_1, \ldots, \alpha_{n-1}$ are auxiliary parameters needed to compute f.

Daniel Graça's result can then be stated as.

Proposition 3 ([Graça, 2004]) *The Γ function is GPAC-computable.*

4.2.4 Computable Analysis

Recursive analysis, or *computable analysis*, was introduced by Turing [Turing, 1936], Grzegorczyk [Grzegorczyk, 1957], and Lacombe [Lacombe, 1955].

The idea underlying computable analysis is to extend the classical computability theory so that it might deal with real quantities. See [Weihrauch, 2000a] for an up-to-date monograph of computable analysis from the computability point of view, or [Ko, 1991] for a presentation from a complexity point of view.

Let $\nu_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ be the following representation[4] of rational numbers by integers:

$$\nu_{\mathbb{Q}}(\langle p, r, q\rangle) \mapsto \frac{p-r}{q+1},$$

where $\langle ., ., .\rangle : \mathbb{N}^3 \to \mathbb{N}$ is a computable bijection.

A sequence of integers $(x_i)_{i\in\mathbb{N}}$ *represents real* x if $(\nu_{\mathbb{Q}}(x_i))$ converges quickly toward x, denoted by $(x_i) \rightsquigarrow x$, in the following sense:

$$\forall i, |\nu_{\mathbb{Q}}(x_i) - x| < \exp(-i).$$

For a sequence of k-tuples $(\mathbf{x}_i)_{i\in\mathbb{N}}$, we write $(\mathbf{x}_i) \rightsquigarrow \mathbf{x}$ when it holds componentwise.

Definition 3 (Recursive Analysis [Weihrauch, 2000a]) *We say that a function $f : \mathbb{R}^k \to \mathbb{R}$ is* computable *if there exists a recursive functional $\Phi : (\mathbb{N}^k)^{\mathbb{N}} \times \mathbb{N} \to \mathbb{N}$ such that for all $\mathbf{x} \in \mathbb{R}^k$, for all sequence $X = (\mathbf{x}_n) \in (\mathbb{N}^k)^{\mathbb{N}}$, we have $(\Phi(X, j))_j \rightsquigarrow f(\mathbf{x})$ whenever $X \rightsquigarrow \mathbf{x}$.*

A function $f : \mathbb{R}^k \to \mathbb{R}^l$, with $l > 1$, will be said computable if its projections are.

4.2.5 Our Result

Our result, proved in [Bournez et al., 2006], is the following: if a real function f defined over a compact interval is computable, then it is GPAC-computable. Conversely, we show that if f is defined over a compact interval is GPAC computable, then it is computable.

In other words:

Theorem 1 (GPAC=Recursive Analysis) *A function $f : [a, b] \to \mathbb{R}$ is computable if and only if it is GPAC computable.*

In other words, GPACs, polynomial Cauchy problems, and recursive analysis are three equivalent paradigms of computation.

Corollary 2 (polynomial Cauchy problems vs Recursive Analysis) *A function $f : [a, b] \to \mathbb{R}$ is computable if and only if it corresponds to a limit of a function definable by a polynomial Cauchy problem in the sense of Chapter 1.*

This may be useful to link these results with our remarks in chapters 1 and 2, on the modelling power of polynomial Cauchy problems, and with the links with a possible Church-Turing thesis for analog models.

From the point of view of computations, our results suggest that polynomial Cauchy problems, and GPACs are real counterparts of Turing machines.

We now go to another series of results, which relate $\mathbb{R}$-recursive functions and recursive analysis.

[3] We assume that $y(t)$ is defined for all $t \geq 0$.

[4] Other natural representations of rational numbers can be used, and lead to the same class of computable functions: see [Weihrauch, 2000a].

4.3 $\mathbb{R}$-Recursive Functions and Recursive Analysis

4.3.1 $\mathbb{R}$-Recursive Functions

In [Moore, 1996], Moore introduced a class of functions over the reals inspired from the classical characterization of computable functions over integers: observing that the continuous analogue of a primitive recursion is a differential equation, Moore proposes to consider the class of $\mathbb{R}$-*recursive functions*, defined as the smallest class of functions containing some basic functions, and closed by composition, differential equation solving (called *integration*), and minimization.

This class of functions, also investigated in articles such as [Mycka, 2003b], [Mycka and Costa, 2006a], [Mycka and Costa, 2006b],[Mycka and Costa, 2005], can be related to functions that can be generated by GPACs: see [Moore, 1996], corrected by [Graça and Costa, 2003].

Putting aside possible objections about the physical feasibility of the μ-operator considered in paper [Moore, 1996], the original definitions of this class in [Moore, 1996] suffer from several technical problems[5]. At least some of them make it possible to use a *"compression trick"* (another incarnation of Zeno's paradox) to simulate in a bounded time an unbounded number of discrete transitions in order to recognize arithmetical reals [Moore, 1996].

In the series of papers [Campagnolo et al., 2000], [Campagnolo et al., 2002], [Campagnolo, 2001], Campagnolo, Costa and Moore propose to consider the (well-defined) class $\mathcal{L}$ of functions over the reals corresponding to the smallest class of functions containing some basic functions and closed by composition and *linear* integration. Class $\mathcal{L}$ is related to functions elementarily computable over integers in classical recursion theory and functions elementarily computable over the real numbers in recursive analysis (discussed in [Zhou, 1997]): any function of class $\mathcal{L}$ is elementarily computable in the sense of recursive analysis, and conversely, any function over the integers elementarily computable in the sense of classical recursion theory is the restriction to integers of a function that belongs to $\mathcal{L}$ [Campagnolo et al., 2002], [Campagnolo, 2001].

These results establish some links between $\mathbb{R}$-recursive functions, GPAC and classical computability over the integers.

4.3.2 Presentation of Our Results

However, the previous results do not provide a characterization of *all* functions over the reals that are elementarily computable in the sense of recursive analysis.

We proved that this is possible to get one.

Theorem 2 *For functions over the reals of class $\mathcal{C}^2$ defined on a product of compact intervals with rational endpoints, f is elementarily computable in the sense of recursive analysis iff it belongs to the smallest class of functions containing some basic functions and closed by composition, linear integration and a simple limit schema.*

We extended this theorem to a characterization of all higher levels of the Grzegorczyk hierarchy (observe that previous theorem is a consequence of this theorem).

Theorem 3 *For functions over the reals of class $\mathcal{C}^2$ defined on a product of compact intervals with rational endpoints, f is computable in the sense of recursive analysis in level $n \geq 3$ of the Grzegorczyk hierarchy iff f belongs to the smallest class of functions containing some (other) basic functions and closed by composition, linear integration and a simple limit schema.*

Furthermore, we extended this to computable functions in the sense of recursive analysis, and not only to *elementarily computable* functions. Indeed, we proved that this is possible to define a rather natural minimization schema, so that

[5]For example not well defined functions are considered, $\infty \times 0$ is always considered as 0, etc.... Some of them are discussed in [Campagnolo et al., 2000], [Campagnolo et al., 2002], [Campagnolo, 2001] and even in the original paper [Moore, 1996].

Theorem 4 *For functions over the reals of class $\mathcal{C}^2$ defined on a product of compact intervals with rational endpoints, f is computable in the sense of recursive analysis iff f belongs to the smallest class of functions containing some basic functions and closed by composition, linear integration, minimization and a simple limit schema.*

We now present formally these results.

4.3.3 Mathematical Preliminaries

The following well-known mathematical results can help to motivate some of our schemas: see for example [Ramis et al., 1995] for a proof of the first. The second, very simple to establish, is explicitly proved in [Bournez and Hainry, 2005].

Lemma 1 (Implicit Function Theorem) *Let $f : \mathcal{D} \times \mathcal{I} \subset \mathbb{R}^{k+1} \to \mathbb{R}$, where $\mathcal{D} \times \mathcal{I}$ is a product of closed intervals, be a function of class[6] $\mathcal{C}^k$, for $k \geq 1$. Assume that for all $\mathbf{x} \in \mathcal{D}$, the equation $f(\mathbf{x}, y) = 0$ has exactly one solution y_0 and that this y_0 belongs to the interior of $\mathcal{I}$. Assume for all $\mathbf{x}$ that*

$$\frac{\partial f}{\partial y}(\mathbf{x}, y_0) \neq 0$$

in the corresponding root y_0. Then function $g : \mathbb{R}^k \to \mathbb{R}$ that maps $\mathbf{x}$ to the corresponding root y_0 is defined over $\mathcal{D}$ and also of class $\mathcal{C}^k$.

Lemma 2 (Simple Convergence Result) *Let $F : \mathbb{R} \times \mathcal{V} \subset \mathbb{R}^{k+1} \to \mathbb{R}^l$ be a function of class $\mathcal{C}^1$, and $\beta(\mathbf{x}) : \mathcal{V} \to \mathbb{R}$, $K(\mathbf{x}) : \mathcal{V} \to \mathbb{R}$ be some continuous functions. Assume that for all t and $\mathbf{x}$,*

$$\|\frac{\partial F}{\partial t}(t, \mathbf{x})\| \leq K(\mathbf{x}) \exp(-t\beta(\mathbf{x})).$$

Let $\mathcal{D}$ be the subset of the $\mathbf{x} \in \mathcal{V}$ with $\beta(\mathbf{x}) > 0$.
Then,

- *for all $\mathbf{x} \in \mathcal{D}$, $F(t, \mathbf{x})$ has a limit $L(\mathbf{x})$ in $t = +\infty$.*
- *Function $L(\mathbf{x})$ is a continuous function.*
- *Furthermore*

$$\|F(t, \mathbf{x}) - L(\mathbf{x})\| \leq \frac{K(\mathbf{x}) \exp(-t\beta(\mathbf{x}))}{\beta(\mathbf{x})}.$$

4.3.4 Classical Recursion Theory

Classical recursion theory deals with functions over integers. Most classes of classical recursion theory can be characterized as closures of a set of basic functions by a finite number of basic rules to build new functions [Clote, 1998], [Rose, 1984], [Odifreddi, 1992]: given a set $\mathcal{F}$ of functions and a set $\mathcal{O}$ of operators on functions (an operator is an operation that maps one or more functions to a new function), $[\mathcal{F}; \mathcal{O}]$ will denote the closure of $\mathcal{F}$ by $\mathcal{O}$.

Proposition 4 (Classical Settings: see e.g. [Rose, 1984], [Odifreddi, 1992]) *Let f be a function from $\mathbb{N}^k$ to $\mathbb{N}$ for $k \in \mathbb{N}$.*
Function f is

[6] Recall that function $f : \mathcal{D} \subset \mathbb{R}^k \to \mathbb{R}^l$, $k, l \in \mathbb{N}$, is said to be of class $\mathcal{C}^r$ if it is r-times continuously differentiable on $\mathcal{D}$. It is said to be of class $\mathcal{C}^\infty$ if it is of class $\mathcal{C}^r$ for all r.

- elementar *iff it belongs to*

$$\mathcal{E} = [0, S, U, +, \ominus; \text{COMP}, \text{BSUM}, \text{BPROD}];$$

- *in class $\mathcal{E}_n$ of the Grzegorczyk hierarchy ($n \geq 3$) iff it belongs to*

$$\mathcal{E}_n = [0, S, U, +, \ominus, E_{n-1}; \text{COMP}, \text{BSUM}, \text{BPROD}];$$

- primitive recursive *iff it belongs to*

$$\mathcal{PR} = [0, S, U; \text{COMP}, \text{REC}];$$

- recursive[7] *iff it belongs to*

$$\mathcal{R}ec = [0, S, U; \text{COMP}, \text{REC}, \text{MU}].$$

A function $f : \mathbb{N}^k \to \mathbb{N}^l$ is elementar (resp: primitive recursive, recursive) iff its projections are elementar (resp: primitive recursive, recursive).

The basic functions $0, (U_i^m)_{i,m\in\mathbb{N}}, S, +, \ominus$ *and the operators* BSUM, BPROD, COMP, REC, MU *are given by*

1. 0 *is constant function* 0*;*
2. $U_i^m : \mathbb{N}^m \to \mathbb{N}$, $U_i^m : (n_1, \ldots, n_m) \mapsto n_i$*;*
3. $S : \mathbb{N} \to \mathbb{N}, S : n \mapsto n + 1$*;*
4. $+ : \mathbb{N}^2 \to \mathbb{N}$, $+ : (n_1, n_2) \mapsto n_1 + n_2$*;*
5. $\ominus : \mathbb{N}^2 \to \mathbb{N}$, $\ominus : (n_1, n_2) \mapsto max(0, n_1 - n_2)$*;*
6. BSUM *: bounded sum. Given f, $h = \text{BSUM}(f)$ is defined by $h : (\mathbf{x}, y) \mapsto \sum_{z<y} f(\mathbf{x}, z)$;*
7. BPROD *: bounded product. Given a function f, the bounded product $h = \text{BPROD}(f)$ is defined by $h : (\mathbf{x}, y) \mapsto \prod_{z<y} f(\mathbf{x}, z)$;*
8. COMP *: composition. Given $f_1, \ldots, f_p$ and g, $h = \text{COMP}(f_1, \ldots, f_p, g)$ is defined as the function verifying $h(\mathbf{x}) = g(f_1(\mathbf{x}), \ldots, f_p(\mathbf{x}))$;*
9. REC *: primitive recursion . Given f and g, $h = \text{REC}(f, g)$ is defined as the function verifying $h(\mathbf{x}, 0) = f(\mathbf{x})$ and $h(\mathbf{x}, n + 1) = g(\mathbf{x}, n, h(\mathbf{x}, n))$;*
10. MU *: minimalization. Given a function f, function μf is defined on all $\mathbf{x}$ for which there is a y such that $\forall z \leq y$, $f(\mathbf{x}, z)$ is defined and $f(\mathbf{x}, y) = 0$. For such $\mathbf{x}$, the minimalization of f is $\mu f : \mathbf{x} \mapsto \inf\{y; f(\mathbf{x}, y) = 0\}$.*
11. *Functions E_n, involved in the definition of the classes $\mathcal{E}_n$ of the Grzegorczyk Hierarchy, are defined by induction as follows (when f is a function, $f^{[d]}$ denotes its d-th iterate: $f^{[0]}(\mathbf{x}) = x$, $f^{[d+1]}(\mathbf{x}) = f(f^{[d]}(\mathbf{x}))$):*
    (a) *$E_0(x, y) = x + y$, $E_1(x) = (x + 1) \times (y + 1)$, $E_2(x) = 2^x$;*
    (b) *$E_{n+1}(x) = E_n^{[x]}(1)$ for $n \geq 2$.*

[7] This class is often called *partial recursive* since it contains partial functions as opposed to the class of total recursive functions.

We have (see [Rose, 1984], [Odifreddi, 1992])

$$\mathcal{E} \subseteq \mathcal{PR} \subseteq \mathcal{R}ec,$$

and the inclusions are strict. One has also

$$\mathcal{E}_3 = \mathcal{E}$$

and

$$\mathcal{PR} = \cup_i \mathcal{E}_i.$$

If TIME(t) and SPACE(t) denote the classes of functions that are computable with time and space t, then, for all $n \geq 3$,

$$\mathcal{E}_n = \text{TIME}(\mathcal{E}_n) = \text{SPACE}(\mathcal{E}_n),$$

and

$$\mathcal{E} = \text{TIME}(\mathcal{E}),$$

$$\mathcal{E} = \text{SPACE}(\mathcal{E})$$

and

$$\mathcal{PR} = \text{TIME}(\mathcal{PR}) = \text{SPACE}(\mathcal{PR}).$$

Class $\mathcal{PR}$ corresponds to functions computable using *For-Next programs*. Class $\mathcal{E}$ corresponds to computable functions bounded by some iterate of the exponential function. At most two nested For-Next loops are required for a function of class $\mathcal{E}$, whereas general functions from class $\mathcal{PR}$ may require an arbitrary high number of such nested loops [Rose, 1984], [Odifreddi, 1992].

Let's close this presentation by observing that the minimalization operator can be reinforced into a unique minimalization operator as follows.

Proposition 5 *A function f from $\mathbb{N}^k$ to $\mathbb{N}^l$, for $k, l \in \mathbb{N}$, is recursive iff its projections belong to*

$$[0, U, S; \text{COMP}, \text{REC}, \text{UMU}]$$

where operator UMU *is defined as follows:*

1. UMU*: unique minimalization. Given f, that satisfies that for all $\mathbf{x}$ that there is at most one y with $f(\mathbf{x}, y)$ defined and equals to 0, the unique minimalization of f, denoted by $!\mu(f)(\mathbf{x})$, is defined on all $\mathbf{x}$ for which there is a (unique) y with $f(\mathbf{x}, y) = 0$. For such $\mathbf{x}$, $!\mu(f)(\mathbf{x})$ is defined as that unique y.*

In classical computability, more general objects than functions over the integers can be considered, in particular functionals, i.e. functions $\Phi : (\mathbb{N}^m)^{\mathbb{N}} \times \mathbb{N}^k \to \mathbb{N}^l$. A functional will be said to be *elementarily* (or *primitive recursively*, *recursively*) computable when it belongs to the corresponding[8] class.

In recursive analysis, a function f over the real numbers will be said *elementarily* (respectively $\mathcal{E}_n$) computable if the corresponding functional Φ in definition 3 is. The class of computable functions (respectively elementarily computable, $\mathcal{E}_n$ computable) over the reals will be denoted by $\mathcal{R}ec(\mathbb{R})$ (resp. $\mathcal{E}(\mathbb{R})$, $\mathcal{E}_n(\mathbb{R})$).

[8]Formally, a function f over the integers can be considered as functional $\overline{f} : (V, \vec{n}) \mapsto f(\vec{n})$. Similarly, an operator Op on functions $f_1, \ldots, f_m$ over the integers can be extended to an operator over functionals by fixing first argument $\overline{Op}(F_1, \ldots, F_m) : (V, \vec{n}) \mapsto Op(f_1(V, .), \ldots, f_m(V, .))(\vec{n})$.

In that spirit, given some set $\mathcal{F}$ of basic functions $\mathbb{N}^k \to \mathbb{N}^l$ and a set $\mathcal{O}$ of operators on functions over the integers, we will still (abusively) denote by $[f_1, \ldots, f_p; O_1, \ldots, O_q]$ for the smallest class of functionals that contains basic functions $\overline{f_1}, \ldots, \overline{f_p}$, plus the functional $Map : (V, n) \to V_n$, the nth element of sequence V, and which is closed by the operators $\overline{O_1}, \ldots, \overline{O_q}$. For example, a functional will be said elementarily computable iff it belongs to $\mathcal{E} = [Map, \overline{0}, \overline{S}, \overline{U}, \overline{+}, \overline{\ominus}; \overline{\text{COMP}}, \overline{\text{BSUM}}, \overline{\text{BPROD}}]$.

4.3.5 Results from Campagnolo, Costa, Moore

Following the original ideas from [Moore, 1996], but observing that the minimization schema considered in [Moore, 1996] is the source of many technical problems, Campagnolo, Costa and Moore proposed in [Campagnolo et al., 2000], [Campagnolo et al., 2002], [Campagnolo, 2001] not to consider classes of functions over the reals defined in analogy with the full class of recursive functions, but with subclasses. Indeed, they consider classes built in analogy with the class of elementar functions over the integers and the classes of the Grzegorczyk hierarchy over the integers. Furthermore, they proposed to restrict the integration schema to a simpler (and better defined) linear integration schema.

We call *real extension of a function* $f : \mathbb{N}^k \to \mathbb{N}^l$ a function $\tilde{f}$ from $\mathbb{R}^k$ to $\mathbb{R}^l$ whose restriction to $\mathbb{N}^k$ is f.

Definition 4 ([Campagnolo, 2001, Campagnolo et al., 2002]) *Let $\mathcal{L}$ and $\mathcal{L}_n$ be the classes of functions $f : \mathbb{R}^k \to \mathbb{R}^l$, for some $k, l \in \mathbb{N}$, defined by*

$$\mathcal{L} = [0, 1, -1, \pi, U, \theta_3; \mathrm{COMP}, \mathrm{LI}]$$

and

$$\mathcal{L}_n = [0, 1, -1, \pi, U, \theta_3, \overline{E}_{n-1}; \mathrm{COMP}, \mathrm{LI}]$$

where the basis functions 0, 1, -1, π, $(U_i^m)_{i,m\in\mathbb{N}}$, θ_3, $\overline{E}_n$ *and the schemas* COMP *and* LI *are defined as follows:*

1. $0, 1, -1, \pi$ *are the corresponding constant functions;* $U_i^m : \mathbb{R}^m \to \mathbb{R}$ *are, as in the classical settings, projections:* $U_i^m : (x_1, \ldots, x_m) \mapsto x_i$*;*
2. $\theta_3 : \mathbb{R} \to \mathbb{R}$ *is defined as* $\theta_3 : x \mapsto x^3$ *if* $x \geq 0, 0$ *otherwise.*
3. $\overline{E}_n$*: for* $n \geq 3$*, let* $\overline{E}_n$ *denote a monotone real extension of the function* $\exp_n$ *over the integers defined inductively by* $\exp_2(x) = 2^x$*,* $\exp_{i+1}(x) = \exp_i^{[x]}(1)$*.*
4. COMP*: composition is defined as in the classical settings: Given f and g, $h = \mathrm{COMP}(f, g)$ is the function verifying $h(\mathbf{x}) = g(f(\mathbf{x}))$;*
5. LI*: linear integration. From g and h, $\mathrm{LI}(g, h)$ is the maximal solution of the linear differential equation $\frac{\partial f}{\partial y}(\mathbf{x}, y) = h(\mathbf{x}, y) f(\mathbf{x}, y)$ with $f(\mathbf{x}, 0) = g(\mathbf{x})$.*

   In this schema, if g goes to $\mathbb{R}^n$, $f = \mathrm{LI}(g, h)$ also goes to $\mathbb{R}^n$ and $h(\mathbf{x}, y)$ is a $n \times n$ matrix with elements in $\mathcal{L}$.

These classes contain usual functions like $id : x \mapsto x$, sin, cos, exp, $+, \times$, $x \mapsto r$ for all rational r, but only total functions, and $\mathcal{C}^2$ functions[Campagnolo, 2001], [Campagnolo et al., 2002].

A major contribution of [Campagnolo et al., 2002], [Campagnolo, 2001] is to relate these classes of functions over the reals to the previous classes over the integers. In order to compare functions over the reals with functions over the integers, we introduce the following notation: given some class $\mathcal{C}$ of functions from $\mathbb{R}^k$ to $\mathbb{R}^l$, we write $\mathrm{DP}(\mathcal{C})$ (DP stands for discrete part) for the class of functions from $\mathbb{N}^k$ to $\mathbb{N}^l$ which have a real extension in $\mathcal{C}$.

Proposition 6 ([Campagnolo et al., 2002, Campagnolo, 2001])

- $\mathrm{DP}(\mathcal{L}) = \mathcal{E}$*;*
- $\mathrm{DP}(\mathcal{L}_n) = \mathcal{E}_n$.

Actually, stronger inclusions were proved in [Campagnolo et al., 2002, Campagnolo, 2001]:

Proposition 7 ([Campagnolo et al., 2002, Campagnolo, 2001])

- $\mathcal{L} \subset \mathcal{E}(\mathbb{R})$.
- $\mathcal{L}_n \subset \mathcal{E}_n(\mathbb{R})$.

However there is no hope to get the other inclusion, since $\mathcal{E}(\mathbb{R})$ and $\mathcal{E}_n(\mathbb{R})$ contain partial functions, whereas classes $\mathcal{L}$ and $\mathcal{L}_n$ are classes of total functions.

4.3.6 Characterization of Elementary Computable Functions

We proposed to consider new classes of functions that turn out to precisely correspond to classes $\mathcal{E}(\mathbb{R})$ and $\mathcal{E}_n(\mathbb{R})$.

To do so, we restrict to functions defined over a compact domain. One motivation is that elementarily computable functions over an arbitrary domain are not stable by composition.

For technical reasons, we need (actually for the next section) to replace the previous LI schema by the following CLI schema.

Definition 5 (Schema CLI **)** *From functions g, h, and c, with*

- *the norm of each of the partial derivatives of h bounded by c, except possibly its partial derivative in t,*

$\mathrm{CLI}(g,h,c)$ *is a solution*[9] *of the linear differential equation*

$$\frac{\partial f}{\partial y}(\mathbf{x},y) = h(\mathbf{x},y)f(\mathbf{x},y)$$

with $f(\mathbf{x},0) = g(\mathbf{x})$.

In this schema, if g goes to $\mathbb{R}^n$, $f = \mathrm{CLI}(g,h,c)$ goes to $\mathbb{R}^{n+1}$ and $h(\mathbf{x},y)$ is a $(n+1)\times(n+1)$ matrix with elements in $\mathcal{L}$.

One can show that replacing LI schema by CLI schema does not change the previous discussion. Indeed, we have.

Proposition 8

$$\begin{aligned}\mathcal{L} &= [0,1,-1,\pi,U,\theta_3;\mathrm{COMP},\mathrm{LI}]\\ &= [0,1,-1,\pi,U,\theta_3;\mathrm{COMP},\mathrm{CLI}]\end{aligned}$$

and

$$\begin{aligned}\mathcal{L}_n &= [0,1,-1,\pi,U,\theta_3,\overline{E}_{n-1};\mathrm{COMP},\mathrm{LI}]\\ &= [0,1,-1,\pi,U,\theta_3,\overline{E}_{n-1};\mathrm{COMP},\mathrm{CLI}]\end{aligned}$$

Furthermore, we introduce a limit schema. The idea of adding a limit operator has already been investigated in papers like [Mycka and Costa, 2004], [Mycka, 2003b]. However, unlike Costa and Mycka, we are interested in $\mathbb{R}$-subrecursive functions, and not to build a whole hierarchy above recursive functions. As a consequence, our limit schema is more restricted than the one of these articles.

The conditions that we impose on our LIM_w schema are inspired from Lemma 2.

Definition 6 (Schéma LIM**)** *Let $f : \mathbb{R}\times\mathcal{D}\subset\mathbb{R}^{k+1}\to\mathbb{R}^l$ and $K:\mathcal{D}\to\mathbb{R}$ be two functions, and $\beta:\mathcal{D}\to\mathbb{R}$ be a polynomial function with the following hypothesis: for all $\mathbf{x}$, $t\geq\|\mathbf{x}\|$,*

$$\|\frac{\partial f}{\partial t}(t,\mathbf{x})\| \leq K(\mathbf{x})\exp(-t\beta(\mathbf{x})).$$

Then, on any closed product of intervals $I\subset\mathbb{R}^k$ on which $\beta(\mathbf{x})>0$,

$$F(\mathbf{x}) = \lim_{t\to+\infty} f(t,\mathbf{x})$$

exits by Lemma 2.

If F is of class $\mathcal{C}^2$, then we define $\mathrm{LIM}_w(f,K,\beta)$ as this function $F:I\to\mathbb{R}$.

We can now define our classes.

[9] Actually, any restriction to a product of closed intervals of the maximal solution

Definition 7 (Classes $\mathcal{L}^*$, $\mathcal{L}^*_n$) *Classes $\mathcal{L}^*$, and $\mathcal{L}^*_n$, for $n \geq 3$, of functions from $\mathbb{R}^k$ to $\mathbb{R}^l$, for $k, l \in \mathbb{N}$, are defined as the following classes.*

$$\mathcal{L}^* = [0, 1, -1, U, \theta_3; \mathrm{COMP}, \mathrm{CLI}, \mathrm{LIM}],$$

and

$$\mathcal{L}^*_n = [0, 1, -1, U, \theta_3, \overline{E}_{n-1}; \mathrm{COMP}, \mathrm{CLI}, \mathrm{LIM}].$$

These classes can contain some partial functions, and extend the previous classes. Indeed, we have.

Proposition 9

$$\mathcal{L} \subsetneq \mathcal{L}^*,$$

and

$$\mathcal{L}_n \subsetneq \mathcal{L}^*_n$$

for all $n \geq 3$.

Furthermore, we proved in publications [Bournez and Hainry, 2005] and [Bournez and Hainry, 2004a] that these classes characterize precisely elementary computable functions in the sense of recursive analysis.

Indeed, previous theorems 2 and 3 are formally.

Theorem 2 (Characterization of $\mathcal{E}(\mathbb{R})$) *Let $f : \mathcal{D} \subset \mathbb{R}^k \to \mathbb{R}^l$ be a function over the reals of class $\mathcal{C}^2$, with $\mathcal{D}$ a product of compact intervals with rational endpoints.*

f is in $\mathcal{E}(\mathbb{R})$ if and only if $f \in \mathcal{L}^*$.

Actually, this can be generalized to all levels of the Grzegorczyk hierarchy (observe that theorem 2 is the particular case $n = 3$ of theorem 3).

Theorem 3 (Characterization of $\mathcal{E}_n(\mathbb{R})$) *Let $f : \mathcal{D} \subset \mathbb{R}^k \to \mathbb{R}^l$ be a function over the reals of class $\mathcal{C}^2$, with $\mathcal{D}$ a product of compact intervals with rational endpoints. Let $n \geq 3$.*

f is in $\mathcal{E}_n(\mathbb{R})$ if and only if $f \in \mathcal{L}^*_n$.

A few extensions of these results can be found in our articles, in particular a normal form theorem for functions of our classes.

4.3.7 Characterization of Computable Functions

We later on proved that this is possible to characterize computable functions, and not only elementarily computable functions.

To do so, we need to introduce a minimization operator, that makes it possible to simulate discrete minimization over the integers.

However, this operator needs to be stricter than a simple "return the smallest root" since this idea, investigated in [Moore, 1996], has shown to be the source of numerous problems, including ill-defined problems and super-Turing Zeno phenomena. These problems are discussed, and pointed in [Campagnolo, 2001], [Campagnolo et al., 2002], [Mycka, 2003b], [Mycka, 2003a], [Moore, 1996]. The two papers [Mycka, 2003b], [Mycka and Costa, 2004] do provide well-defined alternatives, replacing minimalizations by limit-takings. We propose here to keep to a minimalization schema, not as general as the one from [Moore, 1996].

Our idea is to use the alternative UMU schema, which is equivalent to schema MU for classical computability (see Proposition 5), but has real counterparts which turn out to preserve real computability.

Indeed, motivated by Proposition 5, by Lemma 1 (implicit function theorem), and by the results of recursive analysis about the computability of zeros of a function (see e.g. [Weihrauch, 2000b] where theorems 6.3.5 and 6.3.8 state that the search of a unique zero is computable), we define our unique-zero-finding operator UMU as follows:

Definition 8 (Schema UMU**)** *Given a differentiable function f from $\mathcal{D} \times \mathcal{I} \subset \mathbb{R}^{k+1}$ 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 on $\mathcal{I}$,*
- *and such that y_0 belongs to the interior of $\mathcal{I}$,*
- *with*

$$\frac{\partial f}{\partial y}(\mathbf{x}, y_0) > 0,$$

then UMU(f) *is defined on $\mathcal{D}$ as follows:*

$$\mathrm{UMU}(f) : \left\{ \begin{array}{ccc} \mathcal{D} & \longrightarrow & \mathbb{R} \\ \mathbf{x} & \mapsto & y_0 \text{ such that } f(\mathbf{x}, y_0) = 0. \end{array} \right.$$

We then define.

Definition 9 (Class $\mathcal{L}+!\mu$) *Let $\mathcal{L}+!\mu$ be the class of functions over the reals defined by*

$$\mathcal{L}+!\mu = [0, 1, U, \theta_3; \mathrm{COMP}, \mathrm{CLI}, \mathrm{UMU}].$$

One can show that

$$\mathcal{L} \subset \mathcal{L}+!\mu,$$

and that $\mathcal{L}+!\mu$ contains only functions of class $\mathcal{C}^2$, defined over products of closed intervals.

Our main result in [Bournez and Hainry, 2004b], that constitutes the formalization of previous informal Theorem 4, is given by the two following two theorems.

Theorem 4 *For total functions $\mathcal{R}ec = DP(\mathcal{L}+!\mu)$. I.e:*

- *If a function from $\mathcal{L}+!\mu$ extends some total function over the integers, this latter function is total recursive.*
- *Any total recursive function over the integers, has a real extension that belongs to $\mathcal{L}+!\mu$.*

Definition 10 (Class $\mathcal{L}+!\mu + \mathrm{LIM}_w$) *Let*

$$\mathcal{L}^*_{!\mu} = [0, 1, U, \theta_3; \mathrm{COMP}, \mathrm{CLI}, \mathrm{UMU}, \mathrm{LIM}_w]$$

We have the following theorem, proved in [Bournez and Hainry, 2005].

Theorem 5 *For functions of class $\mathcal{C}^2$ defined over a product of compact intervals with rational endpoints,*

$$\mathcal{L}^*_{!\mu} = \mathcal{R}ec(\mathbb{R}).$$

We proposed a few extensions of this result in this paper.

In other words, one can relate computable functions in the sense of recursive analysis to $\mathbb{R}$-recursive functions.

4.4 Discussions

Concerning *analog models*, all these results provide a characterization of the power of a natural class of analog models over the real numbers and provide new insights for understanding the relations between several analog computational models: GPACs, $\mathbb{R}$-recursive functions, and computable functions in recursive analysis. Moreover, they prove that no hypercomputation phenomenon can happen for this class of functions. In particular, we have a class of functions that is robust in some sense. See the discussion of Chapter 3, about possible notions of robustness.

Concerning *recursive analysis*, our theorems provide *a purely algebraic* and *machine independent* characterization of computable and elementarily computable functions over the reals. Observe the potential benefits offered by these characterizations compared to classical definitions of these classes in recursive analysis, involving discussions about higher-order (type 2) Turing machines (see *e.g.* [Weihrauch, 2000b]), or compared to characterizations in the spirit of [Brattka, 2003], [Kawamura, 2005].

Furthermore, observe that we prove that computable functions over the reals can be defined by *continuous schema*, i.e. by schema on continuous functions. This seems really more natural to use continuous schema to define continuous functions, than always relying on limit schema over discrete functions [Brattka, 2003], [Kawamura, 2005].

4.5 A Complexity Theory?

More generally, in this chapter, we related several models, that were *à priori* distinct.

First we recalled that GPAC generated functions correspond to polynomial Cauchy problems. Then we proved that if an adequate notion of computation for the GPAC is considered, then GPAC and polynomial Cauchy problems have exactly the power of recursive analysis: computable functions are the same.

In next section, we related this computational power to some classes of $\mathbb{R}$-recursive functions. We thus obtained the first algebraic characterization, by analysis, of computable functions in recursive analysis.

The stake to understand a possible Church-Turing thesis for analog models would be to relate these models to other existing models (see for example all the models discussed in Chapter 3).

But we think that a very important question is to understand if one can go from computability to complexity. Can we define a notion of complexity, valid, and universally accepted for analog models?

We presented some existing results in this direction in Chapter 3, and we already discussed several leads. However, the results in this chapter, point out some particular ones.

Concerning first section: Is this possible to relate polynomial-time computable functions to a class of GPAC-computable functions where the error ε would be given by some polynomial function of time variable t? And if such results could hold, can they be extended to other complexity classes?

Concerning second section, the book [Ko, 1991] presents a robust notion of polynomial time computable functions in recursive analysis. Can we characterize complexity classes of recursive analysis algebraically with schemas such as the ones in this chapter?

Some leads to do so are given by algebraic characterizations of complexity classes in classical complexity. For example the characterization of Bellantoni and Cook of polynomial time in [Bellantoni and Cook, 1992], in terms of safe recursive functions. Can these characterizations be adapted to define some classes of $\mathbb{R}$-recursive functions that would correspond to polynomial time? To polynomial space?

Next chapter is precisely related to characterizations in the spirit of the characterization of polynomial time of Bellantoni and Cook, but for another class of computational models. Indeed, we consider the Blum Shub and Smale model, which is a very specific model of computations with a continuous space, but a discrete time.

Bibliography

[Arnold, 1992] Arnold, V. I. (1992). *Ordinary differential equations*. Springer-Verlag, Berlin.

[Bellantoni and Cook, 1992] Bellantoni, S. and Cook, S. (1992). A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2(2):97–110.

[Blum et al., 1989] Blum, L., Shub, M., and Smale, S. (1989). 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.

[Bournez et al., 2006] Bournez, O., Campagnolo, M. L., Graça, D. S., and Hainry, E. (2006). The general purpose analog computer and computable analysis are two equivalent paradigms of analog computation. In Cai, J., Cooper, S. B., and Li, A., editors, *Theory and Applications of Models of Computation, Third International Conference, TAMC 2006, Beijing, China, May 15-20, 2006, Proceedings*, volume 3959 of *Lecture Notes in Computer Science*, pages 631–643. Springer.

[Bournez and Hainry, 2004a] Bournez, O. and Hainry, E. (2004a). An analog characterization of elementarily computable functions over the real numbers. In Diaz, J., Karhumäki, J., Lepisto, A., and Sannella, D. T., editors, *31th International Colloquium on Automata Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 269–280, Turku, Finland. Springer.

[Bournez and Hainry, 2004b] Bournez, O. and Hainry, E. (2004b). Real recursive functions and real extentions of recursive functions. In Margenstern, M., editor, *Machines, Computations and Universality (MCU'2004)*, volume 3354 of *Lecture Notes in Computer Science*, Saint-Petersburg, Russia.

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

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

[Bush, 1931] Bush, V. (1931). The differential analyser. *Journal of the Franklin Institute*, 212(4):447–488.

[Campagnolo et al., 2000] Campagnolo, M., Moore, C., and Costa, J. F. (2000). An analog characterization of the subrecursive functions. In Kornerup, P., editor, *Proc. 4th Conference on Real Numbers and Computers*, pages 91–109. Odense University Press.

[Campagnolo et al., 2002] Campagnolo, M., Moore, C., and Costa, J. F. (2002). An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 18(4):977–1000.

[Campagnolo, 2001] Campagnolo, M. L. (2001). *Computational complexity of real valued recursive functions and analog circuits*. PhD thesis, IST, Universidade Técnica de Lisboa.

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

[Graça et al., 2005] Graça, D., Campagnolo, M., and Buescu, J. (2005). Robust simulations of Turing machines with analytic maps and flows. In Cooper, B., Loewe, B., and Torenvliet, L., editors, *Proceedings of CiE'05, New Computational Paradigms*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179. Springer-Verlag.

[Graça, 2004] Graça, D. S. (2004). Some recent developments on Shannon's general purpose analog computer. *Mathematical Logic Quarterly*, 50(4–5):473–485.

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

[Grzegorczyk, 1957] Grzegorczyk, A. (1957). On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71.

[Kawamura, 2005] Kawamura, A. (2005). Type-2 computability and Moore's recursive functions. In Brattka, V., Staiger, L., and Weihrauch, K., editors, *Proceedings of the 6th Workshop on Computability and Complexity in Analysis*, volume 120 of *Electronic Notes in Theoretical Computer Science*, pages 83–95, Amsterdam. Elsevier. 6th International Workshop, CCA 2004, Wittenberg, Germany, August 16–20, 2004.

[Ko, 1991] Ko, K.-I. (1991). *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston.

[Lacombe, 1955] Lacombe, D. (1955). 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.

[Lipshitz and Rubel, 1987] Lipshitz, L. and Rubel, L. A. (1987). A differentially algebraic replacement theorem, and analog computability. *Proceedings of the American Mathematical Society*, 99(2):367–372.

[Moore, 1996] Moore, C. (1996). Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44.

[Moore, 1998] Moore, C. (1998). Dynamical recognizers: real-time language recognition by analog computers. *Theoretical Computer Science*, 201(1–2):99–136.

[Mycka, 2003a] Mycka, J. (2003a). Infinite limits and r-recursive functions. *Acta Cybernetica*, 16:83–91.

[Mycka, 2003b] Mycka, J. (2003b). μ-recursion and infinite limits. *Theoretical Computer Science*, 302:123–133.

[Mycka and Costa, 2004] Mycka, J. and Costa, J. F. (2004). Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835–857.

[Mycka and Costa, 2005] Mycka, J. and Costa, J. F. (2005). What lies beyond the mountains, computational systems beyond the Turing limit. *European Association for Theoretical Computer Science Bulletin*, 85:181–189.

[Mycka and Costa, 2006a] Mycka, J. and Costa, J. F. (2006a). Analog computation and beyond. Submitted.

[Mycka and Costa, 2006b] Mycka, J. and Costa, J. F. (2006b). The $P \neq NP$ conjecture. Submitted.

[Odifreddi, 1992] Odifreddi, P. (1992). *Classical Recursion Theory*, volume 125 of *Studies in Logic and the foundations of mathematics*. North-Holland.

[Pour-El, 1974] Pour-El, M. B. (1974). Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers). *Transactions of the American Mathematical Society*, 199:1–28.

[Pour-El and Richards, 1989] Pour-El, M. B. and Richards, J. I. (1989). *Computability in Analysis and Physics*. Springer-Verlag.

[Ramis et al., 1995] Ramis, E., Deschamp, C., and Odoux, J. (1995). *Cours de Mathématiques Spéciales, Tome 3, Topologie et éléments d'analyse*. Masson.

[Rose, 1984] Rose, H. (1984). *Subrecursion*. Oxford university press.

[Rubel, 1989] Rubel, L. A. (1989). A survey of transcendentally transcendental functions. *American Mathematical Monthly*, 96(9):777–788.

[Shannon, 1941] Shannon, C. E. (1941). Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354.

[Siegelmann, 1999] Siegelmann, H. T. (1999). *Neural Networks and Analog Computation - Beyond the Turing Limit*. Birkauser.

[Turing, 1936] Turing, A. (1936). On computable numbers, with an application to the Ëntscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265.

[Weihrauch, 2000a] Weihrauch, K. (2000a). *Computable Analysis*. Springer.

[Weihrauch, 2000b] Weihrauch, K. (2000b). *Computable Analysis*. Springer.

[Zhou, 1997] Zhou, Q. (1997). Subclasses of computable real valued functions. *Lecture Notes in Computer Science*, 1276:156–165.