UNIVERSIDADE DE LISBOA
INSTITUTO SUPÉRIOR TECNICO

Analog characterization of complexity classes

Riccardo Gozzi

Supervisor: Doctor João Filipe Quintas dos Santos Rasga

Co-supervisor: Doctor Daniel da Silva Graça

Thesis approved in public session to obtain the PhD Deegre in Information Security

Jury final classification: Pass with Distinction

2022

UNIVERSIDADE DE LISBOA
INSTITUTO SUPÉRIOR TECNICO

Analog characterization of complexity classes

Riccardo Gozzi

Supervisor: Doctor João Filipe Quintas dos Santos Rasga

Co-supervisor: Doctor Daniel da Silva Graça

Thesis approved in public session to obtain the PhD Deegre in Information Security

Jury final classification: Pass with Distinction

Jury

Funding institution:
Fundação para a Ciência e a Tecnologia

2022

Abstract

The purpose of the present dissertation is to characterize complexity classes from standard computational complexity theory by means of systems of ordinary differential equations. We start this work by recalling the concepts of generable functions as functions represented by solutions of polynomial ordinary differential equations. Then we present the details about a previous existing analog characterization for the classes P and FP. Building on this result we show how to extend the characterization for the class $\mathrm{FEXPTIME}$, and then for the classes of elementary functions and primitive recursive functions. To be able to obtain these extensions we have to overcome the fact that the previous construction designed for FP relied on composition properties of polynomials which does not stand for other general boundaries. We also demonstrate that with some modifications to the main definitions of our simulation it is possible to provide a description of complexity classes of decidable sets, such as $\mathrm{EXPTIME}$. Moreover, modifying the way the continuous simulation of Turing machines is performed we are able to provide a complete characterization of the polynomial space complexity classes $\mathrm{FPSPACE}$ and PSPACE.

Finally, we discuss the complexity of the problem of computing the complex square root over simply connected domains of the complex plane, describing an algorithm that proves the upper complexity bound to belong to the class $\mathrm{P}^{\bigoplus \mathrm{P}}$. This description improves the existing computational complexity result for this problem, which was achieved with an algorithm of complexity P^{MP}.

Keywords: Computability and complexity, analog computation, dynamical systems, computable analysis, ordinary differential equations, Grzegorczyk hierarchy, general purpose analog computer, EXPTIME, PSPACE, complex analysis, complex square root.

We acknowledge the support of Instituto de Telecomunicações. This work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020.

Resumo

O objetivo da presente dissertação é caracterizar classes de complexidade da teoria da complexidade computacional através de equações diferenciais ordinárias polinomiais.

Inicialmente relembramos o conceito de função gerável, que são funções representadas por meio de soluções de equações diferenciais ordinárias polinomiais. De seguida, utilizando funções geráveis, apresentamos em detalhe uma caracterização analógica já existente na literatura para as classes P e FP. Com base neste resultado, é mostrado como se pode estender a caracterização anterior para a classe $\mathrm{FEXPTIME}$, e de seguida para a hierarquia de Grzegorczyk. Em consequência da caracterização da hierarquia de Grzegorczyk, obtemos também uma caracterização da classe das funções elementares e da classe das funções recursivas primitivas. De modo a poder obter estas extensões, é necessário ultrapassar alguns problemas. Em particular, a construção anteriormente utilizada para caracterizar FP depende do facto da classe dos polinómios ser fechada sob composição, enquanto isso não é necessariamente verdadeiro para outras classes de funções como a classe das funções exponenciais. Adicionalmente mostramos que, com algumas modificações nas principais definições de simulação de sistemas discretos através de sistemas contínuos, é possível fornecer uma descrição puramente analógica de classes de complexidade envolvendo linguagens, como $\mathrm{EXPTIME}$. Além disso, modificando a forma como a simulação contínua das máquinas de Turing é realizada, é também apresentada uma caracterização analógica das classes de complexidade $\mathrm{FPSPACE}$ e PSPACE envolvendo espaço polinomial.

Finalmente, discutimos a complexidade computacional do problema de calcular a raiz quadrada complexa sobre domínios simplesmente conexos no plano complexo, e descrevemos um algoritmo de complexidade computacional $\mathrm{P}^{\oplus\mathrm{P}}$, melhorando o resultado existente de complexidade computacional para este problema, que utilizava um algoritmo de complexidade P^{MP}.

Palavras-chave: Computabilidade e complexidade, computação analógica, sistemas dinâmicos, análise computável, equações diferenciais ordinárias, hierarquia de Grzegorczyk, GPAC, EXPTIME, PSPACE, análise complexa, raiz quadrada complexa.

Agradecemos o apoio do Instituto de Telecomunicações. Este trabalho é financiado pela FCT/MCTES através de fundos nacionais e, quando aplicável, cofinanciados por fundos da UE no âmbito do projeto UIDB/50008/2020.

Acknowledgments

I would like to mention in this section some people that made this thesis possible and helped me during the course of my PhD. I would like to start by thanking professor Daniel Graça who has always been kind, patient and professional with me and whose many ideas, suggestions and inputs contributed to keep me interested and satisfied about our work. I am also thankful to professor João Rasga, who has always been present to contribute to the growth of our research as well as the correct development of my PhD plan. On this note, I would like to mention here professor Yasser Omar for having given me the opportunity to join the DP-PMI program, and professor Paulo Mateus, who has helped me transition from the physics curriculum to the information security curriculum and has always been helpful to tackle any possible inconvenience during the course of these four years. Moreover, an important acknowledgment goes to professors Paulo Mateus and José Félix Costa, whose insightful courses on computability and complexity theory have greatly inspired the direction taken by my research. I am also particularly grateful to professor Dieter Spreen from University of Siegen for his support to my work. I would like to thank professor Akitoshi Kawamura, who guided me through my first scientific collaboration. Our collaboration has led to interesting, ongoing work that has been presented in this thesis in chapter 7.

Finally, a special mention goes to Instituto Superior Tecnico and to Instituto de Telecomunicações for hosting my programme. My research has been funded by Fundação para a Ciência e a Tecnologia, grant PD/BD/135190/2017 throught the Doctoral Programme in the Physics and Mathematics of Information. Moreover, this thesis work was partially funded by FCT/MCTES through national funds and co-funded EU funds under the project UIDB/50008/2020.

To conclude, the last mention goes to the unbounded support of my family and friends, with whom I shared all the experiences I made in these last four years.

Contents

Chapter 1

Introduction

The main purpose of this thesis work is to investigate the nature of analog computation through the help of dynamical systems. This investigation may provide important tools to future research on the validity of the generalized form of the Church-Turing hypothesis.

Since its first formulation, the Church-Turing postulate has made computer scientists wonder about the intrinsic limitations of algorithmic computation. This hypothesis, named after Alan Turing and Alonzo Church, states that a function on the natural numbers can be calculated by an effective method if and only if it is computable by a Turing machine. Many empirical evidences as well as theoretical confirmations has been found through the years by different, independent teams of researchers leading to a universal agreement on the content of the postulate [1], [2], [3], [4], [5]. Due to the impressive generality of such a result, different formulations have been stated with the purpose of extending the postulate validity for other areas of computer science, such as complexity theory or analog computation. This led to the formulation of the generalized physical form of the Church-Turing hypothesis, which states that any physically realistic (macroscopic) computer is equivalent to Turing machines both in terms of computability and complexity.

Nevertheless, in the context of continuous-time and continuous-space models of computation, no convincing confirmation or disproof of this generalized statement has been provided and universal agreement has not been reached. Therefore, the hope of discovering some super-Turing algorithmic solutions to relevant problems has survived for models of computation that could make use of continuous measurements of complexity. An example of one of these models is Quantum computation, a continuous-space, discrete-time model described by Richard Feynman in 1982 [6] that has the power of allowing significant speedups for the solution of complex problems such as factorization [7]. However, most of these analog or half-analog models of computation present the same type of obstacles for a reasonable characterization of their complexity. One of these obstacles goes under the name of Zeno phenomenon. This phenomenon consists in the possibility of contracting or enlarging the time boundaries of a continuous method of computation without affecting the outcome. This contraction can be applied to the extent of transforming infinite time of computation into finite time, and therefore can generate surprising super-Turing results, such as the ones presented by [8], [9], [10], [11].

This technique can also be used to continuously simulate Turing machines in any desirable short time [12], [13]. A simple way to qualitatively understand how this phenomenon presents itself is through the example of ordinary polynomial differential equations. Let us consider p to be a polynomial on the reals and $x(t) : \mathbb{R} \to \mathbb{R}$ to be the solution of the equation

$$x' = p(x(t)). \tag{1.1}$$

Consider now the following system

$$\begin{cases} y' = p(y(t))z(t) \\ z' = z(t) \end{cases} \tag{1.2}$$

The solution $y(t)$ of the first equation of (1.2) will reproduce the solution $x(t)$ of (1.1) with an exponential speedup, i.e. $y(t) = x(e^t)$, although the ODE is remaining polynomial.

The Zeno phenomenon is a transversal problem that seems to be somehow dependent on the nature of continuous-time and takes place in many different methods of continuous computation [14]. The main model considered for analog computation in this thesis work is the model called GPAC, or *General Purpose Analog Computer*. Claude Shannon introduced this model in 1941 [15] inspired by the idea of giving a strong theoretical foundation to devices like the differential analyzer, a machine made by a system of disks and wheels that could compute solutions of simple differential equations [16]. Shannon worked as an operator on this machine years prior his formulation of the model. A section of this introduction will be dedicated to describe this model and its more recent modifications.

Another model that is taken in consideration is computable analysis. Computable analysis is an approach to computation with real numbers (and, in general, continuous entities) that makes use of type-2 Turing machines, (or, equivalently, of oracle Turing machines) to describe approximations of infinite objects with arbitrary precision. Although this model uses type-2 Turing machines as a base for computations, it is useful to consider computations over a great variety of continuous objects and a great variety of problems of analysis, constituting therefore a paradigm for analog computation. A more detailed treatment of the subject can be found in [17], and a brief description will be given in a section following this introduction.

The two models just listed above present an interesting connection both on a computability and on a complexity level. One important explanation of this connection came in a paper of 2007 by Bournez, Campagnolo, Graça and Hainry [18] where they proved that the GPAC and computable analysis are two equivalent paradigms of analog computation. Specifically, this means that every computable function in the sense of computable analysis can be obtained using a system of polynomial differential equations of the same type of the ones described by the GPAC model, and vice versa. It is important to stress the relevance of this result on the investigation on the generalized Church-Turing postulate. One immediate consequence of this result is acknowledging that the GPAC, a purely continuous (both in time and space) model of computation, is Turing complete, and therefore possesses all the power of digital, discrete computation. This fact constituted a surprise at the time because, due to some technical aspects of the original definition of the GPAC model, the scientific community believed computable analysis to be more powerful in terms of computational resources. Nevertheless, a slight modification on the GPAC definition allowed the authors to boost the computational power of the model and obtain the equivalence. Indeed, within the right set of assumptions, computation by means of ODEs possesses an impressive computational power. For example, it was later shown in 2020 in [19] that there exists a single system of polynomial ODEs whose solutions can approximate up to any desirable precision every continuous function over the reals. The proof of this fact relies on the construction of a universal ordinary differential equation using only polynomial terms.

In 2017 a similar conclusion to what it was done for computability in [18] was reached for the case of polynomial complexity. In [20] the authors found an efficient way to solve the problem posed by the Zeno phenomenon, identifying the length of the curve of the solutions of dynamical systems as a suitable parameter to measure the complexity of the model. Precisely, it has been proved that discrete functions computable in polynomial time correspond to functions which are emulable by polynomial ordinary differential equations of polynomial length. The paper [20] constitutes the main inspiration and reference to this thesis work. However, little was known about the possibility of extending this result to functions (and sets) computable (decidable) in greater times, such as FEXPTIME (EXPTIME). A straightforward extension was indeed hindered by technical complications connected with the requirement of specific composition properties in the structure of the proofs of [20]. In particular it relied on the fact that the class of polynomials is closed under composition. However, closure under composition is not satisfied for the class of exponential functions. Moreover, a treatment of discrete classes defined by space bounds, such as PSPACE, was still missing for this approach. With this thesis work, we found a way to obtain the same characterization of [20] for exponential classes, showing that discrete complexity classes defined by greater time boundaries can always be analyzed with the same analog approach. This extension allowed us to identify some conditions that are sufficient for a greater generalization of the approach, and this lead us to a complete analog characterization of the Grzegorczyk hierarchy, which implies a characterization of all elementary functions and all primitive recursive functions. Moreover, we show that it is also possible to describe with systems of ODEs classes defined by polynomial space boundaries, such as FPSPACE and PSPACE. We believe that these results can provide a deeper understanding of the effectiveness and limitations of the generalized Church-Turing postulate on a complexity level.

Another area in which this thesis work has interesting applications is implicit complexity theory. It is not easy to explain the exact meaning of the word implicit in that expression, because its interpretation can vary depending on the area of interest of the scientific group that is using it. Heuristically, goes under the name of implicit complexity theory the attempt to describe complexity classes without the help of any basic computational device, such as Turing machines. One of the main motivations for this research area is to find an alternative perspective from where to address the most complicate and cryptic theorems of complexity theory. Indeed, it is common opinion between computer scientists that one possible reason to explain the amount of open and unsolved problems in complexity theory (such as the famous P vs NP) could be a lack of mathematical tools able to effectively handle the process of computation through Turing machines. Therefore, establishing a machine-independent way to define complexity classes could circumvent this obstacle, allowing to borrow tools from other well known areas of mathematics and apply them in this new context.

Many machine-independent characterizations of complexity theory have been provided by different research teams from all over the world by means of function algebras [21], rewriting systems, lambda calculus [22], neural networks [23] and many others. However, none of these characterizations present a fully continuous parametrization of the space and time boundaries. Therefore, one important achievement of [20] was to construct the first entirely continuous characterization of the complexity class P. Extending the work done by Bournez, Graça and Pouly this thesis work proves how to systematically characterize any complexity class (of computable functions, or decidable sets) defined by time boundaries (such as FEXPTIME, or EXPTIME), as long as these boundaries satisfies a specific set of properties, such as being closed when composed with polynomials. Indeed, the analysis we made allow us to precisely define a set of conditions on the boundaries which are sufficient for extending the characterization. Following this direction we also show that this characterization procedure can be applied for functions that are computable by Turing machines in time greater than an exponential on the size of the input; specifically, it can be generalized for functions belonging to any of the levels of the Grzegorczyk hierarchy. Concretely, the latter includes all elementary functions as well as all primitive recursive functions. Moreover, according with the spirit of implicit complexity theory, we have been able to formulate an alternative version of the open complexity problem EXPTIME vs PSPACE that only relies on theoretical objects and definitions from analysis.

The remaining part of the introduction aims to illustrate the basics of three of the main areas related to this thesis work. It will be organized in three separate sections, one for dynamical systems and ODEs, one for computable analysis and one for the GPAC and generable functions.

1.1 Dynamical systems and ordinary differential equations

Dynamical systems are arguably the most common mathematical model for the description of physical phenomenons. Their applications extend also to other scientific areas, such as chemistry, finance or biology. Some standard textbooks on the topic are [24], [25]. A dynamical system is a system whose properties evolve over time in a specific external environment modeled by a mathematical space X. Depending on the context X can represent a common Euclidean space or have a more complex structure of a manifold.

A formal mathematical definition of a dynamical system is the following.

Definition 1.1 (Dynamical system) *A dynamical system is a tuple* (T, X, Φ) *where* T *is an additive monoid,* X *is a set and* $\Phi : U \subseteq T \times X \to X$ *is a map satisfying*

- $\Phi(0, x) = x$
- $\Phi(t_2, \Phi(t_1, x)) = \Phi(t_1 + t_2, x)$ *for* $t_1, t_2 \in T$

The monoid T is generally described as the time, the set X as the state or phase space and the map Φ as the flow.

Generally the description of the evolution is given in implicit form, by means of iterations of certain classes of functions or by means of systems of differential equations. The trajectories described over the phase space by the evolving properties are called orbits. If the time evolution of these properties is ruled by a deterministic law, we call the system a deterministic dynamical system. Otherwise, if the evolution depends on some probability distribution, we call the system a stochastic dynamical system. In this thesis work we will deal only with deterministic systems. A detailed analysis of stochastic systems can be found in [24]. Moreover, dynamical systems can be distinguished in two classes: discrete and continuous. Hybrid dynamical systems also exist, as systems that make use of some continuous components and some discrete components, and their use is common within the context of control theory [24], [26]. We will return on some aspects of control theory later in this section.

Discrete time dynamical systems are systems whose evolving properties are measured only at integer times. Examples of discrete dynamical systems are sequences, cellular automata, shifts or Turing machines themselves. From the point of view of computation, the treatment of discrete-time and discrete-space systems is simpler than the continuous case, and much is known on the behavior of these systems. Nevertheless, discrete-time and discrete-space systems are extremely relevant also because they can arise as a possible alternative description of certain classes of differential equations, by means of the Poincaré Map [25]. In some cases, it is still possible to generate extremely complex behaviors just by iterating a function defined on a closed interval. Discrete systems that present such behaviors are the discrete chaotic maps, which are transitive maps that present infinitely many, dense, periodic points on their definition domain and that are extremely sensitive to initial conditions. The logistic map and the horseshoe map are two famous examples of discrete and chaotic dynamical systems [27], [28].

Continuous dynamical systems are systems in which the time is measured continuously, meaning that $t \in \mathbb{R}$. A subclass of continuous systems is the class of smooth dynamical systems, where properties of the systems depend on time in a continuously differentiable manner. The computational task of determine whether the orbit of a continuous systems passes through specific areas of the phase space is called *reachability*. Connected with reachability are key questions from control and verification theory. Indeed if we can fully solve reachability problems related to specific continuous systems, from a computational perspective, then we can provide automated verification tools (where a digital computer is used) for those systems, ensuring that they will behave correctly [29], [30]. Unfortunately the latter is in general a difficult problem, and negative results for reachability also exist in literature; for instance, in [11].

Continuous-time dynamical systems can present chaotic behaviors as well, where the definition of chaos has to be intended in a similar fashion to the discrete case. In dimensions greater than one, these systems can manifest the so called strange attractors, which are attractors whose description is not a simple geometrical object. The most famous example is perhaps the Lorenz attractor, generated from certain solutions of the Lorenz system, a system of differential equations originally proposed to model atmospheric convection phenomenons [31]. Understanding the chaotic behavior of this system constituted one of the eighteen problems proposed by Steve Smale in his list of 1998 [32], and a satisfying description was later given by Warwick Tucker in 2002 [33].

The main category of continuous dynamical systems studied in this thesis work is the one of ordinary differential equations, and, more precisely, the class of *Polynomial Initial Value Problems*, or PIVP. A differential equation is called ordinary if it contains only functions $f, g, ..$ that depend on one independent variable, and the derivatives of those functions $f', f'', .., g', g'', ...$ Other possible classes of differential equations are differential algebraic equations, which are not completely solvable for the derivatives of all components, and partial differential equations, where more than one independent variable and the use of partial derivatives are allowed. Neither of these two classes are taken in consideration by this thesis work. An initial value problem is defined as the mathematical task of finding the solution of a given differential equation when the initial conditions are also given. In general, a method to obtain solutions of a generic initial value problem involving ODEs is not known, and the most common approaches consider only approximation methods for solutions with bounded domains of definition. Areas of research related to this topic are computable analysis and numerical analysis of ODEs, which studies digital implementations of procedures such as the Euler's method. Most of these methods share common ideas inspired by the Euler method and are based on considering discrete sequence of points and time steps, converging to the solution. Each procedure can present a different degree of efficiency and accuracy, depending on the computational speed and precision of the algorithm used to obtain the approximation of the solution. Both the parameters can be estimated by a theoretical analysis of the algorithm prior to any implementation, and they usually depend on the way the considered interval is discretized into separate subintervals [34].

For some specific restricted subclasses of IVP there are theorems that ensure the existence of solutions. One of these particular cases is the case of Lipschitz differential equations.

Definition 1.2 (Lipschitz function) *Let $O \subset \mathbb{R}^n$ be an open set. A function $F : O \to \mathbb{R}^n$ is said to be Lipschitz on O if there exists a constant K such that*

$$|F(X) - F(Y)| \leq K|X - Y| \textit{ for all } X, Y \in O$$

we call K a Lipschitz constant for F. More generally, we say that F is locally Lipschitz if each point in O has a neighborhood O' in O such that the restriction F to O' is Lipschitz. The Lipschitz constant of $F|O'$ may vary with the neighborhoods O'.

Notice that every function F such that F is C^n (the first n derivatives of F exist and are continuous) with $n \geq 1$ is locally Lipschitz. An initial value problem whose definition makes use of functions that satisfy a locally Lipschitz condition on their existence domain can be successfully solved with the help of the following existence and uniqueness theorem [35].

Theorem 1.3 (The existence and uniqueness theorem) *Consider the initial value problem*

$$X' = F(X), \quad X(0) = X_0 \tag{1.3}$$

where $X_0 \in \mathbb{R}^n$. Suppose that $F : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz on $\mathbb{R}^n$. Then there exists a unique solution of this initial value problem. More precisely, there exists $a > 0$ and a unique solution

$$X : (-a, a) \to \mathbb{R}^n$$

of (1.3) satisfying the initial condition $X(0) = X_0$.

A PIVP, or *Polynomial Initial Value Problem* is an IVP whose differential equations are defined using only polynomial terms. As we will see further on in this introduction, the solutions of these problems are closely related to solutions generated by alternative versions of the GPAC.

Concerning the computability and complexity of computing ODEs solutions, an important result given by Graça and Pouly in 2016 will be used extensively in this thesis work. In [36] they presented an algorithm to compute up to any desirable precision the solution of a PIVP on a fixed time interval using oracle Turing machines with a computational time polynomial on the length of the solution on that domain. The speed of the algorithm also depends polynomially on the required precision, the initial condition, and the degree and coefficients of the polynomial defining the PIVP.

For the more general case, where the IVP is defined by a locally Lipschitz differential equation on a bounded domain, it was shown in [37] that the computational complexity of the solution lies in the class of PSPACE problems.

1.2 Computable analysis

The idea of studying computable properties of real numbers and continuous mathematical objects dates back to Alan Turing [38]. Turing stated that a computable real number is one whose decimal (or binary) expansion can be enumerated by a finite procedure on a Turing machine. He subsequently developed the theory of computable functions of computable real numbers, where he considered computable functions defined on these computable numbers. The computable functions are defined not on the reals themselves, however, but on sequences of digits of those reals. Although greatly innovative, this concept contained some problems. Indeed, under this definition of computability, simple operations on the reals such as addition and multiplication are not computable due to the impossibility of obtaining part of the infinite decimal expansion of the outcome of these operations with the only help of finitely many digits of the inputs. Nevertheless, the approach inspired other scientists, such as Daniel Lacombe and Andrzej Grzegorczyk, to propose alternative definitions of what means to compute real numbers in order to solve this problem [39], [40]. Different ways to work with computability and complexity of continuous objects have been proved to be more useful and efficient with respect to others, and vice versa, depending on the main goal of the research. For example, an efficient way to work with real numbers, when the main focus is computational complexity, is to work with *representations* of these numbers. In general, a representation of a continuous object is an infinite string of binary symbols that encodes that object. An approach to real numbers that makes use of representations is to consider real numbers as quickly converging sequences of rational numbers. This is the most popular approach under the framework of computable analysis, and it is the one considered by this thesis work. The sequences of rationals are then encoded in binary and disposed on a infinite string of symbols, forming this way a representation that can be manipulated by the chosen computational devices. The devices can be type-2 machines, following the approach of Klaus Weihrauch [17], or oracle Turing machines, following the approach of Ker-I Ko [41]. In general, for a generic continuous object, such as a set, a number or a function on the reals, many different choices of representations are possible, and the results obtained by these approaches often vary depending on the chosen representation. However, for what concerns computability of real numbers, we will show later in this chapter that the most common chosen representations are proved to be equivalent.

It is important to notice that computable analysis is not the only computational paradigm over the reals considered by the scientific community. Approaches to real computability from Lenore Blum, Mike Shub, and Steve Smale [42] (BSS) and Cristopher Moore [14] yield to very different and in some cases more powerful models of computations. Comparisons between these models can be found in literature: see [43] for a comparison of type-2 computability and Moore's recursion, and [44] for some connections between BSS and computable analysis.

The remaining part of this section will make use of [41] as main reference. We now present three definitions of computable real numbers, each one based on a different representation.

Definition 1.4 (Computable real number) *Let $\mathbb{D}$ be the set of rational dyadic numbers, where a dyadic rational number $d \in \mathbb{D}$ is a rational number that has a finite binary expansion; that is, $d = \frac{m}{2^n}$ for some integers m, n, $n \geq 0$. Assume each dyadic number y to be written in its binary expansion and let $prec(y)$ denotes the number of bits to the right of his binary point*

- *For each real number x, a function $\phi : \mathbb{N} \to \mathbb{D}$ is said to binary converge to x if it satisfies the condition that for all $n \in \mathbb{N}$, $prec(\phi(n)) = n$ and $|\phi(n) - x| \leq 2^{-n}$. Let CF_x (Cauchy function) denote the set of all functions binary converging to x*

- *For each real number x, the set $\{d \in \mathbb{D} | d < x\}$ is called the left cut of x. Let LC_x denote this set*

- *For each real number x, a function $\phi : \mathbb{N} \cup \{-1\} \to \mathbb{N}$, with $\phi(-1) \in \{1, -1\}$ (the sign of x) and $\phi(i) \in \{0, 1\}$ for all $i > 0$, is called the binary expansion of x if it satisfies the equation:*

$$x = \phi(-1)\phi(0) + \sum_{i=1}^{\infty} \phi(i) 2^{-i}$$

  let BE_x denote this function. We have the three following definitions

- *A real number x is CF-computable if CF_x contains a computable function ϕ*

- *A real number x is LC-computable if LC_x is a computable set*

- *A real number x is BE-computable if BE_x is a computable function*

As mentioned above, these different definitions yield to the following equivalence theorem.

Theorem 1.5 *The three definitions of CF-computable numbers, LC-computable numbers and BE-computable numbers are equivalent.*

Working with these definitions of real computable numbers offers the possibility of defining a criterion for measuring complexity in a natural way. However, when the complexity context is introduced, the previous equivalence is not valid any longer for numbers computable in polynomial time. Let us define the complexity classes of real numbers computable in polynomial time following the three definitions listed above.

Definition 1.6 (Real numbers computable in polynomial time) *We give three alternative definitions of real numbers computable in polynomial time*

- *Let P_{CF} be the class of all real numbers for which there is a function $\phi \in CF_x$ that is computable in polynomial time, with the input written in unary notation. Equivalently, a real number x is in P_{CF} if there is a polynomial-time Turing machine M such that for all n, $M(0^n)$ outputs a dyadic rational $d \in \mathbb{D}$ with $prec(d) = n$ and such that $|d - x| \leq 2^{-n}$*
- *Let P_{LC} be the class of all real numbers x for which the set $LC_x = \{d \in \mathbb{D} | d < x\}$ is computable in polynomial time, with inputs d written in their natural binary representations*
- *Let P_{BE} be the class of real numbers x for which the function BE_x is computable in polynomial time, with the input n written in unary notation.*

Then we obtain the following theorem.

Theorem 1.7 $P_{BE} = P_{LC} \subsetneq P_{CF}$

Specifically, both the classes of P_{BE} and P_{LC} are not closed under addition, while P_{CF} instead is a real closed field. Therefore, the most suitable choice is to identify polynomial time computable real numbers with the class P_{CF}.

Following this guideline, if real numbers are seen as functions from integers to rational numbers, or type-1 objects [17], it is intuitive to extend the computable notion also for real functions, that will be considered as operators that map a type-1 object into another. Therefore, a computable real function is viewed as a type-2 object. Because type-1 objects are infinite, any device that computes them have to be able to deal with infinite strings of symbols. This can be done with the help of type-2 machines, or with the help of oracle Turing machines. Here we present how to use oracle Turing machine to this purpose. In general, computable real functions are studied under bounded domains, but extensions to infinite domains are also possible when extra care is applied to take in consideration the size of the input. We present here the definition of computable real function taken from [41].

Definition 1.8 (Computable real function) *A real function $f : \mathbb{R} \to \mathbb{R}$ is computable if there is an oracle Turing machine M such that for each $x \in \mathbb{R}$ and each $\phi \in CF_x$, the function Ψ computed by M with oracle ϕ (i.e., $\Psi(n) = M^{\phi}(n)$) is in $CF_{f(x)}$. We say that the function f is computable on interval $[a, b]$ if the above condition holds for all $x \in [a, b]$.*

Similarly to computable real numbers, there are several different formulations for computable real functions, but they all are proved equivalent [39], [40], [45].

Notice that as a natural consequence of the definition above it follows that a function on the reals needs to be continuous in order to be computable. Indeed, it can be shown that any computable function possesses a recursive modulus function.

Definition 1.9 (Modulus function) *Let $f : [a, b] \to \mathbb{R}$ be a continuous function on $[a, b]$. Then, a function $m : \mathbb{N} \to \mathbb{N}$ is said to be a modulus function of f on $[a, b]$ if for all $n \in \mathbb{N}$ and all $x, y \in [a, b]$, we have*

$$|x - y| \leq 2^{-m(n)} \Rightarrow |f(x) - f(y)| \leq 2^{-n}$$

Theorem 1.10 *If $f : [a, b] \to \mathbb{R}$ is computable on $[a, b]$ then f is continuous on $[a, b]$; furthermore, f has a recursive modulus function m on $[a, b]$.*

An equivalent characterization of computable functions can then be given with the help of modulus functions.

Theorem 1.11 *A function $f : \mathbb{R} \to \mathbb{R}$ is computable if and only if there exist two recursive functions $m : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $\psi : \mathbb{D} \times \mathbb{N} \to \mathbb{D}$ such that*

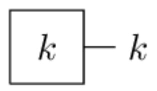
- *for all $k, n \in \mathbb{N}$ and all $x, y \in [-k, k]$, $|x - y| \leq 2^{-m(k,n)}$ implies $|f(x) - f(y)| \leq 2^{-n}$*
- *for all $d \in \mathbb{D}$ and all $n \in \mathbb{N}$, $|\psi(d, n) - f(d)| \leq 2^{-n}$*

In a similar fashion to what it was done for computable reals, it is then straightforward to obtain a notion of polynomial time complexity based on the previous definition of computable functions. We will state that a function f is computable in polynomial time if and only if both the function ψ and the modulus function m of the previous theorem are computable in polynomial time. Analog treatments exist for computability and complexity of sets in $\mathbb{R}^n$ or other continuous objects.

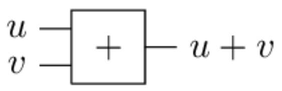
Finally, it is important to mention a model proposed by Akitoshi Kawamura and Stephen Cook [46], where complexity of continuous operators is designed in a general uniform way, so that many weaker results concerning properties of computable functions (maximization, integrations or derivation problems) can be obtained naturally under this single unified model [47].

1.3 GPAC and generable functions

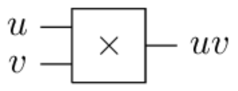
As originally described by Claude Shannon in 1941, the GPAC was a model for analog computation defined by circuits, each one composed by different combinations of several basic units. The basic units represented the atoms of the model, and each of them was able to perform a specific simple operation such a addition, multiplication or integration.



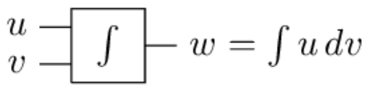
A constant unit



An adder unit



An multiplier unit



An integrator unit

The inspiration for the formulation of the model came from the mechanical device called differential analyzer, invented in 1876 by James Thomson [48], brother of Lord Kelvin. This machine could compute solutions of certain differential equations up to the sixth order, but it required great amount of time and resources to perform its computations, due to the huge size of the wheels and the complex structure of the mechanism. The GPAC model was later applied to describe the behavior of more efficient and developed machines, still based on the example of the differential analyzer, but built using electronic circuits and operational amplifiers. The use of these machines became then obsolete with the growth in popularity of digital computers.

In the first formulation of the GPAC, Shannon claimed that the functions described by his model corresponded with solutions of algebraic differential equations. Given a function f and an interval on the reals I, an algebraic differential equation is a differential equation of the form

$$f(y(t), y'(t), y''(t), ..., y^{(k)}(t)) = 0$$

However, a few years later Marian Boykan Pour-El modified the original construction and consequently presented a new version of the GPAC model in her paper in 1974 [49]. Her version of the model was founded on the concept of quasi-linear differential equations. Another alternative description of the properties and computational power of the model was then given by Daniel Graça and José Félix Costa in [50], where, restricting the amount of possible connections in the circuits, they managed to show that GPAC functions could always be written as components of solutions of PIVPs. Another vantage of their new definition was to allow the model to generate multivariate functions. The type of functions generated by this model are often called *generable functions*. Therefore, this result established an equivalence between the GPAC model and polynomial differential equations.

Given a fixed field $\mathbb{K}$ for the coefficients of the considered polynomials, we present here the definition of the class of functions generated by this version of the model [51]. We call these functions polynomially bounded generable functions with $\mathbb{K}$, or $\mathrm{GPVAL}_{\mathbb{K}}$ [51].

Definition 1.12 ($\mathrm{GPVAL}_{\mathbb{K}}$) *Let I be an open and connected subset of $\mathbb{R}^d$ and $f : I \to \mathbb{R}^e$. Let the notation $J_y(x)$ indicate the Jacobian of y evaluated at point x. We say that $f \in \mathrm{GPVAL}_{\mathbb{K}}$ if and only if there exists a polynomial $sp : \mathbb{R}_+ = [0, +\infty[\to \mathbb{R}_+$, $n \geq e$, a $n \times d$ matrix p consisting of polynomials with coefficients in $\mathbb{K}$, $x_0 \in \mathbb{K}^d \cap I$, $y_0 \in \mathbb{K}^n$ and $y : I \to \mathbb{R}^n$ satisfying for all $x \in I$*

- $y(x_0) = y_0$ *and* $J_y(x) = p(y(x))$
- $f(x) = (y_1(x), .., y_e(x))$
- $\|y(x)\| \leq sp(\|x\|)$

A more general definition of generable functions can be given in the same way with the use of a generic boundary sp. In this case we will refer to functions belonging to $\text{GVAL}_{\mathbb{K}}(sp)$. If the boundary is not specified, then the class $\text{GVAL}_{\mathbb{K}}$ stands for all the functions satisfying the first two conditions of definition 1.12 above, without any condition on the norm of the solution.

The $\text{GPVAL}_{\mathbb{K}}$ class can be seen as an extended version of functions generated by PIVPs. Indeed, when I is an interval, the Jacobian of y simply becomes the derivative of y and we get the solutions of $y' = p(y)$ where p is a vector of polynomials. This class contains many of the most common elementary functions. Indeed, all polynomials are in the class, as well as sin, cos and arctan functions. Although functions in $\text{GPVAL}_{\mathbb{K}}$ can be viewed as solutions of partial differential equations (PDEs) (as we use a Jacobian), we will never have to deal with classical problems related to PDEs. Indeed, PDEs have no general theory about the existence of solutions. Therefore, in this thesis work, we will explicitly present the functions in $\text{GPVAL}_{\mathbb{K}}$ which will be used and we will show that they satisfy the conditions of the previous definition. Note also that it can be shown [Remark 15] [51] that a solution to the PDE defined with the Jacobian is unique, because the condition $J_y(x) = p(y(x))$ is not general enough to capture the class of all PDEs. We also remark that, because a function in $\text{GPVAL}_{\mathbb{K}}$ must be polynomially bounded, it is defined everywhere on I.

This class is more stable than the original class of generable functions, and satisfies important composition properties; namely, it is closed under composition, addition and subtraction [51]. Another crucial property of the class $\text{GPVAL}_{\mathbb{K}}$ is that it is closed under solutions of ODEs. In practice, this means that we can write differential equations of the form $y' = g(y)$ where g is generable, knowing that this can always be rewritten as a PIVP. The following lemma is taken from [20].

Lemma 1.13 (Closure by ODE of $\text{GPVAL}_{\mathbb{K}}$**)** *Let* $J \subseteq \mathbb{R}$ *be an interval,* $f : \mathbb{R}^d \to \mathbb{R}^d \in \text{GPVAL}_{\mathbb{K}}$*,* $t_0 \in \mathbb{Q} \cap J$ *and* $y_0 \in \mathbb{Q}^d \cap \operatorname{dom} f$*. Assume there exists* $y : J \to \operatorname{dom} f$ *and a polynomial* $sp : \mathbb{R}_+ \to \mathbb{R}_+$ *satisfying for all* $t \in J$

- $y(t_0) = y_0$ *and* $y'(t) = f(y(t))$
- $\|y(t)\| \leq sp(t)$

Then y *is unique and belongs to* $\text{GPVAL}_{\mathbb{K}}$*.*

The $\text{GPVAL}_{\mathbb{K}}$ class is the milestone on which are based all the main classes presented in [20], as well as the new classes created during this thesis work, that will be introduced in the upcoming chapters.

One of the interests of this thesis work is to study the connection between this latest version of the GPAC model and the computational models based on Turing machines. As described in previous sections of this introduction, this connection was already established first on a computability level [18], and then on a complexity level in case of polynomial time complexity [20]. However, based on its original definition, the GPAC is a model that outputs the solution in real time: the result of its computation is the graph of the function itself, while the most interesting aspect of Turing machines computations is not the whole list of configurations but the asymptotic behavior, i.e. the way it reaches the accepting or rejecting state. Therefore, to be able to construct their equivalence, the authors needed an asymptotic version of the GPAC model. This asymptotic version of the polynomially bounded generable functions defined above is the one used by the authors in [20], and will be the one considered and extended in the course of this thesis work.

Following the definition of polynomially bounded generable functions it is possible to define the concept of generable fields [50]. For the purpose of this thesis work, the key property of generable fields to consider is the following one.

Theorem 1.14 *Let $\mathbb{K}$ be a generable field. If $\alpha \in \mathbb{K}$ and f is generable using coefficients in $\mathbb{K}$ (i.e. $f \in \mathrm{GPVAL}_{\mathbb{K}}$) then $f(\alpha) \in \mathbb{K}$.*

Without the help of this property it would be possible to have some fields $\mathbb{K}$ for which the relative classes $\mathrm{GPVAL}_{\mathbb{K}}$ are not closed under composition, which is not natural and would create a great obstacle to our work. For the whole construction built in [20] to hold, in order to be able to continuously simulate computations of Turing machines, the coefficients field $\mathbb{K}$ used for defining the main analog classes has to be a generable field. Unfortunately, $\mathbb{Q}$ is not a generable field. Nevertheless, It has been shown in [51] that there exists a generable field $\mathbb{R}_G$ lying somewhere between $\mathbb{Q}$ and $\mathbb{R}_P$: $\mathbb{Q} \subseteq \mathbb{R}_G \subseteq \mathbb{R}_P$, where $\mathbb{R}_P$ stands for the class of real numbers computable in polynomial time. Moreover, the class $\mathbb{R}_P$ is a generable field. Unless stated otherwise, we will assume the coefficients of our functions to belong to $\mathbb{R}_P$ throughout all the rest of this thesis work. This will allow to shortly indicate the class as GPVAL, where the presence of the field $\mathbb{R}_P$ is implicit.

1.4 Related work

The goal of this section is to offer to the reader a concise and unified picture of which other works and papers could be associated with the content of this thesis. Therefore, we will be listing again some of the articles and publications we previously mentioned in this chapter, as well as include works that can be ideally located closer or further away from this thesis, but that all share with it similarities in terms of topics, arguments or approach.

The first field to mention has to be the study of differential equations from the perspective of computability and complexity theory. Specifically, the latter ranges from studies over the computability properties of IVP, such as the problem of computing maximal intervals and the decidability of their domain of existence [52], [53], [54] to studies over complexity properties of smooth differential equations [55] or of ordinary differential equations such as the ones in [56], [36], [57]. Investigations over the complexity of solving ODEs have also treated the case of IVPs with Lipschitz continuous right-hand terms, and in [37] have been proved the PSPACE-completeness of such problem.

Extremely important connections with this thesis work have also to be found with the vast area of research conducted on analog models of computation, and specifically on the GPAC model and models based on systems of ODEs. A survey on the subject of continuous-time models of computation have been produced in [58]. As an example, we can mention here the model of signal machines [59]. Formulations of complexity measures and boundaries for continuous-time models are discussed in [60], [61]. The efficiency of the GPAC model in terms of computational power has been carefully analyzed in [51]. The work illustrated in [62], [63] demonstrate its equivalence with the model of computable analysis, and this contribution has been deeply considered and applied to achieve the polynomial characterization result in [20], which represents the main inspiration of this thesis work.

Moreover, different ways of exploiting systems of ODEs to simulate Turing machines are well described in [64], [65], [18] and a particular focus over robust simulation has been developed in [66]. This last point of view has been particularly relevant in our treatment of the FPSPACE characterization. Concerning our extension of the analog characterization of FP in [20] to elementary functions, similar ideas and similar goals can be found in [67], [68]; moreover the authors of [69] obtained an analog characterization of the Grzegorczyk hierarchy by means of systems of nonlinear differential equations. However, the characterization of [69] is achieved using nonanalytic functions, with an approach that is more recursive compared to the one developed in this thesis work. Indeed, our characterization is obtained just by modifying resources on our computational model based on systems of polynomial ODEs.

Finally, the problem of computing analytic branches of the complex square root over simply connected domains have been originally presented in [70], while similar studies connected with the complexity of winding numbers could be found in [71]. Moreover, computational complexity of problems related to two dimensional regions has been studied in several directions, including the study of Julia sets [72], [73], the study of fractals [74], [75], and the study of the path-finding problems [76], [77].

Chapter 2

Computing with polynomial differential equations

In this chapter we will review several definitions and results from [20], which were used there to present an analog characterization of functions computable in polynomial time FP. We will also show how to continuously perform simulations of Turing machine computations in order to achieve this goal. This whole chapter does not contain any fresh result; however, the concepts introduced here are necessary for the understanding of our original contribution. The reader can find technical details and missing proofs in [20].

2.1 Definitions of the analog classes

The classes that will be defined in this chapter represent an asymptotic version of definition 1.12, and they can be proved to be all equivalent. A deeper treatment of the subject can be found in [51], where other equivalent classes, out of the scope of this thesis work, are introduced. All the functions belonging to the following analog classes share the same common principle, which is the key concept for defining a correct criterion of how to measure complexity in this type of dynamical systems: they are related to solutions of polynomial differential equations with length polynomially bounded on the norm of their input. This concept inspires the following definition of *Analog-Length-Computable* functions, or ALC.

Definition 2.1 (ALC) *Let $f \subseteq \mathbb{R}^n \to \mathbb{R}^m$, and $\Pi : \mathbb{R}_+^2 \to \mathbb{R}_+$. We say that f is Π-length-analog-computable if and only if there exist $d \in \mathbb{N}$, $p \in \mathbb{R}_P^d[\mathbb{R}^d]$ and $q \in \mathbb{R}_P^d[\mathbb{R}^n]$ such that for any $x \in \operatorname{dom} f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$*

- *$y(0) = q(x)$ and $y'(t) = p(y(t))$*
- *$\forall \mu \in \mathbb{R}_+$ if $len_y(0,t) \geq \Pi(\|x\|, \mu)$ then $\|(y_1(t), .., y_m(t)) - f(x)\| \leq e^{-\mu}$*
- *$\|y'(t)\| \geq 1$*

where $\mathbb{R}_P^d[\mathbb{R}^n]$ indicates a polynomial in n variables with coefficients in $\mathbb{R}_P^d$; $len_y(0,t)$ represents the length of y from time 0 to time t and as a norm function we are using the canonical infinity norm, $\|x\| = max_{i=1,2,..,m}(|x_i|)$ for $x \in \mathbb{R}^m$. We recall that the notation $C^k(\mathbb{R}^m, \mathbb{R}^n)$ represents the set of functions from $\mathbb{R}^m$ to $\mathbb{R}^n$ whose first k derivatives exist and are continuous, and that the length of a curve $y \in C^1(I, \mathbb{R}^n)$ defined over some interval $I = [a, b]$ is given by $len_y(a, b) = \int_I \|y'(t)\| \, dt$. We denote by ALC(Π) the class of Π-length-computable functions, and by ALP the class of all analog-length-computable functions bounded by a polynomial. Other two alternative ways of describing the functions in ALP, all proved to be equivalent [51], are the following ones.

Definition 2.2 (ATSP) *Let $f \subseteq \mathbb{R}^n \to \mathbb{R}^m$. We say that $f \in$ ATSP if and only if there exist $d \in \mathbb{N}$, $p \in \mathbb{R}^d_P[\mathbb{R}^d]$, $q \in \mathbb{R}^d_P[\mathbb{R}^n]$ and polynomials $\Pi : \mathbb{R}^2_+ \to \mathbb{R}_+$ and $\Upsilon : \mathbb{R}^2_+ \to \mathbb{R}_+$ such that for any $x \in \operatorname{dom} f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$*

- $y(0) = q(x)$ *and* $y'(t) = p(y(t))$
- $\forall \mu \in \mathbb{R}_+$ *if* $t \geq \Pi(\|x\|, \mu)$ *then* $\|(y_1(t), y_2(t), ..y_m(t)) - f(x)\| \leq e^{-\mu}$
- $\|y(t)\| \leq \Upsilon(\|x\|, t)$

Definition 2.3 (AOP) *Let $f \subseteq \mathbb{R}^n \to \mathbb{R}^m$. We say that $f \in$ AOP if and only if there exist $\delta \geq 0$, $d \in \mathbb{N}$, $p \in \mathbb{R}^d_P[\mathbb{R}^d \times \mathbb{R}^n]$, $y_0 \in \mathbb{R}^d_P$ and polynomials $\Pi, \Upsilon, \Lambda : \mathbb{R}^2_+ \to \mathbb{R}_+$ such that for any $x \in C^0(\mathbb{R}_+, \mathbb{R}^n)$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$*

- $y(0) = q(x)$ *and* $y'(t) = p(y(t), x(t))$
- $\|y(t)\| \leq \Upsilon(\sup_\delta \|x\|(t), t)$
- $\forall I = [a, b] \subseteq \mathbb{R}_+$ *if there exist* $\bar{x} \in \operatorname{dom} f$ *and* $\bar{\mu} \geq 0$ *such that* $\forall t \in I$, *we have* $\|x(t) - \bar{x}\| \leq e^{-\Lambda(\|x\|, \bar{\mu})}$ *then* $\|(y_1(u), y_2(u), ..y_m(u)) - f(x)\| \leq e^{-\bar{\mu}}$ *whenever* $a + \Pi(\|\bar{x}\|, \bar{\mu}) \leq u \leq b$

where with the notation $\sup_\delta f(t)$ for a generic function f over the reals we indicate the supremum of f on a small interval δ around t, i.e. $\sup_\delta f(t) = \sup_{[t, t-\delta] \cap \mathbb{R}_+} f(u)$.

Among these three different definitions, the one considered and modified to obtain the main results of this work is the class of *Analog-Time-Space-Polynomial-Computable functions*, or ATSP. Both ALP and AOP (*Analog-Online-Polynomial-Computable functions*) will just be used as alternative resources to demonstrate proofs or properties of the considered functions. The reason for this choice has to be found in the analogies of the ATSP definition with the world of digital computations by means of Turing machines. Indeed, this class presents two boundaries functions, Π and Υ, whose role in the dynamical system is similar to the one played by the number of steps and the number of visited cells for Turing machine computations. Namely, Υ controls the absolute value of the solution y, which can be thought as the *space* growth of the system, and Π controls the rate of convergence to the desired value of the function f, which can be thought as the *time* boundary of the system. For this reason, during this thesis work we will refer to Π as the *time boundary function* and to Υ as the *space boundary function* for our dynamical systems. In general, given two polynomials Π and Υ we will write $f \in$ ATSP(Π, Υ) as a shortcut notation meaning that $f \in \mathrm{ATSP}$ with time boundary Π and space boundary Υ.

The ATSP class presents a close connection with the class GPVAL; precisely, GPVAL $\subseteq$ ATSP when defined over star domains.

Definition 2.4 (Star domain) *A set X is called a star domain if there exists $x_0 \in X$ such that for all $x \in X$ the line segment from x_0 to x is in X, i.e. $[x_0, x] \subseteq X$. Such an x_0 is called a vantage point.*

Given this definition, we can then state the following theorem taken from [20].

Theorem 2.5 *If $f \in \mathrm{GPVAL}$ has a star domain with a generable vantage point, then $f \in \mathrm{ATSP}$.*

The intuition behind the above theorem comes from the fact that on a star domain, once we know that the solution of our GPVAL dynamical system exists between two different points of the domain, we can always consider as trajectory the straight line connecting those points and being sure that the solution of the asymptotic definition of ATSP converges to the correct value over a parametrization of that line. From now on throughout the rest of this thesis work, if not specified otherwise, we will always deal with domains of definition of analog classes that will be star domains. Indeed, we will mostly consider domains of the form $\mathbb{R}^n \times \mathbb{R}^m$, which happen to be star domains.

2.2 Properties of the ATSP class

The class of ATSP functions is an interesting class also because it can be shown that it possesses many useful properties [51],[20]. We will now present a list of the most important properties repeatedly used during this thesis work.

Theorem 2.6 (ATSP closure by arithmetic operations) *If $g, f \in \mathrm{ATSP}$, then $g + f,\ g - f,\ g \cdot f\ \in \mathrm{ATSP}$, with the obvious restrictions on the domains of definition.*

Theorem 2.7 (ATSP modulus of continuity) *If $f \in \mathrm{ATSP}$, then f admits a polynomial modulus of continuity: there exists a polynomial $\Omega : \mathbb{R}^2_+ \to \mathbb{R}_+$ such that $\forall x, y \in \operatorname{dom} f$ and $\mu \in \mathbb{R}_+$*

$$\|x - y\| \leq e^{-\Omega(\|x\|,\mu)} \quad \Rightarrow \quad \|f(x) - f(y)\| \leq e^{-\mu}$$

in particular, f is continuous.

One the most important properties, that will present a concrete obstacle for a straightforward extension of polynomial results (concerning ATSP) to different and greater boundaries (concerning extended analog classes) is the composition property.

Theorem 2.8 (ATSP composition) *If $f, g \in \mathrm{ATSP}$, and $f(\operatorname{dom} f) \subseteq$ dom g, then $g \circ f \in \mathrm{ATSP}$.*

Because of the key role played by this particular property in our work, we present also here the full proof of the theorem [20].

Proof of theorem 2.8. Let $f : I \subseteq \mathbb{R}^n \to J \subseteq \mathbb{R}^m$ and let $g : J \to K \subseteq \mathbb{R}^l$. We will show that $g \circ f \in$ ATSP by using the fact that g satisfies definition 2.3.

Indeed, if $g \in$ ATSP, by equivalence of the classes, we get that there exist polynomials Π, Υ, Λ such that $g \in \mathrm{AOP}(\Pi, \Upsilon, \Lambda)$ with corresponding r, δ, z_0, where r is the polynomial defining the AOP differential equation, δ is the same parameter of the original definition of AOP, and z_0 is the initial condition of the system. In the same way we obtain that there exist two polynomial boundaries Π', Υ' such that $f \in \mathrm{ATSP}(\Pi', \Upsilon')$ with corresponding d, p, q parameters for dimension and polynomials defining the system. Assume, without loss of generality, that all the functions used as boundaries for definitions of f and g are increasing functions. Let $x \in I$ and consider the following system

$$\begin{cases} y(0) = q(x) \\ y'(t) = p(y(t)) \end{cases} \quad \begin{cases} z(0) = z_0 \\ z'(t) = r(z(t), (y_1(t), y_2(t), ..y_m(t))) \end{cases} \quad \begin{cases} x(0) = x \\ x'(t) = 0 \end{cases} \tag{2.1}$$

Define now $v(t) = (x(t), y(t), z(t))$. Then it immediately follows that v satisfies a PIVP of the form

$$\begin{cases} v(0) = \mathrm{poly}(x) \\ v'(t) = \mathrm{poly}(y(t)) \end{cases} \tag{2.2}$$

Furthermore, by definition of v we have

$$\begin{aligned} \|v(t)\| &= \max(\|x(t)\|, \|y(t)\|, \|z(t)\|) \\ &\leq \max(\|x\|, \|y(t)\|, \Upsilon(\sup_{u\in[t,t-\delta]} \|y_{1..m}(u)\|, t)) \\ &\leq \mathrm{poly}(\|x\|, \sup_{u\in[t,t-\delta]} \|y(u)\|, t) \\ &\leq \mathrm{poly}(\|x\|, \sup_{u\in[t,t-\delta]} \Upsilon'(\|x\|, u), t) \\ &\leq \mathrm{poly}(\|x\|, t) \end{aligned} \tag{2.3}$$

where with the notation *poly* we indicate an unspecified polynomial. Now define $\bar{x} = f(x)$, $\Upsilon^*(\alpha) = 1 + \Upsilon'(\alpha, 0)$ and $\Pi''(\alpha, \mu) = \Pi'(\alpha, \Lambda(\Upsilon^*(\alpha), \mu)) + \Pi(\Upsilon^*(\alpha), \mu)$. From the fact that $f \in$ ATSP (Π', Υ'), we get $\|\bar{x}\| \leq \Upsilon'(\|x\|, 0) + 1 = \Upsilon^*(\|x\|)$. Let $\mu > 0$, then by definition of Π', if $t \geq \Pi'(\|x\|, \Lambda(\Upsilon^*(\|x\|), \mu))$ then $\|(y_1(t), y_2(t), ..y_m(t)) - \bar{x}\| \leq e^{-\Lambda(\Upsilon^*(\|x\|),\mu))} \leq e^{-\Lambda(\|x\|,\mu)}$. Therefore, if we consider $a = \Pi'(\|x\|, \Lambda(\Upsilon^*(\|x\|), \mu))$ we get that

$$\|(v_1(t), v_2(t), ..v_l(t)) - g(f(x))\| \leq e^{-\mu} \quad \forall t \geq a + \Pi(\|\bar{x}\|, \mu) \tag{2.4}$$

and since $t \geq a + \Pi(\|\bar{x}\|, \mu)$ whenever $t \geq \Pi''(\|x\|, \mu)$, and Π'' is a polynomial, we get as a consequence that $g \circ f \in$ ATSP. ∎

Another interesting type of closure property for this class is a property that we call closure by GPVAL-ODEs of ATSP. This property ensures that using functions in the GPVAL class as right-hand terms of ordinary differential equations for initial value problems always generates as solutions functions that belong to the ATSP class.

Theorem 2.9 (Closure of by GPVAL-ODEs of ATSP) *The class* ATSP *remains the same if instead of restricting p, q to be polynomial in definition 2.2, we allow the extension $p, q \in$* GPVAL.

Finally, we state one last property related to the maximum values of these functions.

Theorem 2.10 (Polynomial bound of ATSP) *Let $f \in$ ATSP, then there exists a polynomial P such that $\|f(x)\| \leq P(\|x\|)\ \ \forall x \in \operatorname{dom} f$.*

2.3 Simulating a Turing machine

In the main result of [20], the authors proved an equivalence relation between the class ATSP and the standard class of polynomial computable functions FP. To prove this result, they designed a way to simulate a regular computation of a standard Turing Machine in a continuous context, reproducing the discrete step-evolution of the machine with a particular system of polynomial differential equations. In this section we will present some of the milestones of this construction, but we suggest the careful reader to read the original article for a full description of the details. We will simulate deterministic, one-tape Turing machines, with complete transition functions.

As just showed at the end of the previous section, the classes of functions that are used to construct the simulation are classes that include only continuous functions. This fact implies that, to be able to reproduce a computation using these classes it is essential to find suitable continuous substitutes for all the discrete operations that would be required for perfectly simulate a Turing machine using real numbers, such as rounding a number to its closest integer, evaluating norms of vectors, or taking the absolute value of certain numbers.

Using some care, it is possible to select in ATSP good candidates to replace the role of these discrete operations at a cost of introducing a small, controlled error, that will be kept bounded along the course of the whole simulation. It follows a list of results that make the latter possible. All the necessary proofs are included in [20].

2.4 Useful functions in ATSP

Theorem 2.11 (Absolute value, maximum and minimum) *The real functions $x \to |x|$, max, min $\in$ ATSP.*

Theorem 2.12 *Let $rnd^*(x, \mu) \in C^0(\mathbb{R}, \mathbb{R})$ be the unique function such that*

- *$rnd^*(x, \mu) = n \quad \forall x \in [n - \frac{1}{2} + e^{-\mu}, n + \frac{1}{2} - e^{-\mu}]\ \ \forall n \in \mathbb{Z}$*
- *$rnd^*(x, \mu)$ is affine over $[n + \frac{1}{2} - e^{-\mu}, n - \frac{1}{2} + e^{-\mu}]\ \ \forall n \in \mathbb{Z}$*

Then the function $rnd^* \in$ ATSP.

Theorem 2.13 (Norm) *For every $\delta \in]0, 1[$ there exists a real function $norm_{\infty,\delta}(x) \in$ ATSP such that for any $x \in \mathbb{R}^n$ we have*

$$\|x\| \leq norm_{\infty,\delta}(x) \leq \|x\| + \delta$$

Other two crucial functions are lxh and hxl that allow to smoothly approximate a step function in a continuous manner.

Proposition 2.14 (Low-X-High and High-X-Low) *For every* $I = [a, b]$, $a, b \in \mathbb{R}_P$ *there exist two real functions* $\mathrm{lxh}_I, \mathrm{hxl}_I \in \mathrm{ATSP}$ *such that for any* $\mu \in \mathbb{R}_+$ *and* $x, t \in \mathbb{R}$ *we have*

- lxh_I *is of the form* $\mathrm{lxh}_I(t, \mu, x) = \phi_1(t, \mu, x)x$ *where* $\phi_1(t, \mu, x) \in \mathrm{ATSP}$
- hxl_I *is of the form* $\mathrm{hxl}_I(t, \mu, x) = \phi_2(t, \mu, x)x$ *where* $\phi_2(t, \mu, x) \in \mathrm{ATSP}$
- *If* $t \le a$, $|\mathrm{lxh}_I(t, \mu, x)| \le e^{-\mu}$ *and* $|x - \mathrm{hxl}_I(t, \mu, x)| \le e^{-\mu}$
- *If* $t \ge b$, $|x - \mathrm{lxh}_I(t, \mu, x)| \le e^{-\mu}$ *and* $|\mathrm{hxl}_I(t, \mu, x)| \le e^{-\mu}$
- *In all cases,* $|\mathrm{lxh}_I(t, \mu, x)| \le x$ *and* $|\mathrm{hxl}_I(t, \mu, x)| \le x$

We will describe now the notation used in this thesis work for the real encoding of a Turing machine.

2.5 Notation used for Turing machines

Definition 2.15 (Turing Machine) *A Turing machine is defined as a tuple* $M = (Q, \Sigma, b, \delta, q_0, q_\infty)$, *where* $Q = \{0, ..., m\}$ *are the states of the machine,* $\Sigma = \{0, ..., k-2\}$ *is the alphabet and* $b = 0$ *is the blank symbol,* $q_0 \in Q$ *is the initial state,* $q_\infty \in Q$ *is the halting state, and* $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, S, R\}$ *is the transition function with* $L = -1, S = 0, R = 1$. *We write* $\delta_1, \delta_2, \delta_3$ *as the components of* δ. *That is* $\delta(q, w) = (\delta_1(q, w), \delta_2(q, w), \delta_3(q, w))$ *where* δ_1 *is the new state,* δ_2 *the new symbol, and* δ_3 *the head move direction. We require that* $\delta(q_\infty, w) = (q_\infty, w, S)$.

Definition 2.16 (Configuration) *A configuration for a Turing machine* M *is a tuple* $c = (x, \sigma, y, q)$, *where* $x \in \Sigma^*$ *is the part of the tape at the left of the head,* $y \in \Sigma^*$ *is the part at the right,* $\sigma \in \Sigma$ *is the symbol under the head, and* $q \in Q$ *is the current state.*

More precisely, x_1 *is the symbol immediately at the left of the head and* y_1 *is the symbol immediately at the right. The set of configurations of* M *is denoted as* C_M. *The initial configuration is defined by* $c_0(w) = (\lambda, b, w, q_0)$ *and the final configuration by* $c_\infty(w) = (\lambda, b, w, q_\infty)$ *where* λ *is the empty word.*

Definition 2.17 (Step function) *The step function of a Turing machine* M *is the function, acting on configurations, denoted by* M *and defined by*

$$M(x, \sigma, y, q) = \begin{cases} (\lambda, b, \sigma' y, q') \ \ if \ \ d = L \ \ and \ \ x = \lambda \\ (x_{2..|x|}, x_1, \sigma' y, q') \ \ if \ \ d = L \ \ and \ \ x \neq \lambda \\ (x, \sigma', y, q') \ \ if \ \ d = S \\ (\sigma' x, b, \lambda, q') \ \ if \ \ d = R \ \ and \ \ y = \lambda \\ (\sigma' x, y_1, y_{2..|y|}, q') \ \ if \ \ d = R \ \ and \ \ y \neq \lambda \end{cases} \quad \begin{cases} q' = \delta_1(q, \sigma) \\ \sigma' = \delta_2(q, \sigma) \\ d = \delta_3(q, \sigma) \end{cases} \tag{2.5}$$

Definition 2.18 (Result of a computation) *The result of a computation of M on input $w \in \Sigma^*$ is defined by*

$$M(w) = \begin{cases} x & if \ \exists\, n \in \mathbb{N} \ \ such\ that \ \ M^{[n]}(c_0(w)) = c_\infty(x) \\ \perp & \ otherwise \end{cases} \tag{2.6}$$

Some of the finite and discrete transitions of the machine can be continuously reproduced by means of an interpolation scheme, in the same way the Lagrange polynomial can describe certain functions in dimension one.

Theorem 2.19 (Finite Interpolation) *For any finite $G \subseteq \mathbb{R}_P^d$ and $f : G \to \mathbb{R}_P$ there exists a function $\mathbb{1}_f \in$ ATSP with $(\mathbb{1}_f)|_G = f$, where $(\mathbb{1}_f)|_G$ denotes the restriction of $\mathbb{1}_f$ to G.*

The most important functions to interpolate in this context are the characteristic functions.

Theorem 2.20 (Characteristic interpolation) *For any finite $G \subseteq \mathbb{R}_P^d$, $f : G \to \mathbb{R}_P$ and $\alpha \in \mathbb{R}_P$ define the functions $\mathbb{D}_{f=\alpha}$, $\mathbb{D}_{f\neq\alpha} : \mathbb{R}^d \to \mathbb{R}$ in the following manner*

$$\mathbb{D}_{f=\alpha} = \mathbb{1}_{f_\alpha}(x) \quad and\ \mathbb{D}_{f\neq\alpha} = \mathbb{1}_{1-f_\alpha}(x)$$

where

$$f_\alpha(x) = \begin{cases} 1 & if \ \ f(x) = \alpha \\ 0 & otherwise \end{cases} \tag{2.7}$$

Then $\mathbb{D}_{f=\alpha}$, $\mathbb{D}_{f\neq\alpha} \in$ ATSP.

Once we have clarified the notation used in this thesis work for configurations of Turing machines, we can proceed to encode them as tuples of rational numbers.

2.6 Real encoding of Turing machines

Definition 2.21 (Real encoding) *Let $c = (x, w, y, q)$ be a configuration of M. Then the real encoding of c is defined as $\langle c \rangle = (0.x, \sigma, 0.y, q) \in \mathbb{Q}\times\Sigma\times\mathbb{Q}\times Q$ where $0.x = x_1k^{-1} + x_2k^{-2} + ... + x_{|w|}k^{|w|} \in \mathbb{Q}$.*

Following the same principle, with a suitable encoding, we want to transform also the step function in an operation over real numbers. To do so we need to allow the system to commit small errors during the computation. This is indeed necessary to obtain continuity during the evolution. Therefore one crucial aspect is to maintain stability for these type of dynamical systems, which means that the error introduced here cannot accumulate and grow over some fixed threshold during the course of the whole evolution.

First, we will define the *ideal real step*: an encoding of the step function that does not admit errors and then we will describe a modified one, introducing errors, that we will call *real step* and that will make use of some of the ATSP functions defined in the previous chapter.

Theorem 2.22 (Ideal real step) *The ideal step of a Turing machine M is the function defined over $\langle C_M \rangle$ by*

$$\langle M\rangle_\infty = \begin{cases} (frac(k\tilde{x}), int(k\tilde{x}), \frac{\sigma'+\tilde{y}}{k}, q') & if \;\; d = L \\ (\tilde{x}, \sigma', \tilde{y}, q') \;\; if \;\; d = S \\ (\frac{\sigma'+\tilde{x}}{k}, int(k\tilde{y}), frac(k\tilde{y}), q') & if \;\; d = R \end{cases} \qquad \begin{cases} q' = \delta_1(q,\sigma) \\ \sigma' = \delta_2(q,\sigma) \\ d = \delta_3(q,\sigma) \end{cases} \tag{2.8}$$

where $int(x) = \max(0, \lfloor x \rfloor)$; $frac(x) = x - int(x)$. Then, for any machine M and configuration c, $\langle M\rangle_\infty(\langle c\rangle) = \langle M(c)\rangle$.

Definition 2.23 (Real step) *For any $\tilde{x}, \tilde{y}, \tilde{\sigma}, \tilde{q}$ and $\mu \in \mathbb{R}$, define the real step function of a Turing machine M by*

$$\langle M\rangle(\tilde{x}, \tilde{y}, \tilde{\sigma}, \tilde{q}, \mu) = \langle M\rangle^*(\tilde{x}, rnd^*(\tilde{\sigma}, \mu), \tilde{y}, rnd^*(\tilde{q}, \mu)))$$

where $\langle M\rangle^*(\tilde{x}, \tilde{y}, \tilde{\sigma}, \tilde{q}, \mu) = \langle M\rangle^\star(\tilde{x}, \tilde{y}, \mathbb{1}_{\delta_1}(\tilde{q}, \tilde{\sigma}), \mathbb{1}_{\delta_2}(\tilde{q}, \tilde{\sigma}), \mathbb{1}_{\delta_3}(\tilde{q}, \tilde{\sigma}), \mu)))$

and

$$\langle M\rangle^\star(\tilde{x}, \tilde{y}, \tilde{q}, \tilde{\sigma}, \tilde{d}, \mu) = \begin{cases} choose[frac^*(k\tilde{x}), \tilde{x}, \frac{\tilde{\sigma}+\tilde{x}}{k}] \\ choose[int^*(k\tilde{x}), \tilde{\sigma}, int^*(k\tilde{y})] \\ choose[\frac{\tilde{\sigma}+\tilde{x}}{k}, \tilde{y}, frac^*(k\tilde{y})] \\ \tilde{q} \\ \mu \end{cases} \tag{2.9}$$

where

$$choose[l, s, r] = \mathbb{D}_{id=L}(\tilde{d})l + \mathbb{D}_{id=S}(\tilde{d})s + \mathbb{D}_{id=R}(\tilde{d})r$$

$$int^*(x) = rnd^*(x - \frac{1}{2} + \frac{1}{2k}, \mu + \; ln \; k) \quad , \quad frac^*(x) = x - int^*(x).$$

It has been proved in [20] that the real step defined this way belongs to ATSP. Moreover, it is a robust operator, meaning that it maintains the final error on the output proportional to the initial error on the input. The latter is clarified by the following theorem.

Theorem 2.24 (Real step is robust) *For any machine M, $c \in C_M$, $\mu \in \mathbb{R}_+$, and $\tilde{c} \in \mathbb{R}^4$, if $\|\langle c\rangle - \tilde{c}\| \leq \frac{1}{2k^2} - e^{-\mu}$, then*

$$\|(\langle M\rangle(\tilde{c}, \mu)_1, \langle M\rangle(\tilde{c}, \mu)_2, \langle M\rangle(\tilde{c}, \mu)_3, \langle M\rangle(\tilde{c}, \mu)_4) - \langle M(c)\rangle\| \leq k \, \|\langle c\rangle - \tilde{c}\| \tag{2.10}$$

Furthermore, $\langle M\rangle \in$ ATSP.

2.7 Continuous simulation of Turing machines computations

To prove the equivalence between ATSP and FP this real step has to be iterated for a polynomial number of times, so that the outcome of any computation of a given Turing machine computing functions in FP can be fully reproduced in a continuous manner. To do so, we will make use of a theorem that ensures us that indeed there exists a function in ATSP that computes iterations of the real step.

Theorem 2.25 (Polynomial iterations of real step) *Let $I \subseteq \mathbb{R}^m$, $f : I \to \mathbb{R}^m \in$ ATSP, $\eta \in [0, \frac{1}{2}]$ and assume there exists a family of subsets $I_n \subseteq I$, for all $n \in \mathbb{N}$ and polynomials $\Omega : \mathbb{R}_+ \to \mathbb{R}_+$, $\Pi : \mathbb{R}_+^2 \to \mathbb{R}_+$ such that $\forall n \in \mathbb{N}$*

- $I_{n+1} \subseteq I_n \quad$ *and* $\quad f(I_{n+1}) \subseteq I_n$
- $\forall x \in I_n \quad \left\|f^{[n]}(x)\right\| \leq \Pi(\|x\|, n)$
- $\forall x \in I_n$, $\forall y \in \mathbb{R}^m$, $\forall \mu \in \mathbb{R}_+ \quad$ *if* $\quad \|x - y\| \leq e^{-\Omega(\|x\|)-\mu} \quad$ *then* $\quad y \in I$ *and* $\|f(x) - f(y)\| \leq e^{-\mu}$

Define $f_\eta^*(x, u) = f^{[n]}(x) \quad$ for $\quad x \in I_n$, $u \in [n - \eta, n + \eta]$ and $n \in \mathbb{N}$. Then, $f_\eta^* \in$ ATSP.

Finally, for converting input words of Turing machines in a desired fashion, it has been used a particularly suitable encoding that for every input word w of the alphabet provides a corresponding rational number while keeping track of the original length of the word. This is done by defining a 2-dimensional encoding that we will call Ψ. Together with this particular encoding the notion of *emulable* function completes the required knowledge to state the equivalence. This is done in the next section.

2.8 Equivalence relation between FP and ATSP

From this point on, we fix an alphabet Γ and all languages are considered over this alphabet. It is common practice to take $\Gamma = \{0, 1\}$ but every other possible choice works equally for any finite alphabet. We will assume that Γ comes with an injective mapping $\gamma : \Gamma \to \mathbb{N} \setminus \{0\}$, which means that every letter of the alphabet has a unique assigned positive number. By extension, γ applies letter wise over words.

Definition 2.26 (Discrete emulation) *Let G be a set of functions over $\mathbb{R}^2$ and let $k = 2 + \max(\gamma(\Gamma))$. The function $f : \Gamma^* \to \Gamma^*$ is called emulable under G if there exists $g \in G$ such that for any word $w \in \Gamma^*$*

$$g(\Psi_k(w)) = \Psi_k(f(w)) \text{ where } \Psi_k(w) = \left(\sum_{i=1}^{|w|} \gamma(w_i) k^{-i}, |w| \right)$$

In this case we say that g emulates f.

The theorem that establishes the equivalence at a polynomial level is then the following.

Theorem 2.27 (FP equivalence) *Let $f : \Gamma^* \to \Gamma^*$. Then $f \in$ FP if and only if f is emulable under ATSP.*

Chapter 3

Exponential analog classes

One of the goals of this thesis work is to characterize the class FEXPTIME by means of an extension of the class ATSP. The idea is to allow greater boundaries for the dynamical system but preserve for the simulation the same core structure used in the polynomial case. We define now the new analog class of functions that we will use to achieve this goal. We will refer to this class as *Analog-Time-Space-Exponential*, or ATSE.

Whenever we refer to an *exponential boundary function* $f : \mathbb{R}_+ \to \mathbb{R}_+$ we intend that there exist polynomials p, q with coefficients in $\mathbb{R}_P$ and a constant $c \in \mathbb{R}_P$ such that $\forall x \in \mathbb{R}$ we have

$$f(x) = q(x)c^{p(x)}. \tag{3.1}$$

3.1 Definitions of the exponential analog classes

Definition 3.1 (ATSE) *Let $f \subseteq \mathbb{R}^n \to \mathbb{R}^m$. Let $\Pi_1 : \mathbb{R}_+ \to \mathbb{R}_+$ and $\Upsilon_1 : \mathbb{R}_+ \to \mathbb{R}_+$ be two exponential boundary functions and let $\Pi_2 : \mathbb{R}_+ \to \mathbb{R}_+$ and $\Upsilon_2 : \mathbb{R}_+ \to \mathbb{R}_+$ be two polynomials.*

We say that $f \in$ ATSE if and only if there exist $d \in \mathbb{N}$, $p \in \mathbb{R}_P^d[\mathbb{R}^d]$ and $q \in \mathbb{R}_P^d[\mathbb{R}^n]$ such that for any $x \in \operatorname{dom} f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$

- $y(0) = q(x)$ *and* $y'(t) = p(y(t))$
- $\forall \mu \in \mathbb{R}_+$ *if* $t \geq \Pi_1(\|x\|)\Pi_2(\mu)$ *then* $\|(y_1(t), y_2(t), .., y_m(t)) - f(x)\| \leq e^{-\mu}$
- $\|y(t)\| \leq \Upsilon_1(\|x\|)\Upsilon_2(t)$

Notice that the key difference between this definition and the definition of ATSP is that in this case the dependence of the two boundaries from the $\|x\|$ variable is raised from a polynomial to an exponential boundary function, but the dependence from their second variable, respectively the precision μ and the time of the system t, is preserved as polynomial. The original definition of ATSP as in [20] was designed with the intent of characterizing polynomial complexity classes and aimed to provide the most simple description of the dynamical systems involved, therefore it made use of just one single term for both the time and the space boundaries. Indeed, at a polynomial level, having two multiplied terms with polynomial dependences or having a single one depending polynomially on two inputs provide the same results. Nevertheless, this is not true for different categories of boundaries, such as exponential functions. The main intuition behind this definition comes from noticing that when using exponential boundaries it is not natural to require exponential dependences on precision and time as well. We show this by means of the following example: assume for the sake of the argument that we defined ATSE with a single term for the boundaries Π and Υ as done for ATSP, but with Π and Υ both raised to exponential boundary functions instead of polynomials. Then given some $f \in$ ATSE, to compute $f(x)$ we would need to wait a time $t^* = \Pi(\|x\|, \mu)$ exponential in $\|x\|$ and μ to get an approximation of $f(x)$ with accuracy $e^{-\mu}$, due to the second condition of definition 3.1. Moreover, at time t^*, we would have that the norm of the solution y of the ODE computing $f(x)$ is bounded by $\Upsilon(\|x\|, t^*) = \Upsilon(\|x\|, \Pi(\|x\|, \mu))$, which is a double exponential in both $\|x\|$ and μ, while what would be natural is that $\|y(t^*)\|$ is bounded by an exponential in $\|x\|$ and μ, and not a double exponential in these parameters. Note that this problem does not happen when both Π and Υ are polynomials: since the composition of two polynomials is again a polynomial, when computing $f(x)$ with accuracy $e^{-\mu}$ we will be able to use an approximation $y(t^*)$ whose norm is bounded by a polynomial in $\|x\|$ and μ. Quite naturally, this exponential class is closely related (by means of theorem 3.6) to a class of *Exponentially-Bounded-Generable-Functions*, or GEVAL, which is defined as the class obtained by GVAL(sp) when the function sp is an exponential boundary function.

Definition 3.2 (GEVAL) *Let I be an open and connected subset of $\mathbb{R}^d$ and $f : I \to \mathbb{R}^e$. We say that $f \in$ GEVAL if and only if there exists an exponential boundary function $sp : \mathbb{R}_+ \to \mathbb{R}_+$, $n \geq e$, a $n \times d$ matrix p consisting of polynomials with coefficients in $\mathbb{R}_P$, $x_0 \in \mathbb{R}_P^d \cap I$, $y_0 \in \mathbb{R}_P^n$ and $y : I \to \mathbb{R}^n$ satisfying for all $x \in I$*

- $y(x_0) = y_0$ *and* $J_y(x) = p(y(x))$
- $f(x) = (y_1(x), y_2(x), .., y_e(x))$
- $\|y(x)\| \leq sp(\|x\|)$.

3.2 Properties of the exponential analog classes

Most of the properties of the GPVAL class are still valid when applied to GEVAL as it is possible to check by repeating the same proofs developed for the polynomial case [20]. In this way we get the following properties.

Theorem 3.3 (GEVAL closure by arithmetic operations)
If $g, f \in$ GEVAL, then $g+f$, $g-f$, $g \cdot f$ $\in$ GEVAL, with the obvious restrictions on the domains of definition.

Theorem 3.4 (GEVAL modulus of continuity) *If $f \in$ GEVAL, then f admits an exponential modulus of continuity: there exists an exponential boundary function $\Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$ such that $\forall x, y \in dom\ f$ and $\mu \in \mathbb{R}_+$*

$$\|x - y\| \leq e^{-\Omega(\|x\|, \mu)} \quad \Rightarrow \quad \|f(x) - f(y)\| \leq e^{-\mu}$$

In particular, f is continuous.

Theorem 3.5 (Exponential bound of GEVAL) *Let $f \in \mathrm{GEVAL}$, then there exists an exponential boundary function E such that $\|f(x)\| \leq E(\|x\|)$ $\forall x \in \operatorname{dom} f$.*

The way these two classes are related is the same that connects GPVAL with ATSP, where the inclusion happens for star domains by means of the following theorem.

Theorem 3.6 *If $f \in \mathrm{GEVAL}$ has a star domain with a generable vantage point, then $f \in \mathrm{ATSE}$.*

Proof of theorem 3.6. The structure of the proof is similar to the polynomial case [20] and it involves connecting two points of the domain of the function $f \in \mathrm{GEVAL}$ with a straight line parametrized by the time of the ATSE system that converges with an exponential rate to the right value of f. We present it to illustrate how results for GPVAL extend in a straightforward manner to GEVAL.

Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function in GEVAL and let $x_0 \in \mathbb{R}^n_P \cap \operatorname{dom} f$ be a generable vantage point. By definition of GEVAL we know that there exist an exponential boundary function sp, two polynomials p, q, two initial points x_0, y_0 and a solution y that satisfy definition 3.2. Moreover, since the point x_0 is a generable point, we know that $y(x_0) \in \mathbb{R}^d_P$. Let $x \in \operatorname{dom} f$ and consider the system

$$\begin{cases} x(0) = x \\ \gamma(0) = x_0 \\ z(0) = y(x_0) \end{cases} \qquad \begin{cases} x'(t) = 0 \\ \gamma'(t) = x(t) - \gamma(t) \\ z'(t) = p(z(t))\,(x(t) - \gamma(t)) \end{cases} \tag{3.2}$$

Notice that all the differential equations of (3.2) are expressed as polynomials of the variables $x(t), \gamma(t), z(t)$. It is immediate to check that the variable $x(t)$ is constant and that $\gamma(t) = x - (x_0 - x)e^{-t}$. Notice that we know that $\gamma(t) \subseteq [x_0, x] \subseteq \operatorname{dom} f$ for all t because we are in a star domain and x_0 is a vantage point. For the solution of $z(t)$ we have $z(t) = y(\gamma(t))$ since $x(t) - \gamma(t) = \gamma'(t)$ and the Jacobian of the GEVAL system defining f is exactly p. Therefore applying definition 3.2 yields $(z_1(t), z_2(t), .., z_m(t)) = f(\gamma(t))$. Thanks to the same definition we also know that y is exponentially bounded and so is $z(t)$ for every $t \in \mathbb{R}_+$, i.e. $\|z(t)\| \leq \|y(\gamma(t))\| \leq \exp(\|\gamma(t)\|) \leq \exp'(\|x\|)$ where exp and $\exp'$ are two exponential boundary functions. Therefore the space boundary required by the ATSE definition is correct, which means that condition 3 of definition 3.1 is satisfied. We can now proceed proving condition 2. Since GEVAL functions have exponential modulus of continuity, by means of theorem 3.4 there exists an exponential boundary function h such that, $\forall x_1, x_2 \in \operatorname{dom} f$ satisfying $[x_1, x_2] \subseteq \operatorname{dom} f$, we have

$$\|f(x_1) - f(x_2)\| \leq \|x_1 - x_2\|\, h(\|x_1, x_2\|)$$

and since $\|\gamma(t)\| \leq \|x_0, x\|$ we get

$$\|f(x) - z(t)\| \leq \|x - x_0\|\, e^{-t} h(\|x_0, x\|)$$

Therefore, there exist an exponential boundary function $\Pi_1 : \mathbb{R}_+ \to \mathbb{R}_+$ and a polynomial $\Pi_2 : \mathbb{R}_+ \to \mathbb{R}_+$ such that

$$\forall \mu \in \mathbb{R}_+, \forall t \geq \Pi_1(|x|)\Pi_2(\mu) \text{ we have } \|f(x) - z(t)\| \leq e^{-\mu} \tag{3.3}$$

This completes the proof for $f \in \text{ATSE}$. ■

What we have seen above shows that most of the properties of GPVAL have a clear correspondence for the exponential counterpart GEVAL. Nevertheless, when dealing with the classes defined by limit procedure, such as ATSP and ATSE, this correspondence is more delicate to obtain and the extension of the polynomial proofs is not straightforward for the exponential cases. One of the reasons for this is that some proofs of ATSP properties make use of the equivalence established in [51] between ATSP, AOP, ALP and other polynomial classes. Specifically, the features of the AOP class are particularly useful in these proofs, since this class allows functions to be applied over values that are not fixed, but can oscillate around a certain point. To see an example of the latter, the reader can check the proof of theorem 2.7 as taken from [20]. Because of the novelties of the ATSE class it is not immediately clear how to describe a new analog class with features comparable to AOP and that could be equivalent to ATSE. Hence, whether some properties of ATSP are still valid for this exponential extension is a complex question that would require new proving techniques to be answered.

Nevertheless, a crucial observation for this thesis work is to notice that the only properties we need for our goal of characterizing FEXPTIME are closure under arithmetic operations and closure under composition with ATSP functions.

Theorem 3.7 (ATSE closure by arithmetic operations) *If $g, f \in \text{ATSE}$, then $g + f,\ g - f,\ g \cdot f\ \in \text{ATSE}$, with the obvious restrictions on the domains of definition.*

Proof of theorem 3.7. We present the proof only for the closure by product, since the other cases are similar. We prove the theorem for dimension one, but the proof can be easily repeated for higher dimensions. Moreover, this proof illustrates why it is impossible to obtain an analog class of this kind that is closed under product without splitting the dependence of the boundaries in two parts, one exponential and one polynomial, as done in the ATSE definition. Let us consider a function $f \in \text{ATSE}(\Pi_1\Pi_2, \Upsilon_1\Upsilon_2)$ and a function $g \in \text{ATSE}(\Pi_1^*\Pi_2^*, \Upsilon_1^*\Upsilon_2^*)$ with parameters d, p, q and d^*, p^*, q^* respectively, where d, d^* represent the dimensions of the two dynamical systems, p, p^* the polynomials defining the right-hand terms of the differential equations and q, q^* the polynomials defining the initial conditions. Recall that, by definition of the class, we have that $\Pi_1, \Upsilon_1, \Pi_1^*, \Upsilon_1^*$ are exponential boundary functions and that $\Pi_2, \Upsilon_2, \Pi_2^*, \Upsilon_2^*$ are polynomials. Assume, without loss of generality, that all the functions used as boundaries for definitions of f and g are increasing functions. Let $x \in \operatorname{dom} f \cap \operatorname{dom} g$ and consider the following system

$$\begin{cases} y(0) = q(x) \\ y'(t) = p(y(t)) \end{cases} \quad \begin{cases} z(0) = q^*(x) \\ z'(t) = p^*(z(t)) \end{cases} \quad \begin{cases} w(0) = y_1(0)z_1(0) \\ w'(t) = p_1(y(t))z_1(t) + y_1(t)p_1^*(z(t)) \end{cases} \tag{3.4}$$

Notice that the solution of (3.4) for the variable w is: $w(t) = y_1(t)z_1(t)$ and that this system has polynomial right-hand terms. We can now proceed with the analysis of the boundaries. Since $f, g \in$ ATSE we have $\|f(x)\| \leq 1 + \Upsilon_1(\|x\|)\Upsilon_2(\Pi_1(\|x\|)\Pi_2(0))$ and $\|g(x)\| \leq 1 + \Upsilon_1^*(\|x\|)\Upsilon_2^*(\Pi_1^*(\|x\|)\Pi_2^*(0))$; denote by $l(\|x\|)$ and $l^*(\|x\|)$ these two bounds respectively. Notice that we get that $l(\|x\|)$ and $l^*(\|x\|)$ are exponential boundaries functions, and this would not have been true if $\Pi_2, \Upsilon_2, \Pi_2^*, \Upsilon_2^*$ were not polynomials. Now consider $t \geq \Pi_1(\|x\|)\Pi_2(\mu + \ln 2l^*(\|x\|))$; then $\|f(x) - y_1(t)\| \leq e^{-\mu + \ln 2\|g(x)\|}$ and in the same way if $t \geq \Pi_1^*(\|x\|)\Pi_2^*(\mu + \ln 2l(\|x\|))$ then $\|g(x) - z_1(t)\| \leq e^{-\mu + \ln 2\|f(x)\|}$. Therefore if we consider times greater than the maximum of these two bounds we have

$$\|y_1(t)z_1(t) - f(x)g(x)\| \leq \|(y_1(t) - f(x))g(x)\| + \|y_1(t)(z_1(t) - g(x))\| \leq e^{-\mu}$$

This proves condition 2 of definition 3.1. Condition 3 is verified by observing that $\|y_1(t)\| \leq l(\|x\|)$ and $\|z_1(t)\| \leq l^*(\|x\|)$ and this concludes the proof. ■

The lack of a closure by composition property is the main obstacle on a straightforward extension of the polynomial equivalence to an exponential level. As we can clearly see in the proof of theorem 2.8, if we try to consider the two boundaries Π' and Υ' to be exponential boundary functions for the function f, then the resulting solution of the composed system will not be bounded by an exponential neither in space nor time. This is the reason that originally brought us to split the dependence of the boundaries in two parts. Indeed, with this modification, which leads to the definition of ATSE, we can state the following property (which was not true for the case of exponential dependence of the boundaries on both their inputs, as it will be clear to the reader following the proof).

Theorem 3.8 (Composition of ATSE and ATSP) *Let f be a function, $f \in$ ATSE and g be a function, $g \in$ ATSP. Let $f(\operatorname{dom} f) \subseteq \operatorname{dom} g$. Then $g \circ f \in$ ATSE.*

Proof of theorem 3.8. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ and let $g : \mathbb{R}^m \to \mathbb{R}^l$. We will show that $g \circ f \in$ ATSE by using the fact that g satisfies definition 2.3. Indeed, if $g \in$ ATSP, we can apply the equivalence between ATSP and AOP to obtain that g satisfies definition 2.3; then there are polynomials Π, Υ, Λ such that definition 2.3 holds with corresponding r, δ, z_0, where r is the polynomial defining the differential equation, $\delta \in \mathbb{R}$ is the same parameter as in definition 2.3, and z_0 is the initial condition of the system. In the same way we obtain that there exist two exponential boundary functions Π_1', Υ_1', and two polynomials Π_2', Υ_2' such that $f \in \text{ATSE}(\Pi_1'\Pi_2', \Upsilon_1'\Upsilon_2')$ with corresponding parameters d, p, q for the dimension and polynomials defining the system. Assume, without loss of generality, that all the functions used as boundaries for definitions of f and g are increasing functions. Let $x \in \mathbb{R}^n$ and consider the following system

$$\begin{cases} y(0) = q(x) \\ y'(t) = p(y(t)) \end{cases} \quad \begin{cases} z(0) = z_0 \\ z'(t) = r(z(t), (y_1(t), y_2(t), ..y_m(t))) \end{cases} \quad \begin{cases} x(0) = x \\ x'(t) = 0 \end{cases}$$

Define $v(t) = (x(t), y(t), z(t))$. Then it immediately follows that v satisfies a

PIVP of the form

$$\begin{cases} v(0) = poly(x) \\ v'(t) = poly(v(t)). \end{cases} \tag{3.5}$$

We will show that the polynomial IVP (3.5) computes $g \circ f$ according to definition 3.1. First we check the space bound condition of that definition. By definition of v we have

$$\begin{aligned}
\|v(t)\| &= \max(\|x(t)\|, \|y(t)\|, \|z(t)\|) \\
&\leq \max(\|x\|, \|y(t)\|, \Upsilon(\sup_{u\in[t,t-\delta]} \|(y_1(u), y_2(u), ..y_m(u))\|, t)) \\
&\leq \exp(\|x\|)\ \text{poly}(\sup_{u\in[t,t-\delta]} \|y(u)\|, t) \\
&\leq \exp(\|x\|)\ \text{poly}(\sup_{u\in[t,t-\delta]} \Upsilon_1'(\|x\|)\Upsilon_2'(u), t) \\
&\leq \exp(\|x\|)\ \text{poly}(t)
\end{aligned}$$

where the notation poly and exp indicate a polynomial and a exponential boundary function, respectively. This proves the space bound.

Let us now tackle the time bound of definition 3.1. Define $\bar{x} = f(x)$, $\Upsilon^*(\alpha) = \Upsilon_1'(\alpha)\Upsilon_2'(\Pi_1'(\alpha)\Pi_2'(0))+1$ and $\Pi''(\alpha, \mu) = \Pi_1'(\alpha)\Pi_2'(\Lambda(\Upsilon^*(\alpha), \mu))+\Pi(\Upsilon^*(\alpha), \mu)$. Notice that Υ^* is an exponential boundary function, and notice that there always exist two functions, Π_1^* and Π_2^*, such that $0 \leq \Pi''(\alpha, \mu) \leq \Pi_1^*(\alpha)\Pi_2^*(\mu)$, where Π_1^* is an exponential boundary function and Π_2^* is a polynomial. From the fact that $f \in \text{ATSE}(\Pi_1'\Pi_2', \Upsilon_1'\Upsilon_2')$, by using time $t^* = \Pi_1'(\|x\|)\Pi_2'(0)$, we conclude that $\|(y_1(t^*), y_2(t^*), ..y_m(t^*)) - f(x)\| \leq e^{-0}$, which implies that

$$\begin{aligned}
\|\bar{x}\| &= \|f(x)\| \\
&\leq \|y(t^*)\| + 1 \\
&\leq \Upsilon_1'(\|x\|)\Upsilon_2'(t^*) + 1 \\
&= \Upsilon_1'(\|x\|)\Upsilon_2'(\Pi_1'(\|x\|)\Pi_2'(0)) + 1 \\
&= \Upsilon^*(\|x\|).
\end{aligned}$$

Let $\mu > 0$. By definition of ATSE, if $t \geq \Pi_1'(\|x\|)\Pi_2'(\Lambda(\Upsilon^*(\|x\|), \mu))$ then $\|(y_1(t), y_2(t), .., y_m(t)) - \bar{x}\| \leq e^{-\Lambda(\Upsilon^*(\|x\|),\mu))} \leq e^{-\Lambda(\|x\|,\mu)}$. Therefore, due to definition 2.3 , we have that

$$t \geq \Pi_1'(\|x\|)\Pi_2'(\Lambda(\Upsilon^*(\|x\|), \mu)) + \Pi(\|\bar{x}\|, \mu)$$

implies that $\|(v_1(t), v_2(t), .., v_l(t)) - g(f(x))\| \leq e^{-\mu}$. At this point note that $\Pi_1'(\|x\|)\Pi_2'(\Lambda(\Upsilon^*(\|x\|), \mu)) + \Pi(\|\bar{x}\|, \mu)$ depends exponentially on $\|x\|$, but only polynomially on μ. This shows the time bound for ATSE. ■

3.3 Equivalence relation between FEXPTIME and ATSE

Finally, we can state the main result of this chapter. Indeed, slightly modifying the construction already developed in [20], we can show the equivalence between the class ATSE and the class FEXPTIME of functions computable in exponential time. This equivalence is described by the following theorem.

Theorem 3.9 (FEXPTIME equivalence) *Let $f : \Gamma^* \to \Gamma^*$. Then $f \in$ FEXPTIME if and only if is emulable under ATSE.*

The proof of this theorem will follow in the next section of this thesis work.

3.4 Proof of the FEXPTIME analog characterization

We start this section by proving theorem 3.9. We will proceed with the direct and reverse direction of the theorem separately.

For the direct direction of theorem 3.9 we will need once again to be able to iterate the step function with the help of theorem 2.25. The only difference to the polynomial case will be the number of iterations of the step function required to simulate a Turing machine computing a function in FEXPTIME. The latter is obviously connected with the maximum amount of steps that such a machine can perform before halting: while this amount was polynomial in the original case analyzed in [20], now it is in principle an exponential on the size of the input. Nevertheless, an exponential version of theorem 2.25 is not necessary, and the original polynomial version it is enough for our goal. Applying theorem 2.25 to our case of interest, the real step function defined in theorem 2.24 will play the role of the f function described in the statement above. This particular choice will automatically satisfy the third condition of the theorem, as a direct consequence of the fact that the real step belongs to ATSP together with the property of ATSP functions of having polynomial modulus of continuity given by theorem 2.7.

For what concerns the first condition of the theorem, the construction can be maintained identical to the one described in [20], and a proper description will be given later in this chapter. As real step function to iterate we will choose $h(\langle c\rangle) = \langle M\rangle(\langle c\rangle, \mu)$ for $\langle c\rangle \in \mathbb{R}^4$ and for a fixed value of μ. Even if this function will need to be iterated an exponential number of times to be able to represent the transition function of a Turing machine that computes functions in FEXPTIME, the second condition of theorem 2.25 will still be valid, due to the particular choice of the family of intervals I_n and the precision μ. Indeed, it is possible to ensure that $\left\| h^{[n]}(\langle \bar{c}\rangle) - h^{[n]}(\langle c\rangle)\right\| \leq k^n \left\|\langle c\rangle - \langle \bar{c}\rangle\right\| \leq \frac{1}{4k^2}$ for every $\langle \bar{c}\rangle \in I_n$, for every encoded configuration $\langle c\rangle$ of M and for every $n \in \mathbb{N}$. Recall now that, for all $n \in \mathbb{N}$ and for every configuration c of M, we have from theorem 2.24: $\langle M^{[n]}(c)\rangle = h^{[n]}(\langle c\rangle)$. Therefore, since the encoding of any configuration $\langle c\rangle$ is never greater than a fixed real number (recall that $0.\gamma(w) \leq 1$ for all $w \in \Gamma^*$, as it follows from the definition of γ together with definition 2.21), we obtain that the second condition of the theorem is always satisfied. We can now proceed showing in more detail how the construction of the simulation is made.

Proof of theorem 3.9. Let $f \in$ FEXPTIME. Then there exists a Turing machine $M = (Q, \Sigma, b = 0, \delta, q_0, F)$ where $\Sigma = \{0, ..., k-2\}$ and $\gamma(\Gamma) \subseteq \Sigma \setminus \{b\}$ and an exponential boundary function $e_M(|w|) \equiv K^{|w|} \in \mathbb{N}$ for some constant $K \in \mathbb{N}$ such that for any word $w \in \Gamma^*$ M halts in at most $e_M(|w|)$ steps, that is $M^{[e_M(|w|)]}(c_0(\gamma(w))) = c_\infty(\gamma(f(w)))$. We assume that $\Psi_k(w) = (0.\gamma(w), |w|)$ for any word $w \in \Gamma^*$. Define $\mu = \ln(4k^2)$ and $h(c) = \langle M\rangle(c, \mu)$ $\forall c \in \mathbb{R}^4$. Define

$I_\infty = \langle C_M \rangle$ and $I_n = I_\infty + [\epsilon_n, \epsilon_n]^4$ where $\epsilon_n = \frac{1}{4k^{2+n}}$ $\forall n \in \mathbb{N}$. Note that $\epsilon_{n+1} \leq \frac{\epsilon_n}{k}$ and that $\epsilon_0 \leq \frac{1}{2k^2} - e^{-\mu}$. By definition of the real step, and because of theorem 2.24, we have that the real step function of this machine, h, satisfies $h \in$ ATSP and $h(I_{n+1}) \subseteq I_n$. In particular $\left\|h^{[n]}(\bar{c}) - h^{[n]}(c)\right\| \leq k^n \|c - \bar{c}\|$ $\forall c \in I_\infty$ and $\bar{c} \in I_n$, $\forall n \in \mathbb{N}$. Let $\delta \in [0, \frac{1}{2}[$ and define $J = \bigcup_{n\in\mathbb{N}} I_n \times [n-\delta, n+\delta]$. Apply then theorem 2.25 to the function h to get $h^*_\delta : J \to I_0 \in$ ATSP such that for all $c \in I_\infty$ and $n \in \mathbb{N}$ we have $h^*_\delta(c, n) = h^{[n]}(c)$. Let P_i denote the i-th projection, that is, $P_i(x) = x_i$; then $P_i \in$ ATSP. Define

$$g(y, l) = P_3(h^*_\delta(0, b, y, q_0, e_M(l))) \text{ for } (y, l) \in \Psi_k(\Gamma^*)$$

and notice that g is well defined. Indeed, if $(y, l) = \Psi_k(w)$ for some $w \in \Gamma^*$, then $y = 0.\gamma(w)$, $l = |w|$, and $(0, b, y, q_0) = \langle(\lambda, b, \gamma(w), q_0)\rangle = \langle c_0(\gamma(w))\rangle \in I_\infty$. Therefore, by construction, for any word $w \in \Gamma^*$ we have

$$\begin{aligned} g(\Psi_k(w)) &= P_3(h^*_\delta(\langle c_0(\gamma(w))\rangle, e_M(|w|))) \\ &= P_3(h^{[e_M(|w|)]}\langle c_0(\gamma(w))\rangle) \\ &= P_3(\langle C_M^{[e_M(|w|)]}(c_0(\gamma(w)))\rangle) \\ &= P_3(\langle c_\infty(\gamma(f(w)))\rangle) \\ &= 0.\gamma(f(w)) = P_1(\Psi_k(f(w))) \end{aligned} \tag{3.6}$$

Recall that to show the validity of the emulation we need to compute $\Psi_k(f(w))$ and so far we only have the first component, the output of the tape encoding, but we miss the second component: its length. To complete the task, we can apply a useful function defined in [20] that allows to extract the length of its input.

Theorem 3.10 (Length recovery [20]) *For any machine M there exists a function $tlength_M : P_3(\langle C_M \rangle) \times \mathbb{N} \to \mathbb{N} \in$ ATSP such that for any word $w \in (\Sigma \setminus \{b\})^*$ and any $n \geq |w|$, $tlength_M(0.w, n) = |w|$.*

Then we can apply this theorem to get

$$tlength_M(g(\Psi_k(w)), |w| + e_M(|w|)) = |\gamma(f(w))| = |f(w)|$$

since $\gamma(f(w))$ does not contain any blank character by definition of γ and $|\gamma(f(w))| \leq |w| + e_M(|w|)$. Therefore, if we define the function $\bar{g}$ as

$$\bar{g}(\Psi_k(w)) \equiv (g(\Psi_k(w)), tlength_M(g(\Psi_k(w)), |w| + e_M(|w|))) \tag{3.7}$$

We have just showed that $\bar{g}$ emulates f as described by definition 2.26. To conclude the direct direction of the theorem, the last result to prove is that $\bar{g} \in$ ATSE. Recall the definition of g, $g(\Psi_k(w)) = P_3(h^*_\delta(\langle c_0(\gamma(w))\rangle, e_M(|w|)))$, and notice that because of theorem 2.25, $h^*_\delta \in$ ATSP. It is trivial to show that $e_M \in$ ATSE. Then, due to the composition theorem 3.8 we obtain that $h^*_\delta(\langle c_0(\gamma(w))\rangle, e_M(|w|)) \in$ ATSE. By definition of $\bar{g}$ in (3.7) and because we know that $tlength, P_3 \in$ ATSP and $h^*_\delta(\langle c_0(\gamma(w))\rangle, e_M(|w|))$, $e_M \in$ ATSE we can apply again the composition theorem 3.8 to state that $\bar{g} \in$ ATSE and conclude the direct direction of the proof of theorem 3.9.

We now proceed with the reverse direction of the proof. This direction is similar to the one already provided by [20] for the ATSP equivalence, where the difference here is the nature of the boundaries considered. To be able to prove this result, it is necessary to introduce a theorem related to the complexity of solving polynomial differential equations. The proof and more details about the theorem can be found in [36]. We need to introduce new notation to state the result. For any multivariate polynomial p, where $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{N}^m$, $|\alpha| = \alpha_1 + \ldots + \alpha_m$, and $x^\alpha = x_1^{\alpha_1} \cdots x_m^{\alpha_m}$, we call k the degree, $k = deg(p)$, if k is the minimal integer for which the condition $p(x) = \sum_{|\alpha| \le k} a_\alpha x^\alpha$ holds, and we denote the sum of the norm of the coefficients by $\Sigma p = \sum_{|\alpha| \le k} |a_\alpha|$ (also known as the length of p). For a vector of polynomials, we define the degree and Σp as the maximum over all components. For any continuous y and polynomial p, define the *pseudo-length* as

$$\mathrm{PsLen}_{y,p}(a,b) = \int_a^b \Sigma p \; \max(1, \|y(u)\|)^{deg(p)} du \tag{3.8}$$

Then the necessary theorem is the following.

Theorem 3.11 (Computing values of ODEs solutions) *Let $I = [a,b]$ be an interval, $p \in \mathbb{R}^n[\mathbb{R}^n]$, k its degree and $y_0 \in \mathbb{R}^n$. Assume that $y : I \to \mathbb{R}^n$ satisfies $\forall t \in I$*

$$y(a) = y_0 \ , \ y'(t) = p(y(t)) \tag{3.9}$$

Then $y(b)$ can be computed with precision $2^{-\mu}$ in time bounded by

$$\mathrm{poly}(k, \mathrm{PsLen}_{y,p}(a,b), \log \|y_0\|, \mu)^n \tag{3.10}$$

More precisely, there exists a Turing machine M such that for any oracle O representing (a, y_0, p, b) and any $\mu \in \mathbb{N}$ we have: $\left\|M^O(\mu) - y(b)\right\| \le 2^{-\mu}$ where y is the solution of (3.9) and the number of steps of the machine is bounded by the above expression in (3.10).

Now we can continue with the proof. Assume that f is emulable under ATSE. Apply then the definition of emulable function to get $g \in \mathrm{ATSE}(\Pi_1\Pi_2, \Upsilon_1\Upsilon_2)$ with respective d, p, q. Recall that, according with definition 3.1, the parameters d, p and q represent respectively the dimension, the polynomial right-hand term and the polynomial initial condition of the ATSE dynamical system relative to g. Recall also that Υ_1, Π_1 are exponential boundary functions and Υ_2, Π_2 are polynomials. Let $w \in \Gamma^*$; we will now describe an FEXPTIME algorithm to compute $f(w)$. Consider the following system

$$y(0) = q(\Psi_k(w)) \ , \ y'(t) = p(y(t)) \tag{3.11}$$

note that the coefficients of p, q and $q(\Psi_k(w))$ are polynomially computable in the sense of computable analysis. The algorithm works in two steps: first, we compute a rough approximation of the output to be able to guess its length. Then we rerun the system with enough precision to get the full output.

Let $t_w = \Pi_1(|w|)\Pi_2(2)$ for any $w \in \Gamma^*$. Note that t_w is computable and that it is exponentially bounded in $|w|$ because Π_1 is an exponential boundary function. Apply now theorem 3.11 to (3.11) to compute $\bar{y}$ such that $\|\bar{y} - y(t_w)\| \le e^{-2}$: this operation takes a computational time that is exponential in $|w|$ because t_w is exponentially bounded, $\log \|q(\Psi_k(w))\|$ is polynomial in $|w|$ and because $\mathrm{PsLen}_{y,p}(0, t_w) \le \mathrm{poly}(t_w, \sup_{[0,t_w]} \|y(t)\|)$ and, by construction, $\|y(t)\| \le$

$\Upsilon_1(\|\Psi_k(w)\|)\Upsilon_2(t_w)$ for $t \in [0, t_w]$ where Υ_1 is an exponential boundary function and Υ_2 is a polynomial. Furthermore, by definition of t_w: $\|y(t_w) - g(\Psi_k(w))\| \leq e^{-2}$ thus $\|\bar{y} - \Psi_k(f(w))\| \leq 2e^{-2} \leq \frac{1}{3}$. But, since $\Psi_k(f(w)) = (0.\gamma(f(w)), |f(w)|)$, from $\bar{y}_2$ we can find $|f(w)|$ by rounding to the closest integer. In other words we can compute $|f(w)|$ in exponential time in $|w|$. Note that this implies that $|f(w)|$ is at most exponential in $|w|$.

Let $t'_w = \Pi_1(|w|)\Pi_2(2+|f(w)|\ln k)$. Notice that there always exists an exponential boundary function on $|w|$ such that it is greater than t'_w. Indeed, Π_1 is an exponential boundary function, Π_2 is a polynomial, and $|f(w)|$ is at most exponential in $|w|$. We can then apply the same reasoning and use again theorem 3.11 to get $\tilde{y}$ such that $\|\tilde{y} - y(t'_w)\| \leq e^{-2-|f(w)|\ln k}$. Once again this operation takes a time exponential in $|w|$. Furthermore, $\|\tilde{y}_1 - 0.\gamma(f(w))\| \leq 2e^{-2-|f(w)|\ln k} \leq \frac{1}{3}k^{-|f(w)|}$. We claim that this allows us to recover $f(w)$ unambiguously in exponential time in $|w|$. Indeed, it implies that $\left\|k^{|f(w)|}\tilde{y}_1 - k^{|f(w)|}0.\gamma(f(w))\right\| \leq \frac{1}{3}$. At this point, if we unfold the definition of the expression $0.\gamma(f(w))$, we are able to notice that $k^{|f(w)|}0.\gamma(f(w)) = \sum_{i=1}^{|f(w)|}\gamma(f(w)_i)k^{|f(w)|-i} \in \mathbb{N}$; thus by rounding $k^{|f(w)|}\tilde{y}_1$ to the closest integer we recover $\gamma(f(w))$ and then $f(w)$. This is all done in polynomial time in $|f(w)|$, and so in exponential time in $|w|$. This completes the proof of theorem 3.9. ■

Chapter 4

Characterization of FPSPACE

Studying the construction developed until now to prove theorem 3.9 it is natural to wonder whether a similar procedure could be applied in the attempt of continuously characterize classes of standard complexity theory defined by means of space boundaries, such as FPSPACE or PSPACE. Indeed, since we know that FPSPACE $\subseteq$ FEXPTIME it makes sense to search for a specific subset of the ATSE functions that could capture in its definition the essential features of polynomially space bounded Turing machine computation. In order to do so, the first element to notice from the analog simulation described in the previous chapters is that the key property that we are exploiting is the total number of steps performed on each input word by the Turing machine M that we want to simulate. This is the only necessary information that we require to know to be able to establish the simulation. Provided that this total number of steps can be generated by an ODE with the correct boundaries, then we are capable to simulate the machine M for the amount of steps we want. Nevertheless, what we know about FPSPACE computations is that they share the same maximum number of steps with FEXPTIME computations, since a polynomially space bounded machine can move its head for a number of times that is an exponential on the size of the input. Therefore, using only the maximum number of steps as a parameter, our construction would produce the same results it produced for the ATSE case when trying to simulate functions in the FPSPACE class. Moreover, since the content of the tape of the simulated machine is encoded in a number between 0 and 1 during the course of the simulation, every information about its length is lost in the process, and the encoding Ψ_k needs to make use of a second component in order to keep track of this element. However, for computations related to space complexity classes, where the length of the working tape is a crucial variable, having this information encoded in a separate component is not practical. Therefore, we need to introduce an extra condition that allows us to preserve a more practical connection with the information concerning the length of the tape of the machine M. One way to do that is to discard the encoding presented in definition 2.26 and introduce another encoding that does not need to have two components to encode a discrete word together with its length. The encoding that we are going to use in this chapter is not new to the scientific community, since it has already been considered in [66], [18]. Nevertheless in these papers the authors focused their interests on computability, while here the encoding is introduced with the clear goal of describing space complexity classes. Let us define the encoding with the symbol $\hat{\Psi}$ as follows [66], [18].

Definition 4.1 ($\hat{\Psi}$) *We say that $\hat{\Psi}(x) = w$ if $x \in \mathbb{R}$ is such that $d(x, \sum_{i=1}^{|w|} w_i k^{i-1}) < 1/2$, where $d(x, y) = \|x - y\|$.*

Note the encoding defined in definition 2.26 and used in the previous chapters could be described as follows.

Definition 4.2 ($\bar{\Psi}$) *We say that $\bar{\Psi}\,(x, y) = w$ if $(x, y) \in \mathbb{R}^2$ is such that $(x, y) = \left(\sum_{i=1}^{|w|} w_i k^{-i}, |w|\right)$.*

Technically both encodings depend on the choice of the integer k but, for readability reasons, we omit the index k when referring to $\hat{\Psi}$ and to $\bar{\Psi}$. The encoding $\bar{\Psi}$ has the advantage of encoding w into a real vector which norm is equal to $|w|$, being in this sense a more efficient encoding than $\hat{\Psi}$. On the other hand, two real values are needed to encode a word w via $\bar{\Psi}$. Each encoding $\hat{\Psi}$ and $\bar{\Psi}$ has its own strengths and apparent limitations, and each one is well-suitable for at least one purpose as we will see in the next section. Usually there is a trade-off between size and required accuracy of the encoding of a word: we can be less demanding on the accuracy at the cost of increasing the size of the norm of the encoding and vice versa.

As we just mentioned above, the encoding $\hat{\Psi}$ is more suitable for the purpose of presenting a characterization of PSPACE and FPSPACE in terms of polynomial ODEs, and this will be more evident during the course of this chapter.

Another important concept related to encodings is the size of an encoded word. Note that there exist more or less efficient encodings of words in terms of the size of the norm of the encoded word. For example, considering the encodings $\hat{\Psi}$ and $\bar{\Psi}$ used in [66] and [20] respectively, we expect that if $\hat{\Psi}$ and $\bar{\Psi}$ encode the same word of Γ^*, i.e. $\hat{\Psi}(x) = \bar{\Psi}\,(x', y')$, then $\|x\|$ will be exponentially larger than $\|(x', y')\|$. This has to be reflected when using bounded continuous systems to compute discrete functions.

Definition 4.3 (Encoding bounds) *Let $\Psi : D \subseteq \mathbb{R}^k \to \Gamma^*$ be an encoding and let $g : \mathbb{N} \to \mathbb{N}$. We say that a function $\phi : \mathbb{R} \to \mathbb{R}$ is a Ψ-bound if for all $v, w \in \Gamma^*$, $|v| \leq |w|$ implies that $\|x\| \leq \phi(\|y\|)$ whenever $\Psi(x) = v$ and $\Psi(y) = w$.*

For example, when considering the encoding $\bar{\Psi}$ we may take as a $\bar{\Psi}$-bound the function $\bar{\phi}(x) = x$, and for the encoding $\hat{\Psi}$ we may take $\phi(x) = 2^{x+1}$.

Once we change the encoding that is used to encode configurations of Turing machines into real numbers, we also have to change accordingly the notion of emulation of definition 2.26 and the way the continuous simulation of Turing machine computations is performed to prove the direct direction of the characterization. Indeed, the way the simulation is conducted in this chapter is by means of an iteration technique inspired by the formulation of M.S. Branicky in [78] and later refined in [79], [66], that allows to define a dynamical systems of

ODEs able to iterate a specific given function f over the course of its evolution. In the following section we formally introduce this technique, modifying it in a way that can produce the desired boundaries over the solution of the dynamical systems involved.

4.1 Reaching a value with ODEs

The goal of this section is to perform a basic construction with an ODE, which is to approximate a value $b \in \mathbb{R}$ in a finite (given) amount of time with some (given) accuracy. This will be essential to perform the simulation of Turing machines with ODEs, which is needed to provide a characterization of PSPACE. In particular, we will study the following basic equation

$$y' = c(b-y)^3\phi(t). \tag{4.1}$$

As mentioned above, this ODE was already studied in [78], [79], [66] but here we present an extended version which provides bounds on $y(t)$ for $t \geq t_0$ (previously those were only established at a time instant t_1), since those bounds are needed to establish bounds on the values that the solution of an ODE simulating a Turing machine can have.

Lemma 4.4 *Consider a point $b \in \mathbb{R}$ (the* target*), some $\gamma > 0$ (the* targeting error*), time instants t_0 (*departure time*) and t_1 (*arrival time*), with $t_1 > t_0$, and a function $\phi : \mathbb{R} \to \mathbb{R}$ with the property that $\phi(t) \geq 0$ for all $t \geq t_0$ and $\int_{t_0}^{t_1} \phi(t)dt > 0$. Then the IVP defined by (4.1) (the* targeting equation*) with the initial condition $y(t_0) = y_0$ and*

$$c \geq \frac{1}{2\gamma^2 \int_{t_0}^{t_1} \phi(t)dt} \tag{4.2}$$

has the following properties

1. *$|y(t) - b| < \gamma$ for $t \geq t_1$, independently of the initial condition $y_0 \in \mathbb{R}$*
2. *$\min(y_0, b) \leq y(t) \leq \max(y_0, b)$ for all $t \geq t_0$.*

Proof of lemma 4.4. For the first condition of the lemma, there are two cases to consider: (i) $y(t_0) = b$ and (ii) $y(t_0) \neq b$. In the first case, the solution is given by $y(t) = b$ for all $t \in \mathbb{R}$ and the lemma is trivially true. For the second case, note that (4.1) is a separable equation, which gives

$$\frac{1}{(b-y(t_1))^2} - \frac{1}{(b-y(t_0))^2} = 2c\int_{t_0}^{t_1} \phi(t)dt \implies$$
$$\frac{1}{2c\int_{t_0}^{t_1} \phi(t)dt} > (b-y(t_1))^2.$$

Hence, we get that $|y(t_1) - b| < \gamma$ if c satisfies (4.2). This yields condition 1 of the lemma. For condition 2, note that $y = b$ yields a fixed point for (4.1), so once the solution of (4.1) reaches b, it cannot leave this point, which trivially proves condition 2 when $b = y_0$. If $b > y_0$, then $y'(t) \geq 0$ for all $t \geq t_0$ and, in particular, $y(t) \in [y_0, b]$ for all $t \geq t_0$. A similar analysis applies when $b < y_0$. ■

We now extend the previous lemma for the case of "perturbed" targeting equations

Lemma 4.5 *Consider a point $b \in \mathbb{R}$ (the* target*), some $\gamma > 0$ (the* targeting error*), time instants t_0 (*departure time*) and t_1 (*arrival time*), with $t_1 > t_0$, and a function $\phi : \mathbb{R} \to \mathbb{R}$ with the property that $\phi(t) \geq 0$ for all $t \geq t_0$ and $\int_{t_0}^{t_1} \phi(t)dt > 0$. Let $\rho, \delta \geq 0$ and let $\bar{b}, E : \mathbb{R} \to \mathbb{R}$ be functions with the property that $\left|\bar{b}(t) - b\right| \leq \rho$ and $|E(t)| \leq \delta$ for all $t \geq t_0$. Then the IVP defined by*

$$z' = c(\bar{b}(t) - z)^3 \phi(t) + E(t) \tag{4.3}$$

with the initial condition $z(t_0) = \bar{z}_0$, where c satisfies (4.2) and $\gamma > 0$ is the targeting error, has the following properties

1. $|z(t_1) - b| < \rho + \gamma + \delta(t_1 - t_0)$*, independently of the initial condition* $\bar{z}_0 \in \mathbb{R}$
2. $\min(\bar{z}_0, b - \rho) - \delta(t_1 - t_0) \leq z(t) \leq \max(\bar{z}_0, b + \rho) + \delta(t_1 - t_0)$ *for all* $t \in [t_0, t_1]$.

Proof of lemma 4.5. Without loss of generality, we take the departure time to be $t_0 = 0$ and the arrival time to be $t_1 = 1/2$ (these specific values will be used later on). Let $\bar{z}$ be the solution of the IVP (4.3), with initial condition $\bar{z}(0) = \bar{z}_0$ and let z_+, z_- be the solutions of $z' = c(b + \rho - z)^3 \phi(t) + \delta$ and $z' = c(b - \rho - z)^3 \phi(t) - \delta$, respectively, with initial conditions $z_+(0) = z_-(0) = \bar{z}_0$. For simplicity denote

$$\begin{aligned} f(t, z) &= c(\bar{b}(t) - z)^3 \phi(t) + E(t) \qquad (4.4) \\ f_+(t, z) &= c(b + \rho - z)^3 \phi(t) + \delta \\ f_-(t, z) &= c(b - \rho - z)^3 \phi(t) - \delta. \end{aligned}$$

We have that for all $(t, x) \in \mathbb{R}^2$

$$f_-(t, x) \leq f(t, x) \leq f_+(t, x). \tag{4.5}$$

Since $\bar{z}$ is the solution of the ODE $z' = f(t, z)$ and $z_\pm$ are the solutions of the ODEs $z' = f_\pm(t, z)$, all with the same initial condition $\bar{z}(0) = z_\pm(0) = \bar{z}_0$, from (4.5) and a standard differential inequality from the basic theory of ODEs (see e.g. [80, Appendix T]), it follows that $z_-(t) \leq \bar{z}(t) \leq z_+(t)$ for all $t \in \mathbb{R}$. Now, if we put upper and lower bounds on z_+ and z_-, respectively, we get immediately bounds for $\bar{z}$.

Let us study what happens with z_+. For convenience, let $y_\pm$ be the solution of

$$y' = c(b \pm \rho - y)^3 \phi(t) \tag{4.6}$$

(i.e. $y'_\pm(t) = f_\pm(t, y_\pm) \mp \delta$), with initial condition $y_\pm(0) = \bar{z}_0$. Since $f_+(t, x) > f_+(t, x) - \delta$ and $f_-(t, x) < f_-(t, x) + \delta$ for all $(t, x) \in \mathbb{R}^2$, we have similarly to the case of z_-, $\bar{z}$, and z_+, that

$$z_-(t) \leq y_-(t) \leq y_+(t) \leq z_+(t) \quad \text{for all } t \geq 0. \tag{4.7}$$

We consider two cases for z_+:

(i) $\overline{z}_0 \leq b+\rho$. Since y_+ is the solution of a targeting equation (4.1), we have from lemma 4.4 and (4.7) that

$$b+\rho-\gamma < y_+(t) < b+\rho \quad \Longrightarrow \quad b+\rho-\gamma < z_+(t)$$

for all $t \geq 1/2$. Moreover, from lemma 4.4 we also conclude that $y_+(t) \in [\overline{z}_0, b+\rho]$ for all $t \geq 0$. Note also that (4.7) implies that $c(b+\rho-y_+)^3\phi(t)+\delta > c(b+\rho-z_+)^3\phi(t)+\delta$ which together with (4.6) implies that

$$\delta > \left|z_+'(t) - y_+'(t)\right| \quad \text{for all } t \geq 0.$$

Integrating the last equation, we get that $\delta/2 > |z_+(t) - y_+(t)|$ for all $t \in [0, 1/2]$, which by its turn together with (4.7) implies that $z_+(t) \in [\overline{z}_0, b+\rho+\delta/2]$ for all $t \in [0,1/2]$. Moreover, because $b+\rho-\gamma < y_+(1/2) < b+\rho$ due to lemma 4.4 and due to (4.7), we also conclude that $b+\rho-\gamma < z_+(1/2) < b+\rho+\delta/2$.

(ii) $\overline{z}_0 > b+\rho$. Since y_+ is solution of the targeting equation (4.6), by lemma 4.4 we conclude that $y_+(t) \in [b+\rho, \overline{z}_0]$ for all $t \geq 0$ and $b+\rho < y_+(1/2) < b+\rho+\gamma$. Proceeding similarly as in the previous case, we get that $\delta/2 > |z_+(t) - y_+(t)|$ for all $t \in [0,1/2]$, which together with (4.7) implies that $z_+(t) \in [b+\rho, \overline{z}_0+\delta/2]$ for all $t \in [0,1/2]$. Moreover, because $b+\rho < y_+(1/2) < b+\rho+\gamma$ and due to (4.7), we also conclude that $b+\rho < z_+(1/2) < b+\rho+\gamma+\delta/2$.

We also consider two cases for z_-:

(i)' $\overline{z}_0 \leq b-\rho$. Since y_- is the solution of a targeting equation (4.1), we have from lemma 4.4 and (4.7) that

$$b-\rho-\gamma < y_-(t) < b-\rho \quad \Longrightarrow \quad z_-(t) < b-\rho$$

for all $t \geq 1/2$. Moreover, from lemma 4.4 we also conclude that $y_-(t) \in [\overline{z}_0, b-\rho]$ for all $t \geq 0$. Note also that (4.7) implies that $c(b+\rho-y_-)^3\phi(t)-\delta < c(b+\rho-z_-)^3\phi(t)-\delta$ which together with (4.6) implies that

$$\delta > \left|z_-'(t) - y_-'(t)\right| \quad \text{for all } t \geq 0.$$

Integrating the last equation, we get that $\delta/2 > |z_-(t) - y_-(t)|$ for all $t \in [0,1/2]$, which by its turn together with (4.7) implies that $z_-(t) \in [\overline{z}_0-\delta/2, b-\rho]$ for all $t \in [0,1/2]$. Moreover, because $b-\rho-\gamma < y_-(1/2) < b-\rho$ due to lemma 4.4 and (4.7), we also conclude that $b-\rho-\gamma-\delta/2 < z_-(1/2) < b-\rho$.

(ii)' $\overline{z}_0 > b-\rho$. Since y_- is solution of the targeting equation (4.6), by lemma 4.4 we conclude that $y_-(t) \in [b-\rho, \overline{z}_0]$ for all $t \geq 0$ and $b-\rho < y_-(1/2) < b-\rho+\gamma$. Proceeding similarly as in the previous case, we get that $\delta/2 > |z_-(t) - y_-(t)|$ for all $t \in [0,1/2]$, which together with (4.7) implies that $z_-(t) \in [b-\rho-\delta/2, \overline{z}_0]$ for all $t \in [0,1/2]$. Moreover, because $b-\rho < y_-(1/2) < b-\rho+\gamma$, we also conclude using (4.7) that $b-\rho-\delta/2 < z_-(1/2) < b-\rho+\gamma$.

Combining the previous cases we have the following conclusion. If

1. $\overline{z}_0 \leq b-\rho$, then by cases (i) and (i)', we conclude that $\overline{z}_0 - \delta/2 \leq z_-(t) \leq \overline{z}(t) \leq z_+(t) \leq b+\rho+\delta/2$ for all $t \in [0,1/2]$ and that $b-\rho-\gamma-\delta/2 < z_-(1/2) \leq \overline{z}(1/2) \leq z_+(1/2) < b+\rho+\delta/2$;

2. $b-\rho < \overline{z}_0 < b+\rho$, then by (i) and (ii)", we conclude that $b-\rho-\delta/2 \leq z_-(t) \leq \overline{z}(t) \leq z_+(t) \leq b+\rho+\delta/2$ for all $t \in [0,1/2]$ and that $b-\rho-\delta/2 < z_-(1/2) \leq \overline{z}(1/2) \leq z_+(1/2) < b+\rho+\delta/2$;

3. $b+\rho \leq \overline{z}_0$, then by (ii) and (ii)", we conclude that $b-\rho-\delta/2 \leq z_-(t) \leq \overline{z}(t) \leq z_+(t) \leq \overline{z}_0+\delta/2$ for all $t \in [0,1/2]$ and that $b-\rho-\delta/2 < z_-(1/2) \leq \overline{z}(1/2) \leq z_+(1/2) < b+\rho+\gamma+\delta/2$.

Combining these 3 cases, we conclude that

$$b-\rho-\gamma-\delta/2 < \overline{z}(1/2) < b+\rho+\gamma+\delta/2$$

which implies condition 1 of the lemma and also

$$\min(\overline{z}_0, b-\rho) - \delta/2 \leq \overline{z}(t) \leq \max(\overline{z}_0, b+\rho) + \delta/2$$

which is condition 2. ■

4.2 Simulation of Turing machines with encoding $\hat{\Psi}$

As we already mentioned, since we are using a different encoding with respect to what it was done in section 2.3, we need a different way to continuously simulate Turing machines accordingly. In this section we will slightly change also the notation previously used in section 2.3, in order to make it suit better with this new context. Following what was done in [66], let M be a one-tape Turing machine computing some function $f : \Gamma^* \to \Gamma^*$ and let Σ be the alphabet of M, with $\Gamma \subseteq \Sigma$, $B \in \Sigma$ where B is the blank symbol, and $B \notin \Gamma$. For simplicity, we assume that $\Gamma = \{1, 2, \ldots, k-1\}$ and $\Sigma = \{0, 1, \ldots, k-1\}$, which implies that we assume $B = 0$. Let

$$...B\,B\,B\,a_{-p}\,a_{-p+1}...\,a_{-1}\,a_0 a_1...\,a_n\,B\,B\,B... \tag{4.8}$$

represent the tape contents of the Turing machine M at a given time. We assume that the head is reading symbol a_0. Assume also that M has m states, represented by numbers 1 to m. We also assume that if M reaches an halting configuration, then it moves to the same configuration. We consider that, in each transition, the head either moves to the left, moves to the right, or does not move. Take

$$\begin{array}{l} y_1 = a_0 + a_1 k + \ldots + a_n k^n \\ y_2 = a_{-1} + a_{-2} k + \ldots + a_{-p} k^{p-1} \end{array} \tag{4.9}$$

which implies that $\hat{\Psi}_k(y_1) = a_0 a_1 ... \, a_n$ and $\hat{\Psi}_k(y_2) = a_{-1} a_{-2} ... a_{-p}$, and suppose that q is the state associated to the current configuration. Then $(y_1, y_2, q) \in \mathbb{N}^3$ gives the current configuration of M. In this sense we can consider that the transition function of M can be encoded as a function $f_M : \mathbb{N}^3 \to \mathbb{N}^3$. Actually, due to the results of [66], this function can be extended to a function $f_M : \mathbb{R}^3 \to$

$\mathbb{R}^3$ which also has the desirable properties that: (i) it can simulate M even in the presence of perturbations and (ii) the expression of each component of f can be written only by using the following terms: variables, polynomial-time computable constants (in the sense of computable analysis), $+$, $-$, $\times$, sin, cos, arctan. More formally, let $\|f\| = \sup_{x \in \mathbb{R}^l} \|f(x)\|$. By $f^{[k]}$ we denote the kth iterate of the function $f : A \to A$, which is defined as follows: $f^{[0]}(x) = x$, $f^{[j+1]}(x) = f^{[j]}(f(x))$. The following result is from [66].

Theorem 4.6 *Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of a Turing machine M, under the encoding described above and let $0 < \delta < \varepsilon < 1/2$. Then ψ admits a globally analytic elementary extension $f_M : \mathbb{R}^3 \to \mathbb{R}^3$, such that the expression of each component can be written only by using the following terms: variables, polynomial-time computable constants, $+$, $-$, $\times$,* sin, cos, arctan. *Moreover, ψ is robust to perturbations in the following sense: for all f such that $\|f - f_M\| \leq \delta$, for all $j \in \mathbb{N}$, and for all $\bar{x}_0 \in \mathbb{R}^3$ satisfying $\|\bar{x}_0 - x_0\| \leq \varepsilon$, where $x_0 \in \mathbb{N}^3$ represents an initial configuration*

$$\left\| f^{[j]}(\bar{x}_0) - \psi^{[j]}(x_0) \right\|_\infty \leq \varepsilon.$$

Note that, as noticed before, we implicitly assumed that if y is a halting configuration, then $\psi(y) = y$. The proof of this theorem is constructive and f can be obtained explicitly. The proof of this theorem can be found in [66]. Note that since the expression of each component of f_M can be written only by using the following terms: variables, polynomial-time computable constants, $+$, $-$, $\times$, sin, cos, arctan, and since all these operations are polynomial-time computable, we conclude that f_M is computable in polynomial time in the sense of computable analysis.

The next important step is to iterate the transition function of a Turing machine to be able to compute the result of its computation. The following result is also from [66].

Theorem 4.7 *Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of a Turing machine M, under the encoding described above and let $0 < \varepsilon < 1/4$. Let also $0 \leq \delta < 2\varepsilon < 1/2$. Then there exist*

- *$\eta > 0$ satisfying $\eta < 1/2$, which can be computed from $\psi, \varepsilon, \delta$*
- *A PIVP function $g_M : \mathbb{R}^7 \to \mathbb{R}^6$ which can be written only by using the following terms: variables, polynomial-time computable constants, $+$, $-$, $\times$,* sin, cos, arctan

such that the ODE $z' = g_M(t, z)$ robustly simulates M in the following sense: for all g satisfying $\|g - g_M\| \leq \delta < 1/2$ and for all $x_0 \in \mathbb{N}^3$ which codes a configuration according to the encoding described above, if $\bar{x}_0, \bar{y}_0 \in \mathbb{R}^3$ satisfy the conditions $\|\bar{x}_0 - x_0\| \leq \varepsilon$ and $\|\bar{y}_0 - x_0\| \leq \varepsilon$, then the solution $z(t)$ of

$$z' = g(t, z), \qquad z(0) = (\bar{x}_0, \bar{y}_0)$$

satisfies, for all $j \in \mathbb{N}_0$ and for all $t \in [j, j + 1/2]$

$$\left\| z_2(t) - \psi^{[j]}(x_0) \right\| \leq \eta,$$

where $z \equiv (z_1, z_2)$ with $z_1 \in \mathbb{R}^3$ and $z_2 \in \mathbb{R}^3$.

Remark 4.8 *We note that, similarly to theorem 4.6, g_M is computable in polynomial time in the sense of computable analysis.*

The following theorem is a refinement of theorem 4.7, since it provides the amount of resources needed to simulate a Turing machine with ODEs when using theorem 4.7.

Theorem 4.9 *Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of a Turing machine M under the encoding described above and let $0 < \varepsilon < 1/4$ and $0 \leq \delta < 2\varepsilon < 1/2$. Suppose that $g_M : \mathbb{R}^7 \to \mathbb{R}^6$ is the function from theorem 4.7 such that the ODE $z' = g_M(t, z)$ robustly simulates M. Let $x_0 \in \mathbb{N}^3$ be an encoding of a configuration according to the encoding described above and let $\bar{x}_0, \bar{y}_0 \in \mathbb{R}^3$ be such that $\|\bar{x}_0 - x_0\| \leq \varepsilon$ and $\|\bar{y}_0 - x_0\| \leq \varepsilon$. Then the solution $z(t)$ of $z' = g_M(t, z)$, $z(0) = (\bar{x}_0, \bar{y}_0)$ satisfies the following condition for all $t \in [0, k]$, where $k \in \mathbb{N}_0$ is arbitrary*

$$\min_{0 \leq j \leq k} \left\| \psi^{[j]}(x_0) \right\| - 1 \leq \|z(t)\| \leq \max_{0 \leq j \leq k} \left\| \psi^{[j]}(x_0) \right\| + 1.$$

The remaining of this section is devoted to prove theorem 4.9.

Proof of theorem 4.9. To prove this result, we have to analyze the structure of the proof of theorem 4.7, so we start by presenting it here. The idea to prove theorem 4.7 is to iterate the transition function given in theorem 4.6 with ODEs. We will also analyze the amount of resources used in the simulation to be able to determine how efficiently one can simulate Turing machines with ODEs.

Following along the lines of the argument presented in [66], the idea is to adapt the construction in [8] to simulate the iteration of an integer function which admits an extension to the real line $\mathbb{R}$. Although our objective is to obtain an analytic function g_M defining an ODE $z' = g_M(t, z)$ which simulates a given Turing machine M, in a first step we iterate the transition function with a non-analytic ODE. First let us introduce some auxiliary functions.

Proceeding as in [81, p. 37], let $\theta_j : \mathbb{R} \to \mathbb{R}$, $j \in \mathbb{N} - \{0, 1\}$ be the function defined by

$$\theta_j(x) = 0 \text{ if } x < 0, \quad \theta_j(x) = x^j \text{ if } x \geq 0.$$

This function is a C^{j-1} version of Heaviside's step function $\theta(x)$, where $\theta(x) = 1$ for $x \geq 0$ and $\theta(x) = 0$ for $x < 0$. Using θ_j we can obtain a function $r : \mathbb{R} \to \mathbb{R}$ which returns the integer part of a real number. More precisely, let

$$r(0) = 0, \; r'(x - 1/4) = c_j \theta_j(-\sin 2\pi x) \tag{4.10}$$

where $c_j = \left(\int_0^1 \theta_j(-\sin 2\pi x) dx \right)^{-1}$. The function r has the property that $r(x) = n$, whenever $x \in [n - 1/4, n + 1/4]$, for all integers n. Now consider the ODE

$$\begin{cases} z_1' &= \lambda_j (\tilde{f}(r(z_2)) - z_1)^3 \theta_j(\sin 2\pi t), \\ z_2' &= \lambda_j (r(z_1) - z_2)^3 \theta_j(-\sin 2\pi t), \end{cases} \tag{4.11}$$

where $\tilde{f} : \mathbb{R} \to \mathbb{R}$ is an extension of the function $f : \mathbb{N} \to \mathbb{N}$, $z_1(0) = z_2(0) = x_0 \in \mathbb{N}$ and $\lambda_j > 8c_j$. Let us see how this construction works. Let us suppose that $t \in [0, 1/2]$. In this case $z_2'(t) = 0$ and hence $z_2(t) = x_0$ and $r(z_2) = x_0$.

Therefore, the first equation of (4.11) becomes the targeting equation (4.1) with $b = \tilde{f}(r(z_2)) = f(x_0)$, $t_0 = 0$, $t_1 = 1/2$, and $\phi(t) = \theta_j(\sin 2\pi t)$. By lemma 4.4 and the choice of λ_j, we conclude that

$$|b - z_1(1/2)| = |f(x_0) - z_1(1/2)| \leq 1/4.$$

Moreover, $\min(x_0, f(x_0)) \leq z_1(t) \leq \max(x_0, f(x_0))$ for all $t \in [0, 1/2]$ also by lemma 4.4. On the time interval $[1/2, 1]$, the roles of z_1 and z_2 are reversed: we will have $z_1'(t) = 0$ and thus $z_1(t) = z_1(1/2)$ for all $t \in [1/2, 1]$. Therefore $r(z_1(t)) = f(x_0)$ for $t \in [1/2, 1]$ and the second equation of (4.11) becomes the targeting equation (4.1) with $b = r(z_1(t)) = f(x_0)$, $t_0 = 1/2$, $t_1 = 1$, and $\phi(t) = \theta_j(-\sin 2\pi t)$. By lemma 4.4 and the choice of λ_j, we conclude that

$$|b - z_2(1)| = |f(x_0) - z_2(1)| \leq 1/4.$$

Moreover, $\min(x_0, f(x_0)) \leq z_2(t) \leq \max(x_0, f(x_0))$ for all $t \in [0, 1]$ also by lemma 4.4 and $\min(x_0, f(x_0)) \leq z_1(t) \leq \max(x_0, f(x_0))$ for all $t \in [0, 1]$. Now the cycle repeats itself. On the time interval $[1, 3/2]$, we get $z_2'(t) = 0$ and thus $\tilde{f}(r(z_2)) = f(f(x_0)) = f^{[2]}(x_0)$. Repeating the same reasoning, we conclude that $\left|f^{[k]}(x_0) - z_2(t)\right| < 1/4$ for all $t \in [k, k + 1/2]$, where $k \in \mathbb{N}_0$ is arbitrary (assuming $f^{[0]}(x) = x$) and $\left|f^{[k+1]}(x_0) - z_1(t)\right| < 1/4$ for all $t \in [k + 1/2, k + 1]$, where $k \in \mathbb{N}_0$ is arbitrary. Moreover, when $t \in [k, k + 1]$, with $k \in \mathbb{N}$, we have that $\min(f^{[k]}(x_0), f^{[k+1]}(x_0)) - 1/4 \leq z_i(t) \leq \max(f^{[k]}(x_0), f^{[k+1]}(x_0)) + 1/4$ for $i = 1, 2$.

In this manner we are able to iterate (the extension of) a discrete function $f : \mathbb{N} \to \mathbb{N}$ with an ODE. However, the ODE is still not analytic as required in theorem 4.7. Notice that to have an analytic ODE, z_1' and z_2' cannot be 0 in half-unit intervals, otherwise it is well-known that if they are analytic and take the value zero in a non-empty interval, then these derivatives have to be identically equal to 0 on all their domain. To avoid this problem, instead of requiring that z_1' and z_2' take the value 0 in half-unit intervals, we require that they take values *very close* to zero; the robustness of f_M in theorem 4.6 will ensure that the whole construction will not break up when iterating the function f_M with an ODE, even if z_1' and z_2' are not exactly 0 in half-unit intervals, and thus the values of $z_1(t)$ and $z_2(t)$ might change slightly. The main challenge is to keep the magnitude of $|z_1'|$ and $|z_2'|$ sufficiently small in the half-unit intervals of interests so that the iteration can be carried out without major problems. To achieve this effect, we have to analyze the effects of introducing perturbations in (4.11).

As we have seen above, Branicky's simulation relies on the use of targeting equations of the type (4.1). If $z_2'(t)$ is not 0 in the time interval $[0, 1/2]$ and hence $z_2(t)$ is not constant there, we are still able to analyze the system (4.11) using lemma 4.5.

Following the ideas sketched above, we replace the non-analytic function $\theta_j(\sin 2\pi t)$ in the first equation of (4.11) by an analytic function which is periodic with period 1 and close to zero when $t \in [1/2, 1]$. This can be done in a two-step process. First we construct a very rough approximation of $\theta_j(\sin 2\pi t)$, which will then be improved upon on a second step. For the first step, let us consider the function s defined by

$$s(t) = \frac{1}{2}\left(\sin^2(2\pi t) + \sin(2\pi t)\right). \tag{4.12}$$

This function ranges between 0 and 1 in $[0, 1/2]$ (and, in particular, between $3/4$ and 1 when $x \in [0.16, 0.34]$), and between $-\frac{1}{8}$ and 0 on the time interval $[1/2, 1]$. Next we consider the function l_2 given in the next lemma, which was proved in [66].

Lemma 4.10 *Let $l_2 : \mathbb{R}^2 \to [0,1]$ be given by $l_2(x,y) = \frac{1}{\pi}\arctan(4y(x-1/2)) + \frac{1}{2}$. Suppose also that $a \in \{0,1\}$. Then, for any $\bar{a}, y \in \mathbb{R}$ satisfying $|a - \bar{a}| \le 1/4$ and $y > 0$, we obtain $|a - l_2(\bar{a}, y)| < 1/y$.*

We can see the function l_2 as a function which reduces the error of any value within distance $\le 1/4$ of 1 (0, respectively) by an amount specified by $1/y$ where $y > 0$ is the second argument of l_2. Now we can introduce the following analytic function $W : \mathbb{R}^2 \to [0,1]$, which is given by

$$W_0(t,y) = l_2(s(t), y) \tag{4.13}$$

to replace the non-analytic function $\theta_j(\sin 2\pi t)$ in (4.11), since $\int_0^{1/2} W_0(t,y) > 3/4 \times (0.34 - 0.16) > 0$ (assuming that $y \ge 4$) and $|W_0(t,y)| < 1/y$ for $t \in [1/2, 1]$ (i.e.y allows us to provide an error bound for $z_1'(t)$ when $z_1'(t)$ should be close to zero).

Let us also consider the following function σ introduced in [66]. This function works as a uniform contraction in a neighborhood of the integers $\mathbb{Z}$. It has the advantage that it works around every integer, contrarily to the function l_2 which works only for the integer values 0 and 1. On the other hand, l_2 allows us to prescribe the rate of contraction of the error (as the second argument of l_2), while this rate of contraction cannot be changed for σ. The following lemma is from [66].

Lemma 4.11 *Let $n \in \mathbb{Z}$ and let $\varepsilon \in [0, 1/2)$. Then there is some contracting factor $\lambda_\varepsilon \in (0,1)$ such that, $\forall \delta \in [-\varepsilon, \varepsilon]$, $|\sigma(n+\delta) - n| < \lambda_\varepsilon \delta$.*

As done in [66], we will henceforth assume that $\varepsilon \in [0, 1/2)$ is fixed and that λ_ε is the respective contracting factor given by lemma 4.11. In particular, we can take $\lambda_{1/4} = 0.4\pi - 1 \approx 0.2566371$.

We now have all the ingredients to prove theorem 4.7. Let $\gamma > 0$ be such that $2\gamma + \delta/2 \le \varepsilon < 1/4$, and let f_M be the map given by theorem 4.6 (where the values ε and δ of the statement of the theorem are replaced by the values $\eta = (\gamma + \delta)/2 + \varepsilon < 1/2$ and $\eta/2$, respectively, in what follows, although the value for δ is not relevant for the present result). Consider the system of ODEs $z' = g_M(t,z)$ given by

$$\begin{aligned} z_{1,i}' &= \lambda_{1,i}(z_{1,i} - \sigma^{[n]} \circ f_{M,i}(z_{2,i}))^3 \, \phi_{1,i}(t, z_{1,i}, z_{2,i}) \qquad (4.14) \\ z_{2,i}' &= \lambda_{2,i}(z_{2,i} - \sigma^{[n]}(z_{1,i}))^3 \, \phi_{2,i}(t, z_{1,i}, z_{2,i}) \end{aligned}$$

where $z(t) = (z_1(t), z_2(t)) \in \mathbb{R}^3 \times \mathbb{R}^3 = \mathbb{R}^6$ and $i = 1,2,3$, with initial conditions $z_1(0) = \bar{x}_0$, $z_2(0) = \bar{y}_0$, where $\bar{x}_0, \bar{y}_0 \in \mathbb{R}^3$ approximate, with error bounded by ε, some initial configuration x_0

$$\begin{aligned} \phi_{1,i}(t, z_{1,i}, z_{2,i}) &= l_2\left(s(t), \tfrac{\lambda_1}{\gamma}(z_{1,i} - \sigma^{[n]} \circ f_{M,i}(z_{2,i}))^4 + \tfrac{\lambda_1}{\gamma} + 10\right) \\ \phi_{2,i}(t, z_{1,i}, z_{2,i}) &= l_2\left(s(-t), \tfrac{\lambda_2}{\gamma}(z_{2,i} - \sigma^{[n]}(z_{1,i}))^4 + \tfrac{\lambda_2}{\gamma} + 10\right) \qquad (4.15) \end{aligned}$$

and $\lambda_{1,i}, \lambda_{2,i}$, $i = 1, 2, 3$, are constants to be defined in a moment. The ODE (4.14) works in a similar way to the ODE (4.11) when simulating the iteration of f_M, although it is formed by a total of 6 components. The constants $\lambda_{1,i}, \lambda_{2,i}$, $i = 1, 2, 3$ are constants associated to a targeting error of value γ. The main difference comparatively to (4.11) is that extra terms are allowed to account for the introduction of errors. Since we want that the ODE $z' = g_M(t, z)$ simulates M in a robust manner, let us assume that the right-hand side of the equations in (4.14) can have an error of absolute value not exceeding δ. To analyze the behavior of (4.14), let us start with the time interval $[0, 1/2]$. For readability reasons, we will now drop the component index when analyzing the behavior of z_1 and z_2 (i.e. we will write simply z_1 instead of $z_{i,1}$, etc.), unless there is risk of notational confusion. We have that $|s(-t)| \leq 1/8$ and, by lemma 4.10 (note that $|x|^3 \leq x^4 + 1$ for all $x \in \mathbb{R}$), ϕ_2 is less than $\min(\gamma(\lambda_2 \left\|z_2 - \sigma^{[n_2]}(z_1)\right\|_\infty^3)^{-1}, 1/10)$. This implies that $\|z_2'(t)\| \leq \gamma + \delta$ for $t \in [0, 1/2]$. Because the initial condition has error bounded by ε, z_2' in (4.15) is perturbed by an amount not exceeding δ and hence $|z_2(t) - z_2(0)| \leq (\gamma + \delta)/2$ for all $t \in [0, 1/2]$, one has

$$\|z_2(t) - x_0\| \leq \frac{\gamma + \delta}{2} + \varepsilon = \eta < \tfrac{1}{2} \quad \text{for all } t \in [0, 1/2]. \tag{4.16}$$

Due to theorem 4.6, we conclude that

$$\|f_M(z_2(t)) - \psi(x_0)\| \leq \eta < \tfrac{1}{2} \quad \text{for all } t \in [0, 1/2].$$

Therefore, for n large enough (depending only on η: it suffices to choose an n such that $\sigma^{[n]}(\eta) \leq \gamma$), one has $\|\sigma^{[n]} \circ f_M(z_2(t)) - f_M(x_0)\| < \gamma$ for all $t \in [0, 1/2]$. Moreover, as we have seen above, there is some non-empty time interval $[0.16, 0.34]$, where $s(t) \in [3/4, 1]$, which implies by lemma 4.10 that ϕ_1 satisfies the assumptions of function ϕ in lemma 4.5. Therefore, the behavior of z_1 is given by lemma 4.5 and

$$\left\|z_1\left(\frac{1}{2}\right) - \psi(x_0)\right\| < 2\gamma + \delta/2 \leq \varepsilon. \tag{4.17}$$

In the interval $[1/2, 1]$ the roles of z_1 and z_2 are reversed and one concludes, using arguments similar to those used in the time interval $[0, 1/2]$ that $\|z_1'(t)\| \leq \gamma + \delta$ for $t \in [1/2, 1]$ and hence that $\|z_1(t) - z_1(1/2)\| \leq (\gamma + \delta)/2$ for all $t \in [1/2, 1]$. This inequality, together with (4.17) yields that

$$\|z_1(t) - \psi(x_0)\| < \varepsilon + (\gamma + \delta)/2 = \eta \quad \text{for all } t \in [1/2, 1].$$

We have that $\left\|\sigma^{[n]}(z_1(t)) - \psi(x_0)\right\| \leq \gamma$ from the definition of n and lemma 4.5 yields

$$\|z_2\,(1) - \psi(x_0)\| < 2\gamma + \delta/2 \leq \varepsilon. \tag{4.18}$$

Again, on the time interval $[1, 3/2]$, the roles of z_1 and z_2 will be switched and we will have that $\|z_2'(t)\| \leq \gamma + \delta$ for $t \in [1, 3/2]$ and hence that $\|z_2(t) - z_2(1)\| \leq (\gamma + \delta)/2$ for all $t \in [1/2, 1]$. This inequality, together with (4.18) yields again that

$$\|z_2(t) - \psi(x_0)\| < \varepsilon + (\gamma + \delta)/2 = \eta \quad \text{for all } t \in [1, 3/2]$$

and therefore the whole procedure can be repeated for subsequent time intervals. Therefore we conclude that for $j \in \mathbb{N}_0$, if $t \in [j, j + \frac{1}{2}]$ then $\|z_2(t) - \psi^{[j]}(x_0)\| \leq \varepsilon$. This proves theorem 4.7.

Concerning the proof of theorem 4.9, we note that if in the previous simulation we apply the bounds given by lemma 4.5, namely condition 2, we conclude that on the time interval $[0, 1/2]$ we have

$$\min(x_0 - \varepsilon, \psi(x_0) - \gamma) - \delta/2 \leq z_1(t) \leq \max(x_0 + \varepsilon, \psi(x_0) + \gamma) + \delta/2.$$

From this equation and noting that $\|z_1(t) - z_1(1/2)\| \leq (\gamma + \delta)/2$ for all $t \in [1/2, 1]$, we conclude that (note that $0 < \gamma < \varepsilon$, $\delta < 1/2$, and $\varepsilon + (\gamma + \delta)/2 = \eta$), for all $t \in [0, 1]$ we have

$$\begin{aligned}
&\min(x_0 - \varepsilon, \psi(x_0) - \gamma) - \delta - \gamma/2 \leq z_1(t) \\
&\qquad \leq \max(x_0 + \varepsilon, \psi(x_0) + \gamma) + \delta + \gamma/2 \quad \Longrightarrow \\
&\min(x_0, \psi(x_0)) - \varepsilon - \delta - \gamma/2 \leq z_1(t) \leq \max(x_0, \psi(x_0)) + \varepsilon + \delta + \gamma/2 \quad \Longrightarrow \\
&\min(x_0, \psi(x_0)) - \eta - \delta/2 \leq z_1(t) \leq \max(x_0, \psi(x_0)) + \eta + \delta/2 \quad \Longrightarrow \\
&\min(x_0, \psi(x_0)) - 1 \leq z_1(t) \leq \max(x_0, \psi(x_0)) + 1.
\end{aligned}$$

Similarly, by (4.16), we have that $\|z_2(t) - x_0\| < (\gamma + \delta)/2$ for all $t \in [0, 1/2]$, which implies that $|z_2(1/2) - x_0| < (\gamma + \delta)/2$ and, by condition 2 of lemma 4.5, we have, on the time interval $[1/2, 1]$,

$$\min(z_2(1/2), \psi(x_0) - \gamma) - \delta/2 \leq z_2(t) \leq \max(z_2(1/2), \psi(x_0) + \gamma) + \delta/2.$$

Combining both inequalities, we conclude that, for all $t \in [0, 1]$

$$\begin{aligned}
&\min(x_0 - (\gamma + \delta)/2, \psi(x_0) - \gamma) - \delta/2 \leq z_2(t) \\
&\qquad \leq \max(x_0 + (\gamma + \delta)/2, \psi(x_0) + \gamma) + \delta/2 \quad \Longrightarrow \\
&\qquad \min(x_0, \psi(x_0)) - \gamma - \delta \leq z_2(t) \leq \max(x_0, \psi(x_0)) + \gamma + \delta \quad \Longrightarrow \\
&\qquad \min(x_0, \psi(x_0)) - 1 \leq z_2(t) \leq \max(x_0, \psi(x_0)) + 1.
\end{aligned}$$

Proceeding similarly for subsequent intervals, we conclude the theorem. ■

Remark 4.12 *We note that since g_M can be written only by using the following terms: variables, polynomial-time computable constants, $+$, $-$, $\times$,* sin*,* cos*,* arctan*, and since all these terms belong to* GPVAL*, it is always possible to rewrite the IVP involving the ODE $z' = g_M(t, z)$ as a PIVP with initial condition in* GPVAL *and whose solution is polynomially bounded thanks to lemma 1.13.*

4.3 Space emulation and FPSPACE equivalence

In this section we present the new notion of emulation that replaces the concept introduced by definition 2.26 for the case of time complexity classes. This notion of space emulation exploits the benefits of using the encoding $\hat{\Psi}$ from definition 4.1 to encode the Turing machines configurations together with the benefits of simulating these machines by means of theorem 4.9, allowing us to achieve our main characterization result for the class FPSPACE. To provide an effective definition of space emulation we have to take in account some new problematics that are not present for the case of time complexity classes. A problem which does not appear directly in the characterization of P is the need to know when the system already has the correct solution – just reaching some space bound is not enough to conclude that the computation has stopped, contrarily to a time bound. Hence we have to consider a way of detecting if the system has already produced a correct answer. We use the ideas presented in [20] for deciding discrete sets by means of ODEs and use a variable y_1 which allows us to detect if the system already produced the correct answer if a certain threshold has been reached.

Another issue is that we might be able to decrease the bound on the size of the solution by using more accuracy, which also has a cost in practice that might not be reflected in a notion of space. For this reason, in our notion of space emulation, we assume that our "measuring instruments" only have access to some fixed precision. This yields the following definition (see the comments below the definition for more explanations).

Definition 4.13 *Let $f : \Gamma^* \to \Gamma^*$ and $g : \mathbb{R}_+ \to \mathbb{R}_+$ be two functions. We say that f is (Ψ-)emulable in space g by an ODE*

$$\begin{cases} y' = p(y, z) \\ z' = q(y, z) \end{cases} \tag{4.19}$$

where p, q are functions formed by polynomial components, if there are two (vector-valued) function $r, s \in \mathrm{GPVAL}$, and $\varepsilon > 0$, $\tau \geq \alpha > 0$, $j, l, k \in \mathbb{N}$ with $0 < j \leq l$ such that, for all $w \in \mathrm{dom}(f) \subseteq \Gamma^*$ and for any Ψ-bound ϕ one has that the solution of the IVP formed by (4.19) and the initial condition $y(0) = r(x)$, $z(0) = s(y(0)) = s \circ r(x)$, where $\Psi(x) = w$, satisfies

1. ***(halting decision is irreversible)*** *If $t_0 > 0$ is such that $y_1(t_0) \geq 1$, then $y_1(t) \geq 1$ for all $t \geq t_0$ and $y_1(t) \geq 3/2$ for all $t \geq t_0 + 1$*
2. ***(halting decision is eventually taken)*** *There is some $t_0 \geq 0$ such that $y_1(t_0) \geq 1$*
3. ***(correct output)*** *If $y_1(t) \geq 1$, then $\Psi(y_2(t), \ldots, y_j(t)) = f(w)$*
4. ***(bounded space)*** *$\|(y(t), z(t))\| \leq \phi \circ g(x)$ for any x satisfying $\Psi(x) = w$ and for all $t \geq 0$*
5. ***(robustness to perturbations)*** *For any $\bar{t}_0 \geq 0$, if $z_1(\bar{t}_0) \geq 1$ and $\bar{y}_0$ is such that $\|\bar{y}_0 - y(\bar{t}_0)\| \leq \varepsilon$, then the solution $(\bar{y}, \bar{z})$ of (4.19) with the initial condition $\bar{y}(0) = \bar{y}_0$ and $\bar{z}(0) = s(\bar{y}_0)$ satisfies conditions 1–4 above*
6. ***(robustness is common)*** *For any $b > a \geq 0$ such that $|b - a| \geq \tau$, there is an interval $I = [c, d] \subseteq [a, b]$, with $|d - c| \geq \alpha$, such that $z_1(t) \geq 3/2$ for all $t \in I$*
7. ***(robustness preserves main properties)*** *If $(\tilde{y}, \tilde{z})$ is a solution of (4.19) with the initial condition $\tilde{y}(0) = \tilde{y}_0$, $\tilde{z}(0) = s(\tilde{y}_0)$ such that $\|\tilde{y}_0 - y(\bar{t}_0)\| \leq \varepsilon$ as in condition 5, then we can take $\tilde{y}$ in place of y in conditions 5 and 6 if $\bar{t}_0 \geq \tau$. Moreover, if we repeat this procedure an arbitrarily number of times, there will still be some t_0, counted from $t = 0$ in the original ODE (4.19), such that condition 1 holds.*

Some comments are in order for definition 4.13. First we note that the system (4.19) is really just one ODE, which is separated into two parts. The first part defines the dynamics of the *master* variables y while the second part defines the dynamics of the *auxiliary* variables z. The idea is that, in each instant, the meaningful information about the current state (or "configuration") is fully encoded on the master variables, while the remaining variables are just auxiliary in nature and can be readily and easily computed from the master variables, as we will see. The initial condition depends on two functions $r, s \in$ GPVAL. This means that the initial conditions can be easily computed from the input x by a polynomial ODE using reasonable (polynomial) resources. The fact that r and s may be multidimensional is not problematic, since r (and s) is the unique solution of a polynomial ODE and we may consider that each argument x_i depends on a time variable t. Due to the uniqueness of the solution of the polynomial ODE [51] the value of r only depends on the current value of each $x_i(t)$ for a given time instant t. This allows us to update the values of r and of s to their values $r(x)$ and $s \circ r(x)$ using a finite amount of resources. The condition $y_1(t) \geq 1$ signals that the computation has already ended with the correct result (condition 3) and that once the system ended, it will stay on the same state (condition 1). We also require that the system eventually halts (condition 2) and if $y_1(t_0) \geq 1$, then $y_1(t) \geq 3/2$ after, at most, one time unit. This is to avoid problems sampling the output and deciding whether the computation has finished if we only have access to a fixed precision measuring "device". Condition 4 states that if (4.19) computes f in space g, then $\|(y(t), z(t))\|$ must be less or equal to any Ψ-bound for g for all times $t \geq 0$. Condition 5 can be seen as follows. The ODE has the master variables $y_1, \ldots, y_l$ which provide all the relevant information about the computation. The other remaining variables $z_1, \ldots, z_k$ can be thought as "auxiliary" – they are useful to help update the main variables $y_1, \ldots, y_l$, but have no other relevant function besides that. Similarly to what happens with Turing machine, we would like to be able to stop the computation, store the "configuration" of the ODE, and then restart the computation where we left it and still be able to carry on the computation faithfully. For that reason we just need to store the values of the main variables $y_1, \ldots, y_l$, which work as the configuration. Since in practice we only can store these values with some restricted precision, say ε, we want that any meaningful computation is robust to this slight change of values. We do not need to record the values of the auxiliary variables, since those can be easily recovered from the values of the master variables, and thus we do not need to ensure a robustness condition for the auxiliary variables. Moreover, some parts of the computation might be too sensitive to be sampled and re-used as the start of another simulation. For example, if we see a digital computer as an analog computer, the voltages of the circuit might be at certain time instants changing quickly and between the voltage values which correspond to the bits "0" and "1". Sampling the circuit at that time with a certain accuracy might be especially prone to errors, so it might be more suitable to sample when the values of the voltages have stabilized. That's why we use the variable z_1 to flag when it is appropriate to store the values of the master variables, and this must happen frequently. Note that it may well be the case that we can sample the system at any time without problems. Condition 5 also encompasses this case (e.g. add a variable z_1 defined by $z_1(0) = 3/2$, $z_1'(t) = 0$). This provides the rationale behind condition 5. Condition 6 asserts that the system can be sampled often. Condition 7 is mainly for the case where we stop and restart the computation many times. The introduction of the time step $\tau > 0$ is needed to ensure that the system has enough time to "stabilize", using negative feedback or other similar behavior to correct for imprecisions on the input. If we do not use the time step requirement, we could stop and restart the system several times in a very quick succession, without giving the system opportunity to correct itself, until the initial error is made arbitrarily high and the whole simulation is destroyed. We also note that many of the ideas about the conditions 5–7 come from similar requirements from engineering and from topics such as structural stability or shadowing in dynamical systems theory. In condition 7, there could be a problem when we stop the computation and restart it with an approximate initial condition an arbitrary number of times as in condition 5, since each new computation might take a longer time than the preceding one to satisfy condition 1, and thus never halt. Therefore we include a requirement in condition 7 that this cannot happen.

At this point we are ready to present the main theorem that describes our characterization of FPSPACE.

Theorem 4.14 *Let $f : \Gamma^* \to \Gamma^*$ be a function. Then $f \in$ FPSPACE if and only if f is $\hat{\Psi}$-emulable in polynomial space by a polynomial ODE.*

4.4 Proof of theorem 4.14

To prove theorem 4.14, we will use the following result

Theorem 4.15 *Let $f : \Gamma^* \to \Gamma^*$ be a function which is computable by a one-tape Turing machine in space g, where $g : \mathbb{N} \to \mathbb{N}$ is a non-decreasing function satisfying $g(n) \geq n$. Suppose that $\bar{g} : \mathbb{R} \to \mathbb{R}$ is a non-decreasing function which is an extension of g. Then f is ($\hat{\Psi}$-)emulable in space $P \circ \bar{g}$ by a polynomial ODE, where P is a polynomial.*

Proof of theorem 4.15. First we note that if a one-tape Turing machine M computes f in space g and if (4.8) is the tape contents of M at a given instant of the computation which started with an input $w \in \Gamma^*$, where the head is reading symbol a_0, $y_1, y_2 \in \mathbb{R}$ satisfy (4.9), $q \in \{1, \ldots, m\}$ is the current state and $|y_3 - q| \leq 1/4$, we conclude that (note that M uses at most k symbols)

$$\|(y_1, y_2, q)\| \leq k^{g(|w|)+1} + m + 1.$$

From theorem 4.9, we also conclude that M with input w can be simulated by an IVP

$$\begin{cases} y' = g_M(y) \\ y(0) = (x_0, 0, 1, x_0, 0, 1) \end{cases} \tag{4.20}$$

as in theorem 4.7, assuming that $\hat{\Psi}(x_0) = w$ and that 1 corresponds to the initial state. Moreover, we get that the solution x of this IVP satisfies the following condition for all $t \geq 0$

$$\|x(t)\| \leq k^{g(|w|)+1} + m + 2.$$

Now, following the observation of remark 4.12, we can use the closure property of GPVAL expressed by lemma 1.13 to conclude that the IVP (4.20) can be expanded to a polynomial IVP such that the initial condition y_0 satisfies $y_0 = \alpha(x_0)$, where $\alpha \in$ GPVAL. Moreover there is a polynomial R, which without loss of generality we can assume to be increasing, such that if y is the solution of the IVP, then

$$\|y(t)\| \leq R(\|x(t)\|) \leq R\left(k^{g(|w|)+1} + m + 2\right) \leq 2^{P\circ\bar{g}(\|x_0\|)} - 1$$

for some polynomial P which we can assume to have positive integers as coefficients. In particular that implies that the solution of the ODE satisfies condition 4 of definition 4.13. We have now to detect when the computation has halted. Let us assume that m denotes the halting state and suppose y_3 encodes the current state of the TM in the simulation of the TM with our polynomial ODE derived from (4.20). Note that $y_3(t) - m < -1/2$ until the machine halts. There might be a brief transient time period of size bounded by $1/2$ where $y_3(t)$ is quite near to m, but where the other components are not close enough to their correct value. Hence we cannot directly use $y_3(t) - m + 1/2$ as a variable to signal that the simulation has ended. However, we know that there is another variable, y_6, from the "mirror" simulation of the ODE, which will update its value in the next half-unit time interval to near m. So we can use a new variable u_1 to signal the end of the computation, by taking

$$u_1(t) = 2l_2(\sigma^{[b]}(y_3(t) - m + 1), 10) \cdot l_2(\sigma^{[b]}(y_6(t) - m + 1), 10)$$

where $b \in \mathbb{N}_0$ is chosen so that $\sigma^{[b]}(\eta) \leq 1/4$, where σ and l_2 are defined by lemmas 4.11 and 4.10. Note that u_1 can be written as the solution of a PIVP function since it is the composition of PIVP functions. Since the Turing machine eventually halts and the state will be kept near m, we conclude that conditions 1 and 2 are satisfied and condition 4 continues to be satisfied (reordering the components of the ODE so that u_1 is the first component). For condition 5, 6 and 7, they follow immediately by taking $\tau = 2$, and by considering intervals $[c, d]$ with the format $[0.16 + i, 0.34 + i]$, where $i \in \mathbb{N}$ since the function s defined by (4.12) takes values between $3/4$ and 1 when $x \in [0.16, 0.34]$ and it has period 1, so it suffices to take as first component of the auxiliary variables $z_1(t) = 2s(t)$. Note also that the ODEs defining s are autonomous, since $\sin(2\pi t)$ corresponds to the first component of the solution of the IVP $y_1' = 2\pi y_2$, $y_2' = -2\pi y_1$, $y_1(0) = 0$ and $y_2(0) = 1$. ■

An obvious corollary to this theorem, we get the following result.

Theorem 4.16 *If $f \in$ FPSPACE, then f is ($\hat{\Psi}$-)emulable in polynomial space by a polynomial ODE.*

To prove theorem 4.14, we just need the reverse direction of the above theorem.

Theorem 4.17 *If $f : \Gamma^* \to \Gamma^*$ is a function which is ($\hat{\Psi}$-)emulable in polynomial space by a polynomial ODE, then $f \in$ FPSPACE.*

Proof of theorem 4.17. The idea is to numerically simulate (4.19) using the algorithm of [36], due to theorem 3.11. The analysis of the algorithm presented in theorem 3.11 focus on time complexity boundaries, while now we are interested on space complexity. Nevertheless, by a careful observation of the argument described in [36] it is straightforward to define a space boundary for the same algorithm, as presented by the following theorem. Indeed we remark that the algorithm which proves theorem 3.11 in [36] is obtained by using a Taylor method which provides approximations (via Taylor approximations of variable order) of the solution of the polynomial ODE at times $a = t_0, t_1, t_2, \ldots, t_N = b$ and the number of steps performed by the algorithm depends polynomially on $\mathrm{PsLen}_y(a, b)$. On the other hand, the number of bits used depends only on $\log \mathrm{PsLen}_y(a, b)$. Hence we have the following theorem.

Theorem 4.18 *Let $p : \mathbb{R}^n \to \mathbb{R}^n$ be a vector-valued function formed by polynomials of degree at most k. Assume that $y : \mathbb{R} \to \mathbb{R}^n$ is a solution of a PIVP defined with the initial condition $y(a) = y_0$, with $a \in \mathbb{Q}$ and $y_0 \in \mathbb{Q}^n$. Then $y(b)$, where $b \in \mathbb{Q}$ satisfies $b \geq a$, can be computed with precision $2^{-\mu}$, $\mu \in \mathbb{N}_0$, in space bounded by poly$(\deg(p), \log \mathrm{PsLen}_y(a, b), \log \|y_0\|, \log \Sigma p, \mu)^n$.*

At this point we can continue with the proof of theorem 4.17. Let (y, z) be the solution of (4.19) which emulates f with argument $\hat{\Psi}(x) = w$. Due to condition 4 of definition 4.13, we know that there is a polynomial P such that $\|(y(t), z(t))\| \leq 2^{P(|x|)}$ for all $t \geq 0$. Then using the algorithm of [36], we can compute $(y(t_i), z(t_i))$ with accuracy $\varepsilon/2 < 1/4$, obtaining approximations $\bar{y}_{(0)}, \bar{z}_{(0)}, \bar{y}_{(1)}, \bar{z}_{(1)}, \ldots, \bar{y}_{(L)}, \bar{z}_{(L)}$ where $\tau = t_0, t_1, \ldots, t_L = 2\tau$ are rational numbers satisfying $|t_{i+1} - t_i| \leq \alpha$. Due to condition 5 and 6, we know that there is at least some t_i such that $\bar{z}_{(i),1} \geq 5/4$, which implies that $z_1(t_i) \geq 1$ and thus that one can store the values the values $\bar{y}_{(i)}$. We note that the computation needed to obtain $\bar{y}_{(i)}$ can be carried out in polynomial space due to theorem 4.18. Next we repeat the procedure for the initial value problem obtained by using (4.19) and $y(0) = \bar{y}_{(i)}$, $z(0) = s(\bar{y}_{(i)})$. We repeat this procedure until we eventually conclude that $\bar{y}_{(i),1} \geq 5/4$ for some i. At that time we know that the system has halted (in the sense of condition 1 of definition 4.13) and since $\hat{\Psi}$ only has one argument, according with definition 4.13, we round $\bar{y}_{(i),2}$ to the nearest integer a. Then we will have $\hat{\Psi}(a) = f(w)$, and it will be a straightforward computation to return $f(w)$ from the integer a with a Turing machine. Therefore we were able to compute $f(w)$ in space polynomial in $|w|$, which implies that $f \in$ FPSPACE. ■

Chapter 5

Characterizations of classes of languages

Until now we have described connections between analog classes of functions (ATSP, ATSE) and discrete classes of computable functions (FP, FEXPTIME and FPSPACE) by means of discrete emulation or space-emulation. At this point a very natural question arises: is it then possible to extend these results to discrete classes of computable sets as well, such as PSPACE and EXPTIME? The answer to this important question is yes, and the latter can be done by modifying the criterion of acceptance in the definition of the analog classes. Indeed, it won't be required any convergence of the dynamical system, but the solution of the PIVP, with initial condition dependent from the input word as usual, will be required to enter a certain region to determinate whether the input word w belongs or not to the considered set. This modification will allow solutions of ODEs to *decide* a certain class of sets of words over the considered alphabet Γ, and therefore it will be possible to introduce a new analog class closely related to EXPTIME. A similar procedure has been followed by the authors of [20] to extend their results obtained for polynomial computable functions to the complexity class P and, as we just mentioned above, the same idea has inspired the formulation of condition 1 in the definition of space-emulation in 4.13. We start with the definition of this new analog class, related to languages, that we will call *Exponential-Analog-Recognizable*, or EAR.

Definition 5.1 (EAR) *A language $L \subseteq \Gamma^*$ is called exponential analog recognizable if there exist a vector q of bivariate polynomials and a vector p of polynomials with d variables, both with coefficients in $\mathbb{R}_P$, and an exponential boundary function $\Pi : \mathbb{R}_+ \to \mathbb{R}_+$ such that $\forall w \in \Gamma^*$ there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$*

- *$y(0) = q(\Psi_k(w))$ and $y'(t) = p(y(t))$*
- *if $|y_1(t)| \geq 1$ then $|y_1(u)| \geq 1$ for all $u \geq t$*
- *if $w \in L$ (respectively, $w \notin L$) and $len_y(0,t) \geq \Pi(|w|)$ then $y_1(t) \geq 1$ (respectively, $y_1(t) \leq -1$)*
- *$len_y(0,t) \geq t$*

where $len_y(0,t)$ *represents the length of the solution* y *in the interval* $[0,1]$.

As it is possible to see from this definition, the first variable of the system $y_1(t)$ is the variable that is actually deciding the word w considered as an input to the model. As always the relationship between the dynamical system and the input w is obtained by means of the initial condition q which is a polynomial function of the real encoding of the word w, expressed with $\Psi_k(w)$ as usual. The second condition of the above list makes sure that the decision is stable, forcing the variable $y_1(t)$ to *maintain* its decision once it has been made. The third condition requires the length of the solution to be at most exponential on the input in the moment in which the encoded word has been decided. This condition on the length of the system is directly connected with the use of the ATSE class and the implications of theorem 3.9. Finally, condition four is important to avoid including in the class solutions with too slow evolutions, as for example the case of solutions close to an equilibrium point.

For the case of PSPACE, the equivalence for languages is once again described starting from the result obtained for functions, where the notion of space-emulation is modified to describe space-decidability. The key difference in this case is expressed by condition 3, which requires the solution of the dynamical system to reach a fixed threshold represented by the value 1 for words belonging to the language, and by the value -1 for words not in the language.

Definition 5.2 (Space-decidability) *Let* $L \subseteq \Gamma^*$ *be a language and* $g : \mathbb{R}_+ \to \mathbb{R}_+$ *be a function. Then we say that* L *is* $(\Psi$*-)decidable by an ODE (4.19) in space* g *if there are (vector-valued) functions* $r, s \in \mathrm{GPVAL}$ *and* $\varepsilon > 0$, $\tau \geq \alpha > 0$, $\alpha, \tau \in \mathbb{Q}$, $l, k \in \mathbb{N}$ *such that, for all* $w \in \mathrm{dom}(f) \subseteq \Gamma^*$ *and for any* Ψ*-bound* ϕ *one has that the solution of the IVP formed by (4.19) and the initial condition* $y(0) = r(x)$ *and* $z(0) = s(y(0)) = s \circ r(0)$*, where* $\Psi(x) = w$*, satisfies*

1. ***(halting decision is irreversible)*** *If* $t_0 > 0$ *is such that* $y_1(t_0) \geq 1$*, then* $y_1(t) \geq 1$ *for all* $t \geq t_0$ *and* $y_1(t) \geq 3/2$ *for all* $t \geq t_0 + 1$
2. ***(halting decision is eventually taken)*** *There is some* $t_0 \geq 0$ *such that* $y_1(t_0) \geq 1$
3. ***(correct output)*** *If* $\Psi(x) \in L$ *(respectively* $\notin L$*) and* $y_1(t) \geq 1$ *then* $y_2(t) \geq 1$ *(respectively* ≤ -1*)*
4. ***(bounded space)*** $\|(y(t), z(t))\| \leq \phi \circ g(x)$ *for any* x *satisfying* $\Psi(x) = w$ *and for all* $t \geq 0$
5. ***(robustness to perturbations)*** *For any* $\bar{t}_0 \geq 0$*, if* $z_1(\bar{t}_0) \geq 1$ *and* $\bar{y}_0$ *is such that* $\|\bar{y}_0 - y(\bar{t}_0)\| \leq \varepsilon$*, then the solution* $(\bar{y}, \bar{z})$ *of (4.19) with the initial condition* $\bar{y}(0) = \bar{y}_0$ *and* $\bar{z}(0) = s(\bar{y}_0)$ *satisfies conditions 1–4 above;*
6. ***(robustness is common)*** *For any* $b > a \geq 0$ *such that* $|b - a| \geq \tau$*, there is an interval* $I = [c, d] \subseteq [a, b]$*, with* $|d - c| \geq \alpha$*, such that* $z_1(t) \geq 3/2$ *for all* $t \in I$
7. ***(robustness preserves main properties)*** *If* $(\tilde{y}, \tilde{z})$ *is a solution of (4.19) with the initial condition* $\tilde{y}(0) = \tilde{y}_0$*,* $\tilde{z}(0) = s(\tilde{y}_0)$ *such that* $\|\tilde{y}_0 - y(\bar{t}_0)\| \leq \varepsilon$ *as in condition 5, then we can take* $\tilde{y}$ *in place of* y *in conditions 5 and*

6 if $\bar{t}_0 \geq \tau$. Moreover, if we repeat this procedure an arbitrarily number of times, there will still be some t_0, counted from $t = 0$ in the original ODE (4.19), such that condition 1 holds.

The results of the following section provide an interesting new perspective on the famous open problem of standard complexity theory: EXPTIME vs PSPACE. Indeed, proving a separation result between the two analog characterizations consequently proves the same result for the standard open problem, and vice versa. Therefore, this constitutes a reduction from a hard problem of complexity theory to a problem of analysis formulated by means of polynomial ODEs.

5.1 Equivalence relation for EXPTIME

Theorem 5.3 (EXPTIME equivalence) *For any language $L \subseteq \Gamma^*$, $L \in$ EXPTIME if and only if L is exponential analog recognizable.*

Proof of theorem 5.3. The proof is similar to the what was done in [20] for the polynomial case. Let $L \in$ EXPTIME. Then there exists a function $f \in$ FEXPTIME and two distinct symbols $\bar{0}, \bar{1} \in \Gamma$ such that for any $w \in \Gamma^*$, $f(w) = \bar{1}$ if $w \in L$ and $f(w) = \bar{0}$ otherwise. By theorem 3.9 there is some $g \in$ ATSE that emulates f. Since for any $w \in \Gamma^*$ one has $f(w) \in \{\bar{0}, \bar{1}\}$, this implies that $\Psi_k(f(w)) = (k^{-1}\gamma(\bar{0}), 1)$ or $\Psi_k(f(w)) = (k^{-1}\gamma(\bar{1}), 1)$. Next define a function $res : \{k^{-1}\gamma(\bar{0}), k^{-1}\gamma(\bar{1})\} \to \{-2, 2\}$ which is defined by $res(k^{-1}\gamma(\bar{0})) = -2$ and $res(k^{-1}\gamma(\bar{1})) = 2$. Using Lagrange interpolation, by theorem 2.20 we can extend res to a function $L_{res} \in$ ATSP which extends res to $\mathbb{R}$. Now take $g^*(x) = L_{res}(g_1(x))$, where $g(x) = (g_1(x), g_2(x))$. Since g^* is the composition of an ATSP function with an ATSE function, by theorem 3.8 we conclude that $g^* \in$ ATSE. Moreover, $g^*(\Psi_k(w)) = 2$ if $w \in L$ and $g^*(\Psi_k(w)) = -2$ if $w \notin L$.

From the definition of the ATSE class we know that $g^* \in$ ATSE $(\Pi_1\Pi_2, \Upsilon_1\Upsilon_2)$ for some exponential boundary functions Π_1, Υ_1 and some polynomials Π_2, Υ_2 with corresponding d, p, q as parameters and functions defining the dynamical system. Assume, without loss of generality, that these four functions used as boundaries are increasing functions. Let $w \in \Gamma^*$ and consider the following system, where v is a constant variable used to store the input and in particular the input length ($v_2(t) = |w|$), $\tau(t) = t$ is used to keep the time, z is the decision variable, and $\tau^* = \Pi_1(v_2(t))\Pi_2(\ln 2) = \Pi_1(|w|)\Pi_2(\ln 2)$

$$\begin{cases} y'(t) = p(y(t)) \\ v'(t) = 0 \\ z'(t) = \operatorname{lxh}_{[0,1]}(\tau(t) - \tau^*, 1 + \tau(t), y_1(t) - z(t)) \\ \tau'(t) = 1 \end{cases}$$

where $y(0) = q(\Psi_k(w))$, $v(0) = \Psi_k(w)$, $z(0) = 0$, and $\tau(0) = 0$. Let $t \in [0, \tau^*]$. Then, by the properties of lxh (see proposition 2.14), one has $|z'| \leq e^{-1-t}$. This implies that

$$\begin{aligned} z(t) = \int_0^t z'(u)du \quad \Rightarrow \\ |z(t)| \leq \int_0^t |z'(u)|\, du \leq \int_0^t e^{-1-u} du \leq e^{-1} - e^{-1-t} \leq e^{-1}. \end{aligned} \tag{5.1}$$

In particular we conclude that $|z(t)| < 1$ for $t \in [0, \tau^*]$ and therefore that the system has not decided whether w should be accepted or rejected for times $\leq \tau^*$.

Let us now consider the case when $t \geq \tau^*$. By definition of ATSE, we have $\|y_1(t) - g^*(\Psi_k(w))\| \leq e^{-\ln 2}$. Recall that $g^*(\Psi_k(w)) \in \{-2, 2\}$ and let $s \in \{-1, 1\}$ be such that $g^*(\Psi_k(w)) = 2s$. Then $\|y_1(t) - 2s\| \leq \frac{1}{2}$, which means that $y_1(t) = s\lambda(t)$, where $\lambda(t) \geq \frac{3}{2}$. By (5.1), we conclude that $z(\tau^*) \in [-e^{-1}, e^{-1}]$. From proposition 2.14, we also conclude that z satisfies, for $t \geq \tau^*$

$$z'(t) = \phi(t)(s\lambda(t) - z(t))$$

where $1 > \phi(t) > 0$. Let us assume, without loss of generality, that $s = 1$ (a similar reasoning can be applied for the case $s = -1$). Then the previous equation gives us, for $t \geq \tau^*$

$$z'(t) = \phi(t)(\lambda(t) - z(t)) \geq \phi(t) \left(\frac{3}{2} - z(t) \right) \tag{5.2}$$

Furthermore, since $z(\tau^*) \in [-e^{-1}, e^{-1}]$, we conclude that z is strictly increasing when $t \geq \tau^*$. To see this, consider a variable $r(t)$ defined by $r'(t) = 3/2 - r(t)$ and $r(\tau^*) = z(\tau^*)$. We can explicitly solve the ODE for r and conclude that it converges to $3/2$ with a rate of convergence of the order of e^{-t} and stays below $3/2$ for all $t \geq \tau^*$. Furthermore, using standard results from ODEs we can conclude that $3/2 > r(t) \geq z(t)$ for all $t \geq \tau^*$. Knowing that $z(\tau^*) \in [-e^{-1}, e^{-1}]$ this implies that $z(t)$ is strictly increasing for all $t \geq \tau^*$.

By proposition 2.14, we have that for $t \geq \tau^* + 1$

$$\begin{aligned} \left| y_1(t) - z(t) - \mathrm{lxh}_{[0,1]}(\tau(t) - \tau^*, 1 + \tau(t), y_1(t) - z(t)) \right| \leq \\ \leq e^{-1-\tau(t)} \leq e^{-1} \end{aligned}$$

This inequality and the definition of $z(t)$ yield that

$$|y_1(t) - z(t) - z'(t)| \leq e^{-1}. \tag{5.3}$$

We will now show that if $t \geq \tau^{**} = \tau^* + 4/(1 - 2e^{-1})$, then $z(t) \geq 1$. To show that it suffices to show that $z(\tau^{***}) \geq 1$ for some $\tau^{***} \in [\tau^*, \tau^{**}]$ since z is increasing for $t \geq \tau^*$. Suppose, by absurd, that there is no such τ^{***}. Then $z(t) < 1$ for all $t \in [\tau^*, \tau^{**}]$ which implies that $y_1(t) - z(t) > 3/2 - 1 = 1/2$. Then using this last inequality and (5.3), we conclude that $z'(t) \geq 1/2 - e^{-1}$.

Since $z(\tau^*) \in [-e^{-1}, e^{-1}]$, this implies that

$$\begin{aligned} z(\tau^{**}) &= z(\tau^*) + \int_{\tau^*}^{\tau^{**}} z'(t)dt \\ &\geq -e^{-1} + (\tau^{**} - \tau^*)\left(1/2 - e^{-1}\right) \\ &\geq -1 + \frac{4}{1-2e^{-1}}\left(\frac{1}{2} - e^{-1}\right) \\ &= -1 + \frac{2}{\frac{1}{2} - e^{-1}}\left(\frac{1}{2} - e^{-1}\right) \\ &= 1 \end{aligned}$$

which is an absurd. Therefore $z(t) \geq 1$ for all $t \geq \tau^{**}$. This proves conditions 1 and 2 of definition 5.1.

Note that $\|(y, v, z, \tau)'(t)\| \geq 1$ for all $t \geq 1$ so condition 4 of definition 5.1 is also satisfied. To show condition 3, recall that $|g^*(\Psi_k(w))| = 2$. Therefore from (5.1) for $t < \tau^*$ and from the previous analysis for $t \geq \tau^*$, it is possible to conclude that $|z(t)| \leq |g^*(\Psi_k(w))| + 1/2 = 5/2$ for all $t \geq 0$. This shows that if $Y = (y, v, z, \tau)$, then $\|Y(t)\|$ is bounded by an exponential boundary function on $\|\Psi_k(w)\|$ and by a polynomial on t because $\|y(t)\| \leq \Upsilon_1(\|\Psi_k(w)\|)\Upsilon_2(t)$ for all $t \geq 0$. Because $y'(t) = p(y(t))$ and, by proposition 2.14 $|z'(t)| \leq |y_1(t) - z(t)| \leq |y_1(t)| + 5/2$, we conclude that there are an exponential boundary function Υ_1^* and a polynomial Υ_2^* such that $\|Y'(t)\| \leq \Upsilon_1^*(\|\Psi_k(w)\|)\Upsilon_2^*(t)$ and, without loss of generality, we can assume that Υ_1^* and Υ_2^* are increasing functions. Now, since $\|Y'(t)\| \geq 1$, we have that

$$t \leq len_Y(0, t) \leq t \sup_{u \in [0,t]} \|Y'(u)\| \leq t\Upsilon_1^*(\|\Psi_k(w)\|)\Upsilon_2^*(t). \tag{5.4}$$

Define the function Π^* by $\Pi^*(|w|) \equiv \tau^{**}\Upsilon_1^*(|w|)\Upsilon_2^*(\tau^{**})$ which is an exponential boundary function in $\|\Psi_k(w)\| = |w|$, because τ^{**} is an exponential boundary function on $|w|$, Υ_2^* is a polynomial, and Υ_1^* is an exponential boundary function. Let t be such that $len_Y(0, t) \geq \Pi^*(|w|)$. Then, by (5.4)

$$t\Upsilon_1^*(|w|)\Upsilon_2^*(t) \geq \Pi^*(|w|) = \tau^{**}\Upsilon_1^*(|w|)\Upsilon_2^*(\tau^{**})$$

which implies that $t\Upsilon_2^*(t) \geq \tau^{**}\Upsilon_2^*(\tau^{**})$. Since Υ_2^* is increasing, this last condition is only true when $t \geq \tau^{**}$ which, by the previous analysis, implies that $|z(t)| \geq 1$, that is, the system has decided. This concludes the direct direction of the proof of theorem 5.3.

We will now proceed with the reverse direction of the proof of theorem 5.3. Assume that $L \in$ EAR. Apply the definition of the class EAR to get the parameters and polynomials d, p, q characterizing the dynamical system and an exponential boundary function Π which satisfies the third condition of the definition of the class. Let $w \in \Gamma^*$ and consider the following system

$$y(0) = q(\Psi_k(w)), \ y'(t) = p(y(t)).$$

We will show that we can decide in time exponential in $|w|$ whether $w \in L$ or not. Note that q is a polynomial with coefficients in $\mathbb{R}_P$ and $\Psi_k(w)$ is a rational number. Therefore $q(\Psi_k(w)) \in \mathbb{R}^d_P$. Finally, note that

$$\begin{aligned} PsLen_{y,p}(0,t) &= \int_0^t \Sigma p \max(1, \|y(u)\|)^{\deg(p)} du \\ &\leq t\Sigma p \max(1, \sup_{u\in[0,t]} \|y(u)\|^{\deg(p)}) \\ &\leq t\Sigma p \max(1, \sup_{u\in[0,t]} (\|y(0)\| + len_y(0,t))^{\deg(p)}) \\ &\leq t\, poly(len_y(0,t)) \\ &\leq poly(len_y(0,t)) \end{aligned}$$

where the last inequality holds because $len_y(0,t) \geq t$. We can now apply theorem 3.11 to conclude that we are able to compute $y(t) \pm 2^{-\mu}$ in time polynomial in t, μ and $len_y(0,t)$.

At this point some extra care is necessary. Indeed, the temptation is to use theorem 3.11 to compute the value of the curve $y(t)$ with some desired precision at time $\Pi(|w|)$. Nevertheless, it is possible that, at time $\Pi(|w|)$, the length of the solution could be already over exponential in $|w|$. Therefore, it is essential to use carefully the algorithm developed for the proof of theorem 3.11 and stop the computation as soon as the length of the solution is greater than $\Pi(|w|)$. This is possible due to the particular nature of the algorithm developed for the proof of theorem 3.11 in [36]. Let t^* be the time at which the algorithm stops. Then, the running time of the algorithm will be polynomial in t^*, μ and $len_y(0,t^*) \leq \Pi(|w|) + O(1)$. Finally, by definition of the EAR class, we have $t^* \leq len_y(0,t^*)$ and so, because $len_y(0,t^*) \leq \Pi(|w|)$ this algorithm has running time exponential in $|w|$ and polynomial in μ. By taking $\mu = \log 2$ we can obtain $\tilde{y}$ such that $\|y(t^*) - \tilde{y}\| \leq \frac{1}{2}$. By definition of Π we have that $y_1(t^*) \geq 1$ or $y_1(t^*) \leq -1$, so we can decide from $\tilde{y}$ if $w \in L$ or not. This finishes the proof of the theorem. ∎

5.2 Equivalence relation for PSPACE

Theorem 5.4 *A language $L \subseteq \Gamma^*$ belongs to* PSPACE *if and only if there is a polynomial ODE that $\hat{\Psi}$-decides it in polynomial space.*

Proof of theorem 5.4. The proof of theorem 5.4 is similar to the proof of theorem 4.14. First let us assume that $L \in$ PSPACE. Then there is a Turing machine M which recognizes the language in polynomial space. We can assume that M has one halting state, which is coded by the number m if the states are coded by the integers $1, \ldots, m$ as in the proof of theorem 4.7. A word w is assumed to be accepted by M if M halts and has the symbol coded by the number 1 in its tape, and is rejected if when the Turing machine halts the tape has only blanks and thus is coded by the integer 0. Proceeding similarly as in the proof of theorem 4.15, we can simulate the behavior of the Turing machine with a polynomial ODE which satisfies the condition of definition 5.2 if we take

$$\begin{aligned}
u_1(t) &= 10l_2(\sigma^{[b]}(y_3(t) - m + 1), 10) \cdot l_2(\sigma^{[b]}(y_6(t) - m + 1), 10) \\
u_2(t) &= 10l_2(\sigma^{[b]}(y_3(t) - m + 1), 10) \cdot l_2(\sigma^{[b]}(y_6(t) - m + 1), 10) \\
&\qquad \cdot (l_2(\sigma^{[b]}(y_1(t)), 10) - 1/2) \\
&= u_1(t) \cdot (l_2(\sigma^{[b]}(y_1(t)), 10) - 1/2)
\end{aligned}$$

where $b \in \mathbb{N}_0$ is chosen so that $\sigma^{[b]}(\eta) \leq 1/4$, σ and l_2 are defined by lemmas 4.11 and 4.10, y_1 codes the right part of the tape in the simulation given by the dynamical system, y_3 and y_6 code the state in each half-unit interval, and we assume that the system is reordered so that u_1 and u_2 are the first two components of the main variables of the ODE. Thus L is ($\hat{\Psi}$-)decidable by a polynomial ODE in polynomial space.

For the reverse direction, we can proceed as in the proof of theorem 4.17. The only difference is that, after we conclude that $\bar{y}_{(i),1} \geq 5/4$ (the computation has halted), we accept w if $\bar{y}_{(i),2} \geq 3/4$ and reject w if $\bar{y}_{(i),2} \leq -3/4$. The whole procedure takes polynomial space and thus if L is ($\hat{\Psi}$-)decidable by a polynomial ODE in polynomial space, then $L \in$ PSPACE. ■

As a logical consequence of the previous results of this chapter we can state the following theorem.

Theorem 5.5 (EXPTIME vs PSPACE) *We have* PSPACE $\subsetneq$ EXPTIME *if and only if there exists a language* $L \in$ EAR *such that no polynomial ODE* $\hat{\Psi}$*-decides* L *in polynomial space.*

Chapter 6

Generalization of the exponential result

As we have showed in chapter 3, the key intuition of splitting the dependence of the time and space boundaries from one single term, Υ, into the product of two separate components $\Upsilon_1\Upsilon_2$ with different behaviors has allowed us to capture the full power of exponential time computation with the suitable dynamical systems of polynomial differential equations. Since this procedure has successfully divided the part of the construction that may continue to depend polynomially on the input from the part that has to depend exponentially from the input in order to obtain the equivalence, and since this division process does not seem to explicitly depend on properties possessed only by exponential type of boundaries, it is natural to wonder if an equivalence can be obtained in the same way for other standard complexity classes in a straightforward manner. More precisely, if the second part of the time and space boundaries (Π_2 and Υ_2 in the original ATSE definition) is kept polynomial, it is natural to wonder which classes of functions can be used as first term (Π_1 and Υ_1) in order to characterize other classes from standard complexity theory.
Following every step of the proofs, starting from the basic properties and definitions of the GEVAL and ATSE classes, it is possible to see that four conditions are sufficient for the construction to hold.

In this section we extend the result of theorem 2.27 to other complexity classes. More precisely, we list which conditions the analog classes involved have to satisfy in order to repeat the simulation process already obtained for the polynomial case. First, let us define a version of the ATSE class in which the time and space boundaries are defined using functions from a set A of functions over the reals. More concretely, and in a similar manner to definition 3.1, we introduce the following definition

Definition 6.1 (ATSC(A)) *Let A be a class of functions from $\mathbb{R}_+$ to $\mathbb{R}_+$. Let $f \subseteq \mathbb{R}^n \to \mathbb{R}^m$. Let $\Pi_1, \Upsilon_1 : \mathbb{R}_+ \to \mathbb{R}_+ \in A$ and $\Pi_2, \Upsilon_2 : \mathbb{R}_+ \to \mathbb{R}_+$ be two polynomials. We say that $f \in \mathrm{ATSC}(A)$ if and only if there exist $d \in \mathbb{N}$, $p \in \mathbb{R}_P^d[\mathbb{R}^d]$ and $q \in \mathbb{R}_P^d[\mathbb{R}^n]$ such that for any $x \in \operatorname{dom} f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:*

- *$y(0) = q(x)$ and $y'(t) = p(y(t))$*

- $\forall \mu \in \mathbb{R}_+$ *if* $t \geq \Pi_1(\|x\|)\Pi_2(\mu)$ *then* $\|(y_1(t), y_2(t), y_m(t)) - f(x)\| \leq e^{-\mu}$
- $\|y(t)\| \leq \Upsilon_1(\|x\|)\Upsilon_2(t)$

As we already discussed, we need to be able to enforce closure by composition and by arithmetic operations for the class ATSC(A) to extend the result of theorem 2.27 to other complexity classes. Nevertheless, a more careful analysis of the details of the construction shows that the functions in A should satisfy other additional properties that are trivially shared by polynomials and exponentials, but that are not obvious for a generic class A. Before stating sufficient conditions that ensure closure by composition and by arithmetic operations for the class ATSC(A), we recall the notion of time-constructible functions [82].

Definition 6.2 (Time-constructible function) *Let $f : \mathbb{N} \to \mathbb{N}$ be a function. We call f time-constructible if there exists a Turing machine M which, given as an input a string 1^n, outputs the binary representation of $f(n)$ in time $O(f(n))$.*

We now present conditions that ensure that the result of theorem 2.27 can be extended to other complexity classes.

Definition 6.3 (Sufficient conditions) *Let A be a class of functions from $\mathbb{R}_+$ to $\mathbb{R}_+$ such that*

(1) *If $f, g \in A$ then there exists $h \in A$ such that $f \star g(x) \leq h(x)$ for every $x \in \mathbb{R}_+$, where $\star$ denotes any operator in the list of arithmetical operations: $(+, -, \times)$*

(2) *If p is polynomial and $f \in A$ then there exists $g \in A$ such that $p \circ f(x) \leq g(x)$ and $f \circ p(x) \leq g(x)$ for every $x \in \mathbb{R}_+$. Moreover, the identity operator belongs to A*

(3) *If $f \in A$, then there exists $g \in A$ such that $f(n) \leq g(n)$ for every $n \in \mathbb{N}$ and $g \in$ ATSC(A)*

(4) *If $f \in A$ then there exists $g : \mathbb{N} \to \mathbb{N}$ and $h \in A$ and such that $f(n) \leq g(n) \leq h(n)$ for every $n \in \mathbb{N}$ and g is a time-constructible function.*

The first condition enforces a form of closure for the main arithmetical operations which are of interest to us. The second condition provides enough elements so that we can have a variant of theorem 3.8 for the class ATSC(A) as well as allowing us to replace a polynomial in the proofs by a member of A when needed (polynomials may not belong to A). The third condition is sufficient to show that if a function is computed by a Turing machine in time $f|_{\mathbb{N}}$, where $f \in A$ (note that $f(\mathbb{N}) \subseteq [0, +\infty[$. Hence, although $f(n)$ might not belong to $\mathbb{N}$ when $n \in \mathbb{N}$, we can still say that a Turing machine computes in time $\leq f(n)$), then we may replace f by a function $g \in$ ATSC(A) which will play a role similar to the time bound e_M in the proof of theorem 3.9. The fourth and final condition is due to the fact that we do not have any assurances about the computability or complexity needed to compute elements of A. If some bound f of ATSC(A) has these problems, we need to be sure that we can replace it by a *well-behaved* bound g, which can be used to prove the reverse direction of 3.9, e.g. when computing the value t_w. Of course, the function g should not grow quicker than any element of A, hence there is the need to ensure that there is a function $h \in A$ which grows at least as quickly as g over the naturals.

We remark again that if $f \in A$ is a function such that $f(\mathbb{N}) \subseteq [0, +\infty[$, then we say that a Turing machine M computes a set of functions $\mathcal{F}$ in time $O(f(n))$ if there are some $c, n_0 \in \mathbb{N}$ such that if w is a word of length $n \geq n_0$, then M computes $g(w) \in \mathcal{F}$ in time $\leq cf(n)$. Now we can define $\mathrm{FTIME}(A) = \{g | g : \mathbb{N} \to \mathbb{N}$ is a function computable in time $O(f(n))$ for some $f \in A\}$. At this point we can finally state the following generalized equivalence theorem.

Theorem 6.4 (Generalized equivalence) *Let A be a class of functions that satisfies the conditions of definition 6.3. Then given a function $f : \Gamma^* \to \Gamma^*$, we have that $f \in \mathrm{FTIME}(A)$ if and only if f is emulable under* $\mathrm{ATSC}(A)$.

In the following section we will use this theorem to characterize the Grzegorczyk hierarchy with ODEs. In particular, we will also be able to characterize the class of elementary functions and the class of primitive recursive functions with ODEs.

6.1 Application to the Grzegorczyk hierarchy

We start this section by briefly recalling the definition of the Grzegorczyk hierarchy. The Grzegorczyk hierarchy, originally proposed by Andrzej Gregorczyk in 1953 in [83], is a hierarchy of classes of functions from the naturals to the naturals, defined recursively. To our specific purpose, the first two levels of the hierarchy are not relevant, including only trivial functions such as all addition and multiplication functions, which are obviously computable in polynomial time. The third level, which we indicate with the notation ξ^3 coincides with the set of all elementary functions. The definition of each level of the hierarchy for $n \geq 3$ involves the generator functions, G_n, whose definition is also recursive. Let $G_2 : \mathbb{N} \to \mathbb{N}$ be the exponential function $G_2(x) = 2^x$ and for $n \geq 2$ define: $G_{n+1}(x) = G_n^{[x]}(1)$, where the notation $G_n^{[x]}(1)$ stands for the iteration of the function G_n for x times evaluated on the value 1, i.e. $G_n^{[0]}(x) = x$ and $G_n^{[k+1]}(x) = G_n(G_n^{[k]}(x))$. Then, formally [84, definition VIII.8.12].

Definition 6.5 (Grzegorczyk Hierarchy) *For $n \geq 3$ the nth level of the hierarchy, ξ^n, is the smallest class of functions containing the zero function, the successor function, the projections, cut-off subtraction and G_{n-1} which is closed under composition, bounded sum and bounded product.*

It can be shown that each level is properly included in the next one, $\xi^{n-1} \subsetneq \xi^n \subsetneq \xi^{n+1}$, and that all together they constitute a hierarchy that satisfies $\bigcup\limits_{n \in \mathbb{N}} \xi^n = PR$, where PR is the set of primitive recursive functions. Moreover, for any function $f \in \xi^n$, there is some $m \in \mathbb{N}$ such that $f(x) \leq G_{n-1}^{[m]}(x)$ (see [84, theorem VIII.7.8]. Although this theorem is proved for the case of the elementary functions ξ^3, its proof generalizes to ξ^n for any $n \geq 2$). We also note [84, theorem VIII.8.14] that f belongs to ξ^n if and only if it is computable in time belonging to ξ^n. Combining the last two facts we conclude that f belongs to ξ^n if and only if it is computable in time bounded $G_{n-1}^{[m]}(x)$ for some $m \in \mathbb{N}$. We will use this last characterization to characterize ξ^n in the context of theorem 6.4 to avoid having to deal with bounded sums and products.

6.2 Analog characterization of each level

In this section our objective is to use ODEs to characterize the classes ξ^n, $n \geq 3$, defining the Grzegorczyk hierarchy, with the use of theorem 6.4.

However, a problem arises if one wants to use theorem 6.4 to characterize ξ^n for all $n \geq 3$. When characterizing the classes FP and FEXPTIME, we had to rely on the classes ATSP and ATSE which are defined using polynomial and exponential/polynomial bounds, respectively, which are *defined over* $\mathbb{R}_+ = [0, +\infty[$. In the polynomial and exponential cases, this was not problematic since polynomial and exponential functions over $\mathbb{N}$ admit a trivial extension to $\mathbb{R}$. This is not the case for the time bounds $G_{n-1}^{[m]}$ for the Grzegorczyk hierarchy, which are defined using iteration and hence do not admit a trivial extension to $\mathbb{R}$. To solve this problem, in this section we show how we can obtain functions in GVAL which essentially work as an extension of $G_{n-1}^{[m]}$. To achieve this purpose, we have to be able to iterate a function with ODEs over integers, since the definition of $G_{n-1}^{[m]}$ relies on the use of iterations, as mentioned earlier. Therefore we need to consider once again the construction used in chapter 4 to iterate functions with ODEs which is based on the ODE called targeting equation

$$y' = c(b-y)^3\phi(t) \tag{6.1}$$

that was already studied in [78], [79], [66]. We will now show how lemma 4.5 and the targeting ODE (4.3) can be used to obtain a function $f : \mathbb{R}^3 \to \mathbb{R}^2 \in$ GVAL such that the solution of the IVP $y' = f(t, y)$, $y(0) = (1, 1)$ satisfies $|y_1(t) - G_2(k)| \leq 1/4$ (i.e. $\left|y_1(t) - 2^k\right| \leq 1/4$) for all $t \in [k, k+1/2]$ and $k \in \mathbb{N}$, where $y = (y_1, y_2)$. To do so we will make use once again of the same error-correcting functions l_2 and σ defined in lemmas 4.10 and 4.11 respectively, considering again $\lambda_{1/4} = 0.4\pi - 1 \approx 0.2566371$. In the same way, the periodic function s involved is the one already introduced in (4.12), combined with l_2 to obtain W_0 as in (4.13), and we will repeat the main iterating method already used for the proof of theorem 4.9.

Let us now show how we can obtain a function $f : \mathbb{R}^3 \to \mathbb{R}^2 \in$ GVAL such that the solution of the IVP $y' = f(t, y)$, $y(0) = (1, 1)$ satisfies $\left|y_1(t) - 2^k\right| \leq 1/4$ for all $t \in [k, k+1/2]$ and $k \in \mathbb{N}$, where $y = (y_1, y_2)$. This will be done by iterating the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = 2x$ via the ODE

$$\begin{cases} y_1' = c(2\sigma(y_2) - y_1)^3\phi_1(t, y_1, y_2) \\ y_2' = c(\sigma(y_1) - y_2)^3\phi_2(t, y_1, y_2) \end{cases} \tag{6.2}$$

with initial condition $y_1(0) = y_2(0) = 1$, where

$$\begin{aligned} \phi_1(t, y_1, y_2) &= W_0(t, 16c((2\sigma(y_2) - y_1)^4 + 1) + 4) \\ \phi_2(t, y_1, y_2) &= W_0(-t, 16c((\sigma(y_1) - y_2)^4 + 1) + 4) \end{aligned} \tag{6.3}$$

and $c = 1000$. Its (ideal) behavior is depicted in fig. 6.1.

We note that ϕ_1 is essentially the function W_0 from (4.13), where the main change is made on the upper bound for W_0 on the time intervals $[k+1/2, k+1]$,

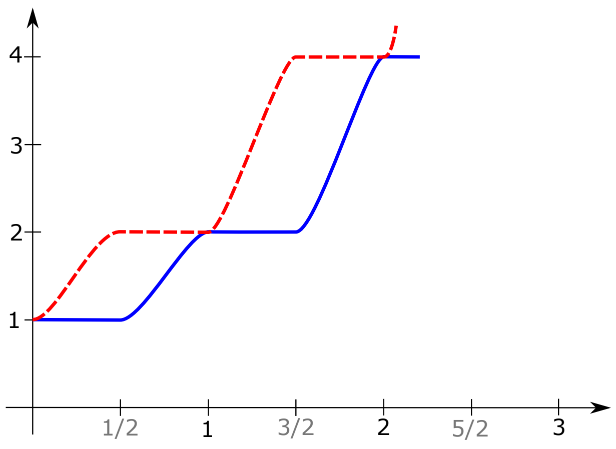


Figure 6.1: Iterating a function with an ODE. Here the red and blue graphs represent the ideal behavior of y_1 and y_2, respectively, in (6.2).

where $k \in \mathbb{Z}$. In this case, noting that $\left|x^3\right| \leq x^4 + 1$ for all $x \in \mathbb{R}$, we have that for any $t \in [k + 1/2, k + 1]$

$$\begin{aligned} |\phi_1(t, y_1, y_2)| &\leq \frac{1}{16c((2\sigma(y_2) - y_1)^4 + 1) + 4} \\ &< \frac{1}{16c(2\sigma(y_2) - y_1)^3} \end{aligned}$$

which implies that $|y_1'(t)| < 1/16$ whenever $t \in [k + 1/2, k + 1]$ for some $k \in \mathbb{Z}$. Furthermore, since $16c((2\sigma(y_2) - y_1)^4 + 1) + 4 \geq 4$, we conclude that $\int_k^{k+1/2} \phi_1(t, y_1, y_2)dt > 0.135 > 0$ and therefore that the first equation of (6.2) defines a targeting equation on the time interval $[0, 1/2]$ (or, more generally, on time intervals with the format $[k, k + 1/2]$ where $k \in \mathbb{Z}$) with targeting error $1/16$, since according to (4.2)

$$c = 1000 > \frac{1}{2\left(\frac{1}{16}\right)^2 0.135} > \frac{1}{2\left(\frac{1}{16}\right)^2 \int_0^{1/2} \phi_1(t, y_1, y_2)dt}.$$

Using a similar argument we conclude that $|y_2'(t)| < 1/16$ whenever $t \in [k, k + 1/2]$ for some $k \in \mathbb{Z}$ and that the second equation of (6.2) defines a targeting equation on the time interval $[1/2, 1]$ (or, more generally, on time intervals with the format $[k+1/2, k+1]$ where $k \in \mathbb{Z}$) with targeting error $1/16$.

Let us now analyze in more detail the ODE (6.2). When $t \in [0, 1/2]$, we have that $|y_2'(t)| \leq 1/16$, which further implies that $|y_2(t) - 1| \leq 1/32$ when $t \in [0, 1/2]$, since

$$\begin{aligned} |y_2(t) - y_2(0)| &= \left|\int_0^t y_2'(t)dt\right| \\ &\leq \int_0^t |y_2'(t)|\, dt \\ &\leq \left(\frac{1}{2} - 0\right)\frac{1}{16} = \frac{1}{32}. \end{aligned}$$

Now notice that, since $|y_2(t) - 1| \leq 1/32$, then

$$\begin{aligned} |2\sigma(y_2(t)) - 2 \cdot 1| &\leq 2\,|y_2(t) - 1| \\ &\leq 1/16. \end{aligned}$$

Hence, since the equation for y_1 in (6.2) defines a targeting equation with targeting error $\gamma = 1/16$, we get that $\left|y_1(1/2) - 2^1\right| < 1/16 + 1/16 = 1/8$. Now, on the next half-unit interval, we have that $|y_1'(t)| \leq 1/16$ which implies that $\left|y_1(t) - 2^1\right| < 1/8 + 1/32 = 5/32$ for all $t \in [1/2, 1]$. This implies that $\left|\sigma(y_1(t)) - 2^1\right| \leq \lambda_{1/4}\left|y_1(t) - 2^1\right| < 1/24$ for all $t \in [1/2, 1]$. Hence y_2 will become a targeting equation in the time interval $[1/2, 1]$ with targeting error $\gamma = 1/16$ and we will have $\left|y_2(1) - 2^1\right| < 1/24 + 1/16 < 1/8$. Now the procedure repeats itself in subsequent intervals. For example, when $t \in [1, 3/2]$, we will have that $|y_2'(t)| \leq 1/16$, which further implies that $\left|y_2(t) - 2^1\right| < 1/8 + 1/32 < 5/32$ when $t \in [1, 3/2]$. By a similar argument as in the previous case, we conclude that

$$\begin{aligned} \left|2\sigma(y_2(t)) - 2 \cdot 2^1\right| &\leq 2\lambda_{1/4}\left|y_2(t) - 2^1\right| \\ &\leq 2\lambda_{1/4}5/32 \\ &\leq 1/12. \end{aligned}$$

and since the equation for y_1 in (6.2) defines a targeting equation with targeting error $\gamma = 1/16$, we get that $\left|y_1(3/2) - 2^2\right| < 1/12 + 1/16 = 7/48$. On the next half-unit interval, we have that $|y_1'(t)| \leq 1/16$ which implies that $\left|y_1(t) - 2^2\right| < 7/48 + 1/32 = 17/96$ for all $t \in [3/2, 2]$. This implies that $\left|\sigma(y_1(t)) - 2^2\right| < 1/22$ for all $t \in [3/2, 2]$. Hence y_2 will become a targeting equation in the time interval $[3/2, 2]$ with targeting error $\gamma = 1/16$ and we will have $\left|y_2(2) - 2^2\right| < 1/22 + 1/16 < 1/8$. The procedure repeats itself on subsequent intervals and we conclude that

$$|y_2(t) - G_2(k)| \leq 1/4 \text{ for all } t \in [k, k + 1/2] \text{ and } k \in \mathbb{N}.$$

This result can be generalized as shown in the following theorem.

Theorem 6.6 *Given the function $G_n : \mathbb{N} \to \mathbb{N}$, $n = 2, 3, \ldots$, there is an IVP $y' = f_n(t, y)$, $y(0) = y_0$, where $f_n \in$ GVAL and $y_0 \in \mathbb{N}^l$, such that $|y_1(t) - G_n(k)| \leq 1/4$ for all $t \in [k, k + 1/2]$ and $k \in \mathbb{N}\backslash\{0\}$.*

Proof of theorem 6.6. The proof is done by induction on n. The base case was performed just before the proof of this theorem (note that y_2 in the above argument corresponds to y_1 in the context of this theorem).

Now we go to the induction step. Suppose that $y' = f_n(t, y)$, $y(0) = y_0$ satisfies the conditions of the theorem. Then we want to show that there is a system $y' = f_{n+1}(t, y)$ which simulates G_{n+1} in the sense mentioned in the theorem. Consider the ODE

$$\left\{\begin{array}{l} z' = f_n(\tau, y)\tau' = cf_n(\tau, y)(w + 1/4 - \tau)^3\theta_1(t, \tau, w) \\ \tau' = c(w + 1/4 - \tau)^3\theta_1(t, \tau, w) \\ w' = c(\sigma(z_1) - w)^3\theta_2(t, z_1, w) \end{array}\right. \tag{6.4}$$

with initial condition $z(0) = y_0$, $\tau(0) = 0$, $w(0) = 1$ where

$$\theta_1(t, \tau, w) = \phi(t, 16c((w + 1/4 - \tau)^4 + 1) + 4)$$
$$\theta_2(t, z_1, w) = \phi(-t, 16c((\sigma(z_1) - w)^4 + 1) + 4)$$

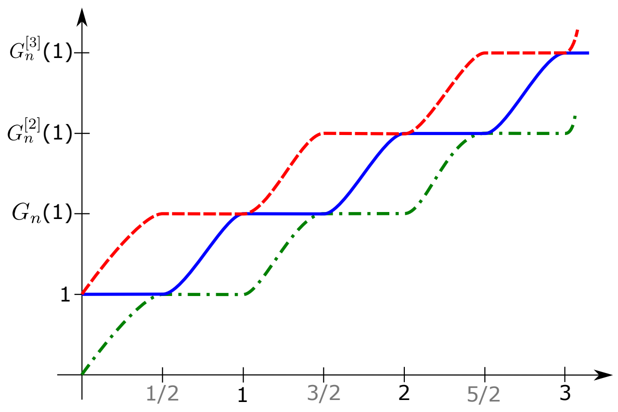


Figure 6.2: Iterating the function G_n with an ODE. Here the red, blue, and green graphs represent the ideal behavior of z_1, w, and τ, respectively, in (6.4).

and $c = 5000$. Its (idealized) behavior is depicted in fig. 6.2. We note that θ_1 and θ_2 behave like ϕ_1 and ϕ_2 of (6.3), respectively. By using arguments similar to those presented after (6.3), we conclude that: (i) $|\tau'(t)| \leq 1/16$ whenever $t \in [k + 1/2, k + 1]$ for some $k \in \mathbb{Z}$ and the equation for τ in (6.4) defines a targeting equation on time intervals with the format $[k, k + 1/2]$ where $k \in \mathbb{Z}$, with targeting error $1/32$ and (ii) $|w'(t)| \leq 1/16$ whenever $t \in [k, k + 1/2]$ for some $k \in \mathbb{Z}$ and that the equation for w in (6.4) defines a targeting equation on time intervals with the format $[k + 1/2, k + 1]$ where $k \in \mathbb{Z}$, with targeting error $1/32$.

We note also that, by construction $z(t) = y(\tau(t))$, where y is the solution of $y' = f_n(t, y)$, $y(0) = y_0$. Therefore the value of z only depends on the value of τ. We now have to analyze the behavior of τ and w. This simulation works again in half-unit time intervals. On the first half-unit interval, where $t \in [0, 1/2]$, we have that w is (almost) constant, since $|w'(t)| \leq 1/16$, which implies that $|w(t) - 1| \leq 1/32$ for all $t \in [0, 1/2]$. Therefore, since the equation governing the behavior of τ in this time interval $[0, 1/2]$ is a targeting equation, we conclude that $|\tau(1/2) - (1 + 1/4)| \leq 1/32 + 1/32 = 1/16$, since the targeting error is $1/32$. This gives that $\tau(1/2) \in [1 + 3/16, 1 + 5/16]$. Since $|\tau'(t)| \leq 1/16$ in the time interval $[1/2, 1]$ we conclude that $\tau(t) \in [1 + 1/8, 1 + 3/8] \subseteq [1, 3/2]$ for $t \in [1/2, 1]$. This implies, by the induction hypothesis, that $z(t) = y(\tau(t))$ is such that $|z_1(t) - G_n(1)| \leq 1/4$ for all $t \in [1/2, 1]$. Note that $|\sigma(z_1(t)) - G_n(1)| \leq \lambda_{1/4} |z_1(t) - G_n(1)| \leq \lambda_{1/4}/4 < 1/15$ for all $t \in [1/2, 1]$. Since the targeting error is $1/32$, we conclude that $|w(1) - G_n(1)| \leq 1/32+1/15$. Now the procedure repeats itself on the time interval $[1, 3/2]$. On this interval, we have $|w'(t)| < 1/16$ and thus $|w(t) - G_n(1)| \leq 1/32 + 1/15 + 1/32 = 31/240$ for all $t \in [1, 3/2]$. Therefore, the targeting equation for τ yields that $|\tau(3/2) - (G_n(1) + 1/4)| \leq 31/240 + 1/32 = 77/480$. Since $|\tau'(t)| \leq 1/16$ in the time interval $[3/2, 2]$, we have that $|\tau(t) - (G_n(1) + 1/4)| \leq 77/480 + 1/16 = 107/480 < 1/4$ and therefore $\tau(t) \in [G_n(1), G_n(1) + 1/2]$ for $t \in [3/2, 2]$. This implies, by the induction hypothesis, that $z(t) = y(\tau(t))$ is such that $|z_1(t) - G_n(G_n(1))| = |z_1(t) - G_{n+1}(2)| \leq 1/4$ for all $t \in [3/2, 2]$. Note that $|\sigma(z_1) - G_{n+1}(2)| \leq \lambda_{1/4} |z_1(t) - G_{n+1}(2)| \leq \lambda_{1/4}/4 < 1/15$ for all $t \in [3/2, 2]$. Considering the targeting equation for w in $[3/2, 2]$, we conclude that $|w(2) - G_{n+1}(2)| \leq 1/32 + 1/15$ since the targeting error is $1/32$. Notice again that, for all $t \in [2, 5/2]$, since $|w'(t)| < 1/16$, we have $|w(t) - G_{n+1}(2)| \leq 1/32 + 1/15 + 1/32 = 31/240 < 1/4$. By repeating this procedure on subsequent intervals and by considering w as the variable y_1 of the statement of the theorem, we conclude the desired result. ■

We now know from the previous theorem that each function G_n admits an GVAL-extension t_n in the sense of theorem 6.6. In other words, there is some GVAL function t_n such that $|t_n(t) - G_n(k)| \leq 1/4$ for all $t \in [k, k + 1/2]$ and $k \in \mathbb{N} \backslash \{0\}$.

Definition 6.7 *For each $n \geq 3$, we define the class $\mathbb{T}^n$ as the smallest class of functions $f : \mathbb{R} \to \mathbb{R}$ containing t_{n-1}, the identity, $\mathbb{R}_P$, and which is closed under sum, difference, product, and composition for $n \geq 3$.*

Note that this definition of $\mathbb{T}^n$ implies that condition 1 and 2 in the list of definition 6.3 are immediately satisfied (for condition 2 remarks that $\mathbb{T}^n$ includes t_2 which grows exponentially fast, and hence which grows more quickly than any polynomial). Furthermore, it is also not difficult to see that condition 4 is satisfied. Indeed, we have that $t_{n-1}(k) \leq G_{n-1}(k) + 1$ and since ξ^n is closed under composition and arithmetic operations, this shows that any function in $\mathbb{T}^n$ is dominated, over the naturals, by a function in ξ^n (note that all function in ξ^n are time-constructible). Reciprocally, if $f \in \xi^n$, then there is some $m \in \mathbb{N}$ such that $f(k) \leq G_{n-1}^{[m]}(k) \leq t_{n-1}(\ldots t_{n-1}(k) + 1 \ldots) + 1$, where $t_{n-1} + 1$ is composed m times. Since $(t_{n-1} + 1) \circ (t_{n-1} + 1) \circ \ldots \circ (t_{n-1} + 1) \in \mathbb{T}^n$, we conclude that condition 4 is satisfied. Moreover, this also shows the following lemma.

Lemma 6.8 FTIME$(\mathbb{T}^n)$ = FTIME$(\xi^n) = \xi^n$.

It is then left to prove condition 3, which requires that given any function $f \in \xi^n$ there is a function $g \in \mathbb{T}^n$ such that $f \leq g$ and $g \in$ ATSC$(\mathbb{T}^n)$, for all $n \geq 3$. We will now show that this condition is satisfied, in a multi-step argument.

From the above argument, we have just showed by means of the above proof of theorem 6.6 that each t_n is generable, meaning that $t_n \in$ GVAL for each $n \geq 3$. We now show that (i) we have $\mathbb{T}^n \subseteq$ GVAL$(\mathbb{T}^n)$, where

$$\text{GVAL}(\mathbb{T}^n) = \bigcup_{g \in \mathbb{T}^n} \text{GVAL}(g),$$

and (ii) GVAL$(\mathbb{T}^n) \subseteq$ ATSC$(\mathbb{T}^n)$. This will show condition 3 of definition 6.3, since any function $f \in \xi^n$ is bounded by a function in $\mathbb{T}^n$, as we have already seen.

To show condition (i), we first present the two following lemmas taken from [51, Lemma 24, Corollary 26].

Lemma 6.9 ([51], Arithmetic on bounded generable functions)
Let $d, l, n, m \in \mathbb{N}$, $sp, \overline{sp} : \mathbb{R}_+ \to \mathbb{R}_+$, $f :\subseteq \mathbb{R}^d \to \mathbb{R}^n \in$ GVAL(sp) and $g :\subseteq \mathbb{R}^l \to \mathbb{R}^m \in$ GVAL$(\overline{sp})$. Then

- $f + g, f - g \in \mathrm{GVAL}(sp + \overline{sp})$ *over* $\operatorname{dom} f \cap \operatorname{dom} g$ *if* $d = l$ *and* $n = m$
- $f \cdot g \in \mathrm{GVAL}(\max(sp, \overline{sp}, sp \cdot \overline{sp}))$ *if* $d = l$ *and* $n = m$
- $f \circ g \in \mathrm{GVAL}(\max(\overline{sp}, sp \circ \overline{sp}))$ *if* $m = d$ *and* $g(\operatorname{dom} g) \subseteq \operatorname{dom} f$.

Lemma 6.10 ([51], Generable functions are closed under ODE) *Let* $d \in \mathbb{N}$*,* $J \subseteq \mathbb{R}$ *an interval,* $sp, \overline{sp} : \mathbb{R}_+ \to \mathbb{R}_+$*,* $f :\subseteq \mathbb{R}^d \to \mathbb{R}^d \in \mathrm{GVAL}(sp)$*,* $t_0 \in \mathbb{R}_P \cap J$ *and* $y_0 \in \mathbb{R}_P^d \cap \operatorname{dom} f$*. Assume there exists* $y : J \to \operatorname{dom} f$ *satisfying for all* $t \in J$*:*

- $y(t_0) = y_0$
- $y'(t) = f(y(t))$
- $\|y(t)\| \leq \overline{sp}(t)$

Then $y \in \mathrm{GVAL}(\max(\overline{sp}, sp \circ \overline{sp}))$ *and is unique.*

Since lemmas 6.9 and 6.10 show closure of $\mathrm{GVAL}(\mathbb{T}^n)$ under the operations used to define $\mathbb{T}^n$, to show (i), i.e. that $\mathbb{T}^n \subseteq \mathrm{GVAL}(\mathbb{T}^n)$ it is enough to show that $t_n \in \mathrm{GVAL}(\mathbb{T}^{n+1})$ (note that the identity and all elements of $\mathbb{R}_P$ belong to $\mathrm{GPVAL} \subseteq \mathrm{GVAL}(\mathbb{T}^n)$).

First, define $\mathbb{T}^2 = \{p \mid p \in poly\}$ as the class of polynomials over $\mathbb{R}_P$. We show that the above proof of theorem 6.6 implies that $f_n \in \mathrm{GVAL}(\mathbb{T}^2)$ and $t_n = y_1 \in \mathrm{GVAL}(\mathbb{T}^{n+1})$ for each $n \geq 2$, where f_n and y_1 are defined in the statement of this theorem. This result can be showed by induction on n thanks to the closure by composition of each class $\mathbb{T}^n$.

To show that $f_n \in \mathrm{GVAL}(\mathbb{T}^2)$ and $t_n = y_1 \in \mathrm{GVAL}(\mathbb{T}^{n+1})$ for each $n \geq 2$, we proceed by induction. For the base case $n = 2$, consider f_2 as defined in (6.2) and note that $f_2 \in \mathrm{GVAL}(\mathbb{T}^2) = \mathrm{GPVAL}$ since all the right-hand terms in the system (6.2) belong to GPVAL. Moreover, note that, from the arguments which follow (6.2), the norm of the solution y of that dynamical system is bounded by a function $sp_2 \in \mathbb{T}^3$, where $sp_2(k) = t_2(k+1)$. Therefore, since $\mathbb{T}^2 \subset \mathbb{T}^3$ and each class is closed by composition, applying lemma 6.10 above yields $y \in \mathrm{GVAL}(\mathbb{T}^3)$ and, in particular, $t_2 = y_1 \in \mathbb{T}^3$. This proves the base case.

Let us now assume that $f_n \in \mathrm{GVAL}(\mathbb{T}^2)$ and that $t_n \in \mathrm{GVAL}(\mathbb{T}^{n+1})$. We now want to show that $f_{n+1} \in \mathrm{GVAL}(\mathbb{T}^2)$ and $t_{n+1} \in \mathrm{GVAL}(\mathbb{T}^{n+2})$. First we note that f_{n+1} is defined as the function used in the right-hand side of the ODE (6.4) and applied to the variables z, τ, and w. Since f_{n+1} is built using arithmetic operations, elements of $\mathbb{R}_P$, and the functions $f_n, \theta_1, \theta_2, \sigma \in \mathrm{GVAL}(\mathbb{T}^2)$, we conclude by lemma 6.9 that $f_{n+1} \in \mathrm{GVAL}(\mathbb{T}^2)$. Also, from the analysis done in the proof of theorem 6.6, we conclude that the solution $x(t)$ of (6.4) is bounded by $t_{n+1}(t+1) \in \mathrm{GVAL}(\mathbb{T}^{n+2})$, which shows by lemma 6.10 that x and hence t_{n+1} belongs to $\mathrm{GVAL}(\mathbb{T}^{n+2})$.

At this point we have shown that $t_n \in \mathrm{GVAL}(\mathbb{T}^{n+1})$ and hence we have proved (i). Now the last element left is condition (ii), which follows from the following theorem, that is a variant of theorems 2.5 and 3.6, but now applied to $\mathbb{T}^n$.

Theorem 6.11 *If* $f \in \mathrm{GVAL}(\mathbb{T}^n)$ *has a star domain with a generable vantage point, then we have* $f \in \mathrm{ATSC}(\mathbb{T}^n)$*.*

We note that (necessarily univariate) functions of $\mathrm{GVAL}(\mathbb{T}^{n+1})$ are always defined at any point of $\mathbb{R}_+ = [0, +\infty[$. This implies that such functions have a vantage point over this domain (e.g. 0 or 1 may be used as vantage points). Hence, we can conclude from theorem 6.11 that if $f \in \mathrm{GVAL}(\mathbb{T}^n)$, then $f \in \mathrm{ATSC}(\mathbb{T}^n)$, which immediately implies condition 3 of definition 6.3. To prove theorem 6.11, we need another result from [51, Proposition 28].

Lemma 6.12 *Let $sp : \mathbb{R}_+ \to \mathbb{R}_+$, $f \in \mathrm{GVAL}(sp)$. There exists a polynomial $q : \mathbb{R} \to \mathbb{R}$, with coefficients in $\mathbb{R}_P$, such that for any $x_1, x_2 \in \operatorname{dom} f$ then $\|f(x_1) - f(x_2)\| \leq \|x_1 - x_2\|\, q(sp(\max(\|x_1\|, \|x_2\|)))$.*

This lemma tells us that each function in $\mathrm{GVAL}(sp)$ has a modulus of continuity expressed by the function $q \circ sp$ where q is a polynomial; therefore, since $\mathrm{GVAL}(\mathbb{T}^n)$ is closed by composition with polynomials, this implies that each function in $\mathrm{GVAL}(\mathbb{T}^n)$ has modulus of continuity in $\mathbb{T}^n$. At this point it is straightforward to check that the proof already performed for theorem 3.6 for the exponential case can be repeated identical for the case in which the boundary sp belongs to $\mathbb{T}^n$, and consequently lead to the desired result.

Hence, the classes $\mathbb{T}^n$ defined above satisfy all the four conditions in the list of definition 6.3, and therefore theorem 6.13 holds. We then have the following result.

Theorem 6.13 (Analog characterization of the Grzegorczyk hierarchy) *Let $n \in \mathbb{N}$. Let $f : \Gamma^* \to \Gamma^*$, then $f \in \xi^n$ if and only if is emulable under $\mathrm{ATSC}(\mathbb{T}^n)$.*

Note that, because of what we discussed in section 6.1 when we recalled the definition of the Grzegorczyk hierarchy, theorem 6.13 above naturally implies an analog characterization of the class of elementary functions and the class of primitive recursive functions by means of polynomial ODEs.

Chapter 7

Complex square root

In this chapter we are going to analyze the problem of computing single-valued, analytic branches of the square root function over some class of complex domains. In particular, this study is focused on describing the upper complexity bound of such a problem in the context of complexity theory of real functions of Ko [41]. The work presented in this chapter is based on reference [70] where the authors described algorithms that could provide complexity bounds for computing the square root function and the logarithm function over simply connected domains. Finding single-valued, analytic branches of a multi-valued function defined on a simply connected domain is a fundamental problem in computational complex analysis [85]. Here we limit our attention to simply connected complex domains S whose boundary is a polynomial time computable Jordan curve. Let us consider two fixed dyadic points on the complex plane, $\mathbf{z}$ and $\mathbf{a}$ and a simply connected domain S whose boundary is a polynomial-time computable Jordan curve. We assume that ∂S is represented by a polynomial time computable function $f : [0, 1] \rightarrow \mathbb{C}$ and also use f to denote the image of f. The problem we are concerned about is to be able to compute the complex square root on the domain S. More precisely, the problem as taken from [70] is defined to be the following one.

Definition 7.1 (Complex square root problem) *Let $\mathbf{z_0}$ be a point in ∂S. Given two points $\mathbf{z} \in S$ and $\mathbf{a} \in \mathbb{C} - \bar{S}$, compute $g_1(\mathbf{z} - \mathbf{a})/g_1(\mathbf{z_0} - \mathbf{a})$, where g_1 is an arbitrary single-valued, analytic branch of $\sqrt{\mathbf{z}}$ on domain $S - \mathbf{a}$.*

In the above definition with $\bar{S} = S \cup \partial S$ we indicate the closure of the domain S and with $S - \mathbf{a}$ we refer to the domain $\{\mathbf{w} - \mathbf{a} | \mathbf{w} \in \mathbf{S}\}$. It is easy to see that the imaginary part of $g_1(\mathbf{z} - \mathbf{a})/g_1(\mathbf{z_0} - \mathbf{a})$ depends on how many times a path in S from $\mathbf{z_0}$ to $\mathbf{z}$ must wind around the point $\mathbf{a}$. Indeed, given a domain $T \subseteq \mathbb{C} - \bar{S}$, if we introduce a function $h_S : \bar{S} \times T \rightarrow \mathbb{R}$ that we call *continuous argument function* such that (i) for any point $\mathbf{a} \in T$, $h_S(\mathbf{z}, \mathbf{a})$ is continuous (ii) $h_S(f(0), \mathbf{a}) = 0$ for any $\mathbf{a} \in T$ and (iii) $2\pi h_S(\mathbf{z}, \mathbf{a}) = \theta_1 - \theta_2$ for some $\theta_1 \in arg(\mathbf{z} - \mathbf{a})$ and some $\theta_2 \in arg(f(0) - \mathbf{a})$ then we have [70].

Theorem 7.2 *The complex square root problem on S introduced by definition 7.1 is polynomial time solvable if and only if the function $(h_S(\mathbf{z}, \mathbf{a}) \mod 2)$ is polynomial time computable.*

Proof of theorem 7.2. For any single valued, analytic branch f_1 of $\sqrt{\mathbf{z}}$ on $S - \mathbf{a}$ we have $f_1(\mathbf{z} - \mathbf{a})/f_1(\mathbf{z_0} - \mathbf{a}) = (\sqrt{\mathbf{z} - \mathbf{a}}/\sqrt{\mathbf{z_0} - \mathbf{a}})e^{h_S(\mathbf{z},\mathbf{a})\pi i}$; but $\sqrt{\mathbf{z} - \mathbf{a}}/\sqrt{\mathbf{z_0} - \mathbf{a}}$ is polynomial time computable and we know that $e^{2\pi i} = 1$. Therefore theorem 7.2 holds. ■

Notice that, according to the relationship between h_S and the multi-valued function $arg(\mathbf{z})$, the fractional part of $(h_S(\mathbf{z}, \mathbf{a}) \bmod 2)$ is trivially polynomial time computable. The integer part instead is dependent on how many times a path in S from $\mathbf{z_0}$ to $\mathbf{z}$ must wind around the point $\mathbf{a}$ and thus computing this quantity is closely related to the problem of computing the winding number for a domain S and is dependent on the computational complexity of the domain S itself. There are a few more helpful assumptions that we can make on the points $\mathbf{z}$ and $\mathbf{a}$. Without losing generality, we may assume the point $\mathbf{a}$ to belong to a bounded domain T. To see this suppose all $\mathbf{z} \in S$ have $|\mathbf{z}| < m$ for some $m > 0$. Then for any point $\mathbf{a}$ with $|\mathbf{a}| \geq m + 1$ the imaginary part of $h_S(\mathbf{z}, \mathbf{a})$ is just a direct angle from the half line $\mathbf{az_0}$ to the half-line $\mathbf{az}$, therefore is trivially polynomial time computable. Thus, as far as time complexity is concerned, we may safely assume that both $\mathbf{a}$ and $\mathbf{z}$ are bounded. Moreover, to avoid any problem connected with the computability properties of functions in regions close to the border of the domain S, we allow our algorithm to commit mistakes if the points $\mathbf{a}$ and $\mathbf{z}$ are too close to the boundary ∂S. To our analysis, this will be implemented by requiring the property $d(\mathbf{z}, \partial S) > 2^{-n}$ and $d(\mathbf{a}, \partial S) > 2^{-n}$ for some n, where d represents the distance function, i.e. $d(\mathbf{z}, \partial S) = \inf\{|\mathbf{z} - \mathbf{z}'| \,|\mathbf{z}' \in \partial S\}$. This approach is a generalization of the notion of polynomial time recognizable sets introduced by Chou and Ko in [71] for the study of the membership problem of two dimensional domains. It follows a more precise definition of this approach.

Definition 7.3 (Function computable on a simply connected domain) *Let S be a bounded, simply connected domain whose boundary ∂S is a computable Jordan curve. A function $f : S \to \mathbb{C}$ is computable on domain S if there exists an oracle Turing machine M such that for any oracles (ϕ, ψ) representing a complex number $\mathbf{z} \in S$ we have $\left|M^{\phi,\psi}(n) - f(\mathbf{z})\right| \leq 2^{-n}$ for all inputs $n > 0$ whenever $d(\mathbf{z}, \partial S) > 2^{-n}$.*

Furthermore, f is polynomial time computable on domain S if f is computable on S by an oracle Turing machine that operates in polynomial time.

Since we have connected the problem of definition 7.1 to the function $h_S(\mathbf{z}, \mathbf{a})$ by means of theorem 7.2, to study its complexity it is enough to study the complexity of computing $h_S(\mathbf{z}, \mathbf{a})$. Intuitively, one direct method to compute the function $h_S(\mathbf{z}, \mathbf{a})$ could be given by two main operations

- Let L be the half-line going in the direction from $\mathbf{a}$ to $\mathbf{z}$. Then find a real number $t_0 \in [0, 1]$ such that $f(t_0)$ lies on L and the line segment $\overline{\mathbf{z}f(t_0)}$ lies entirely in $\bar{S}$

- Compute $h_S(f(t_0), \mathbf{a})$ by integration, then let $h_S(f(t_0), \mathbf{a}) = h_S(\mathbf{z}, \mathbf{a})$.

However, this method presents some hidden complications and cannot be followed in a straightforward manner. Indeed, t_0 can in principle be a non-computable real number, making the first point of the above method impossible to perform. In [70] the authors found a clever way to overcome this obstacle. The key intuition is noticing that for solving problem 7.1 the value of t_0 is not needed, as long as it is possible to compute $h_S(f(t_0), \mathbf{a})$. So they designed an algorithm able to explore the boundary ∂S and finding suitable candidates t for t_0 that could be used to point out the correct value for $h_S(f(t_0), \mathbf{a})$. The complexity of this particular algorithm was showed to belong to the class P^{MP}, which is the class of sets decidable by polynomial time bounded Turing Machines working with an oracle in the class MP. Nevertheless, the authors also suggested that there could be room for improvement for this particular result, proposing an hypothetical complexity bound belonging to the class of $\mathrm{P}^{\bigoplus \mathrm{P}}$. Following this hint, here is presented an algorithm (based on the construction developed in [70]) that is able to compute the quantity $\lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor \mod 2$ in polynomial time just by looking once at the last bit of a function in #P. Therefore, this new algorithm demonstrates that the square root problem of definition 7.1 indeed belongs to $\mathrm{P}^{\bigoplus \mathrm{P}}$.

Before discussing our algorithm we recall the definitions of the standard complexity classes involved [86], [87].

Definition 7.4 (Complexity classes) *Define the following classes as*

- $\#\mathrm{P}$*: the class of functions that counts the number of accepting paths of nondeterministic polynomial time machines*
- $\bigoplus \mathrm{P}$*: the class of sets A for which there exists a nondeterministic polynomial time Turing machine M such that for all x, $x \in A$ if and only if there are an odd number of accepting paths for x in M; equivalently a set A is in $\bigoplus \mathrm{P}$ if there exists a function $G \in \#\mathrm{P}$ such that for all x, $x \in A$ if and only if $G(x) \mod 2 = 1$*
- MP_b*: the class of sets A for which there exists a function $G \in \#\mathrm{P}$ and a function $\phi \in \mathrm{FP}$ such that for all x, $x \in A$ if and only if the $\phi(x)$th bit in the b-ary representation of $G(x)$ is not zero, where b is an integer greater than* 1
- MP*: the union of* MP_b *over all* $b \geq 2$

We also recall two properties of the class #P that are useful to understand our result [86], [87].

Theorem 7.5 *(a) For any function $g : \{0,1\}^* \to \mathbb{N}$, g belongs to $\#\mathrm{P}$ if and only if there exists a set $A \subseteq \{0,1\}^* \in \mathrm{P}$ and a polynomial p such that for each $w \in \{0,1\}^*$, $G(w)$ is equal to the number of strings u of length $p(|w|)$ such that $\langle w, u \rangle \in A$*

(b) Assume that a function $F : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$ is polynomial time computable, where the output is written in binary form. Then the function $G : \{0,1\}^* \times 0^* \to \mathbb{N}$ defined by: $G(w_1, 0^n) = \sum_{|w_2|=n} F(w_1, w_2)$ belongs to $\#\mathrm{P}$.

7.1 Approximation of the curve

Consider the curve described by f and defining ∂S. To be able to correctly work with this curve, and compute certain quantities related to this curve (such as find intersections of the curve with specific straight lines in the complex plane) it is used here the same technique used in [70], where the curve is approximated by a collection of piecewise linear functions, up to some desired precision, following the core ideas of computable analysis. Let M be an oracle Turing machine that computes f in time p for some polynomial p. It follows that p is a modulus function of f. Let n be an integer such that $d(\mathbf{z}, \partial S) > 2^{-n}$ and $d(\mathbf{a}, \partial S) > 2^{-n}$. For each $0 \leq i \leq 2^{p(n)}$ let $s_i = i \cdot 2^{-p(n)}$ and $\mathbf{z_i} = M^{s_i}(n)$. Then, for any $0 \leq i \leq 2^{p(n)} - 1$ and any $t \in [s_i, s_{i+1}]$ we have $|f(t) - \mathbf{z_i}| \leq 2^{-n}$. Let f_n be the piecewise linear function with breakpoints $f_n(s_i) = \mathbf{z_i}$ for $i = 0, .., 2^{p(n)}$, and Γ_n be the image of f_n on $[0, 1]$. Then, Γ_n is an approximation of ∂S within an error of 2^{-n}. Note that Γ_n is not necessarily simple. Often in this work we will shortly refer to the segment $\overline{\mathbf{z}_{i-1}\mathbf{z}_i}$ as the ith segment of the approximation f_n, or as the segment i. For simplicity, we may assume that $f_n(0) = f(0)$ (this can be achieved by, for example, transforming the whole plane so that $f(0)$ becomes the origin). Without loss of generality, we may also assume that $\mathbf{z}_0$ does not lay on the line connecting the points $\mathbf{a}$ and $\mathbf{z}$. Indeed, if it is not the case, we can pick a point $s \in [0, 1]$ such that this condition is satisfied and then use the function $f_1(t) = f(s + t \mod 1)$ on $[0, 1]$ to represent ∂S.

We now define a segment L, a half-line V and two straight-lines L' and V'. Let L be the segment that connects $\mathbf{a}$ and $\mathbf{z}$ and L' the straight line containing L. Assume that the direction of L' is the one going from $\mathbf{a}$ to $\mathbf{z}$. Let V be the half-line starting from $\mathbf{a}$ passing through $\mathbf{z}_0$ and V' the straight-line containing V. Assume that the direction of V' is the one going from $\mathbf{a}$ to $\mathbf{z}_0$. Trivially, the line V' divide the plane in two half planes, the one on the left of V' is denoted as V'_l and the one on the right plus V' is denoted as V'_r. In the same way, the line L', divide the complex plane in two half planes, the one on the left of L' is denoted as L'_l and the one on the right plus L' is denoted as L'_r. This construction allows us to define two functions that we call *sign functions* and that will be crucial for our algorithm.

Definition 7.6 (Sign functions) *Let* $sngV_{f_n} : \{1, 2, ..., 2^{p(n)}\} \to \{-1, 0, 1\}$ *and* $sngL_{f_n} : \{1, 2, ..., 2^{p(n)}\} \to \{-1, 0, 1\}$ *be the functions defined as follows*

$$sngV_{f_n}(i) = \begin{cases} 1 \quad \textit{if } \mathbf{z}_i \in V'_l \textit{ and } \mathbf{z}_{i-1} \in V'_r \textit{ and } \overline{\mathbf{z}_{i-1}\mathbf{z}_i} \cap V \neq \emptyset \\ -1 \quad \textit{if } \mathbf{z}_i \in V'_r \textit{ and } \mathbf{z}_{i-1} \in V'_l \textit{ and } \overline{\mathbf{z}_{i-1}\mathbf{z}_i} \cap V \neq \emptyset \\ 0 \quad \textit{otherwise} \end{cases} \tag{7.1}$$

$$sngL_{f_n}(i) = \begin{cases} 1 \quad \textit{if } \mathbf{z}_i \in L'_l \textit{ and } \mathbf{z}_{i-1} \in L'_r \textit{ and } \overline{\mathbf{z}_{i-1}\mathbf{z}_i} \cap L \neq \emptyset \\ -1 \quad \textit{if } \mathbf{z}_i \in L'_r \textit{ and } \mathbf{z}_{i-1} \in L'_l \textit{ and } \overline{\mathbf{z}_{i-1}\mathbf{z}_i} \cap L \neq \emptyset \\ 0 \quad \textit{otherwise} \end{cases} \tag{7.2}$$

Notice that the function $sngV_{f_n}$ signals when there is an intersection between the half-line V and the approximated image of our curve, Γ_n, distinguishing with a plus or minus sign between two different types of intersections. We will informally refer to these two different types of intersections as the possible *directions of the intersection*, that intuitively can be seen as clockwise or counterclockwise. The same applies to $sngL_{f_n}$ that signals when there is an intersection between the segment L and Γ_n, taking in account the possible directions of the intersection. Every segment identified by an $i \in \{1, 2, ..., 2^{p(n)}\}$ with $sngV_{f_n}(i) \neq 0$ or

$sngL_{f_n}(i) \neq 0$ will be shortly named as a V-crossing segment or a L-crossing segment, respectively.

Definition 7.7 (V and L crossing segments) *For any $i \in \{1, 2, ..., 2^{p(n)}\}$ we say that segment i is a V-crossing segment (L-crossing segment) if and only if $sngV_{f_n}(i) \neq 0$ ($sngL_{f_n}(i) \neq 0$).*

Notice that, due to the definition of the two functions above, segments with an extreme point belonging to V_l' and the other belonging to V will be considered as V-crossing segments, and we will say that they *cross* V, where the meaning of the word *crossing* in this context has to be intended in the sense of this definition. The same consideration obviously applies to L-crossing segments. When $sngL_{f_n}(i)$ is not zero, and hence i is an L-crossing segment, there is a unique $s_i' \in [s_{i-1}, s_i]$ such that $\mathbf{b}_i \equiv f_n(s_i') \in \overline{\mathbf{z}_{i-1}\mathbf{z}_i} \cap L$; we call $\mathbf{b}_i$ an intersection point of $f_n([s_{i-1}, s_i])$ and L. The same consideration obviously applies to V-crossing segments. Moreover, we have the following lemma.

Lemma 7.8 *Assume that ∂S is polynomial-time computable. Then the functions $\phi_1(n, i) = sngL_{f_n}(i)$, $\phi_2(n, i) = s_i'$, $\phi_3(n, i) = \mathbf{b}_i$ (if they exist) are polynomial-time computable.*

Proof. Since $\mathbf{z}_0 = f(0)$, $\mathbf{a}$, $\mathbf{z}$ and $\mathbf{z}_i$ are all dyadic points, the function $sngL_{f_n}(i)$ is computable in polynomial-time by exact arithmetic. When $sngL_{f_n}(i) \neq 0$, the quantities s_i' and $\mathbf{b}_i$ are not necessarily dyadic but must be rational. We can compute, for any $k \geq p(2n)$ a dyadic $d_k \in \mathbb{D}_k$, such that $|d_k - s_i'| \leq 2^{-k}$, which implies that $|f_n(d_k) - f_n(s_i')| \leq 2^{-(2n+k-p(2n))}$ ■

Obviously, an identical lemma can be stated for the case of the function $sngV_{f_n}(i)$ and its intersection points.

The last missing element before being able to describe the algorithm is to set the correct *offset*, where with offset we mean the correct initial point from which the algorithm starts working. To do so, let us consider all the points of intersection between the segment L and the approximation of the border Γ_n. Since there is only a finite number of segments in the approximation, there is also a finite number of intersection points with L, let us say k, where we know that $k \leq 2^{p(n)}$. These points can be ordered according with the ordering of the segments of the approximation. This means that we can identify them with the notation $\mathbf{b}_1, \mathbf{b}_2, .., \mathbf{b}_k$, where $\mathbf{b}_1$ indicates the intersection point between L and the least $i \in \{1, 2, ..., 2^{p(n)}\}$ representing an L-crossing segment. We will shortly refer to $\mathbf{b}_1$ as the intersection point related to the *first* L-crossing segment, to $\mathbf{b}_2$ as the intersection point related to the *second* L-crossing segment, and so on. Then, we call i^* the first L-crossing segment, with intersection $\mathbf{b}_1$, and we consider that as the starting point of our algorithm. Notice that i^* is easily polynomial-time computable given n, $p(n)$ and the approximation of the curve Γ_n. The reason behind choosing such a point as a starting point instead of simply choosing $\mathbf{z}_0$ is that, on this first intersection, the border ∂S has never completed a full spin around the point $\mathbf{a}$ yet, and therefore this particular point has to be excluded from the counting method performed by the algorithm we are going to illustrate below.

7.2 The algorithm

Given a word $w \in \{0,1\}^*$ we denote its length with the notation $|w|$ as usual. Moreover, if $|w| = p(n)$ we indicate with i_w the nonnegative integer $i_w \leq 2^{p(n)}-1$ whose $p(n)-$bit binary representation is w. Consider now the function: $F : (S \cap \mathbb{D}^2) \times (T \cap \mathbb{D}^2) \times \{0,1\}^* \to \mathbb{N}$ defined as the function computed by the following algorithm

```
1. if $|w| \neq 2p(n)$ for any $n > 0$ then output 0.
2. if $|w| = 2p(n)$ for some $n > 0$, let $w = uv$ with $|u| = |v|$ then
       if $i_u \leq i^*$ then let $e_u = e_v = 0$
       if $i_u > i^*$ then let $e_u = |sngL_{f_n}(i_u)|$
         if $i^* < i_v \leq i_u$ then let $e_v = sngV_{f_n}(i_v) + 2$
         else let $e_v = 0$
3. output $e_u \cdot e_v$
```

It is clear that $F(\mathbf{z}, \mathbf{a}, w)$ is polynomial time computable and always nonnegative. At this point we can define a function $G : (S \cap \mathbb{D}^2) \times (T \cap \mathbb{D}^2) \times \{0\}^* \to \mathbb{N}$ by $G(\mathbf{z}, \mathbf{a}, 0^m) = \sum_{|w|=m} F(\mathbf{z}, \mathbf{a}, w)$. Then, G is in $\#P$ thanks to theorem 7.5. We have then the following theorem.

Theorem 7.9 $G(\mathbf{z}, \mathbf{a}, 0^{2p(n)}) \bmod 2 = \lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor \bmod 2$

Notice that if we call R the function $R : (S \cap \mathbb{D}^2) \times (T \cap \mathbb{D}^2) \times \{0\}^* \to \{0,1\}$ such that $R = G(\mathbf{z}, \mathbf{a}, 0^{2p(n)}) \bmod 2$, then from the above theorem combined with theorem 7.2 we obtain that there exists a Turing machine that solves the complex square root problem in polynomial time using as oracle a function R in $\bigoplus P$, and this satisfies our final goal.

It is left to prove theorem 7.9. To explain the proof, we need to look closely at the algorithm described above. First, notice that the only interesting case is the one in which the sum $\sum_{|w|=m} F(\mathbf{z}, \mathbf{a}, w)$ involves a word w of length $2p(n)$ such that $w = uv$ with $|u| = |v| = p(n)$, $i_u > i^*$ and $i^* < i_v \leq i_u$, since all the other cases generate summed terms that are equal to zero by definition of the algorithm. Therefore, we can write the interesting quantity to analyze as

$$\begin{aligned} G(\mathbf{z}, \mathbf{a}, 0^{2p(n)}) &= \sum_{\{|u|=p(n), i_u > i^*\}} \sum_{\{|v|=p(n), i^* < i_v \leq i_u\}} F(\mathbf{z}, \mathbf{a}, uv) \\ &= \sum_{i_u > i^*} |sngL_{f_n}(i_u)| \cdot \sum_{i^* < i_v \leq i_u} (sngV_{f_n}(i_v) + 2) \end{aligned} \tag{7.3}$$

First, we give a very brief, general description of the algorithm, after which we will start analyzing it in detail. Notice that the first factor of the sum is nonzero only when $sngL_{f_n}(i_u) \neq 0$, $i_u > i^*$ and this happens only for segments such that $\overline{\mathbf{z}_{i_u-1}\mathbf{z}_{i_u}} \cap L \neq \emptyset$ and $i_u > i^*$, which are segments that cross the segment L and follow segment i^*, related to the first L-crossing segment. So, we will have a term to sum for each of the intersections of the approximated curve

Γ_n with the L segment, excluding the first one. Since the L segment connects $\mathbf{a}$ to $\mathbf{z}$, this part of the algorithm takes care of computing the winding number only until the point $\mathbf{z}$, and not for the whole domain S. For each of this nonnegative, nonzero terms, the second factor of the above expression, $\sum\limits_{i_v \leq i_u} (sngV_{f_n}(i_v)+2)$, will sum quantities that will be relevant to us only for segments that cross the half-line V. Thanks to the condition $i^* < i_v \leq i_u$, this means that, for each i_u representing a L-crossing segment excluding the first one, we will sum as many relevant terms as many V-crossing segments we encounter starting from i^* before reaching the segment i_u. Notice that in this case, these terms will be summed with their relative sign, depending on the direction of the crossing segments. Notice also that, thanks to the $+2$ contribution in $\sum_{i^*<i_v \leq i_u}(sngV_{f_n}(i_v)+2)$, all the summed terms will always be nonnegative, no matter the sign assigned by $sngV_{f_n}$ to the segments. The segments that do not intersect V, instead of contributing as zero terms to the sum, they all generate an additional $+2$ to the sum. This is irrelevant, since what interests us at the end is the modulus 2 of the result.

7.3 Description of the algorithm and proof of the result

We can now proceed with a more precise description of the above.

Proof of theorem 7.9. We introduce here another notation that will be used extensively in the following analysis. For any intersection point $\mathbf{b}_q$ in the set of intersections of L-crossing segments $\{\mathbf{b}_1, \mathbf{b}_2, .., \mathbf{b}_k\}$, we indicate with the symbol $\lfloor \bar{h}_S(\mathbf{b}_q, \mathbf{a}) \rfloor$ the value of $\lfloor h_S(\mathbf{z}', \mathbf{a}) \rfloor$ where $\mathbf{z}' \in \bar{S}$ is such that $|\mathbf{b}_q - \mathbf{z}'| \leq 2^{-n+1}$. Notice that, thanks to the conditions on our approximation Γ_n, this quantity is well defined, and that there is no ambiguity in this definition, since $\lfloor \bar{h}_S(\mathbf{z}', \mathbf{a}) \rfloor$ has the same value for all the $\mathbf{z}' \in \bar{S}$ such that $|\mathbf{b}_q - \mathbf{z}'| \leq 2^{-n+1}$ due to the conditions $d(\mathbf{z}, \partial S) > 2^{-n}$ and $d(\mathbf{a}, \partial S) > 2^{-n}$.

At this point, notice that there always exists a point $\mathbf{b}_{k^*}$ in $\{\mathbf{b}_1, \mathbf{b}_2, .., \mathbf{b}_k\}$ such that $\lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor = \lfloor (\bar{h}_S(\mathbf{b}_{k^*}, \mathbf{a})) \rfloor$. Therefore, we reduce the problem of computing $\lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor$ to compute $\lfloor (\bar{h}_S(\mathbf{b}_{k^*}, \mathbf{a})) \rfloor$ instead, and we find a way to do so without having to compute the point $\mathbf{b}_{k^*}$, in a similar fashion to what has been done in [70] for the logarithm function. Notice that the number of intersections of the curve described by f with the segment L is always an odd number (since $\mathbf{a}$ is outside the domain S and $\mathbf{z}$ is inside the domain) and notice that this number cannot be less than the quantity $2\lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor + 1$. Nevertheless, simply counting the L-crossing segments is not enough to compute $\lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor$, since in general there can be intersections between L and the curve that do not represent a spin around $\mathbf{a}$. Moreover, since we are only able to deal with an approximation of the curve, Γ_n, some intersections with L may be present in the approximation and not in the curve f, or vice versa. The algorithm described above has a way to rule out those problematic situations, exploiting the fact that the quantity we are really interested to compute is $\lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor \bmod 2$ and not just $\lfloor (h_S(\mathbf{z}, \mathbf{a})) \rfloor$. Specifically what the above algorithm does is to compute, for each of the intersections $\mathbf{b}_k$ the value $\lfloor (\bar{h}_S(\mathbf{b}_k, \mathbf{a})) \rfloor$ and then summing all these values. In the final sum all the irrelevant terms will produce an even quantity, which we will indicate with E, that will add to $\lfloor(\bar{h}_S(\mathbf{b}_{k^*},\mathbf{a}))\rfloor$. Finally, taking the modulus 2 of the result, will cancel out the E term, leaving us with the solution to our problem, the quantity $\lfloor(\bar{h}_S(\mathbf{b}_{k^*},\mathbf{a}))\rfloor \bmod 2$.

For each of the intersection points $\{\mathbf{b}_2,..,\mathbf{b}_k\}$ (corresponding to each of the L-crossing segments i_u excluding the first one i^*), let us now analyze the role of the factor $\sum_{i^*<i_v\leq i_u}(sngV_{f_n}(i_v)+2)$. Let us pick a generic point $\mathbf{b}_q \in \{\mathbf{b}_2,..,\mathbf{b}_{k-1}\}$ and let us consider the portion of the curve f which is approximated by the segments starting from the L-crossing segment related to the intersection point $\mathbf{b}_q$, until the L-crossing segment related to the intersection point $\mathbf{b}_{q+1}$ (in increasing order according to the usual ordering). To be able to quickly refer to this entity, we will informally call it the $\mathbf{b}_q$ *portion of the curve* f in this analysis.

In the same spirit, we will refer as the collection of segments approximating the $\mathbf{b}_q$ portion of the curve f as the $\mathbf{b}_q$ *collection of segments.* By definition, we exclude the L-crossing segments related to the points $\mathbf{b}_q$ and $\mathbf{b}_{q+1}$ from the $\mathbf{b}_q$ collections of segments, so that there are no segments with nonzero $sngL_{f_n}$ in the collection.

Definition 7.10 ($\mathbf{b}_q$ collection of segments) *Let $\mathbf{b}_q \in \{\mathbf{b}_2,..,\mathbf{b}_{k-1}\}$ be an intersection point of the L-crossing segment indicated by number r, where $r \leq 2^{p(n)}$. Recall that, according to the notation described above, we called this segment the rth-segment, or segment r of the approximation Γ_n. We define the $\mathbf{b}_q$ collection of segments as the set of segments composed by segments $\{r+1, r+2, ..., g-1\}$ where segment g is the L-crossing segment that corresponds to the intersection point $\mathbf{b}_{q+1}$ in our ordering of the intersection points.*

Definition 7.11 ($\mathbf{b}_q$ portion of the curve) *Let $\mathbf{b}_q \in \{\mathbf{b}_2,..,\mathbf{b}_{k-1}\}$ be an intersection point of one L-crossing segment indicated by number r, where $r \leq 2^{p(n)}$. We define the $\mathbf{b}_q$ portion of the curve as the curve composed by points that satisfy $\{$ $\mathbf{x} \in \partial\mathbf{S}$ such that $|\mathbf{x}-\mathbf{y}| \leq 2^{-n}$ for some point $\mathbf{y}$ in the $\mathbf{b}_q$ collection of segments $\}$.*

To understand these definitions it is necessary not to make confusion between the ordering of the segments composing the approximation Γ_n, $\{1,2,..,2^{p(n)}\}$ and the ordering given to the L-crossing segments and their relative intersection points, $\{\mathbf{b}_1,\mathbf{b}_2,..,\mathbf{b}_{k-1}\}$. Indeed, notice that each L-crossing segment can be identified with a specific number from both the different ordering. As an example, in the definitions above, segment r can also be identified as the qth L-crossing segment, or the L-crossing segment number q, or by means of its intersection point $\mathbf{b}_q$. We will explicitly make clear to which of the ordering we are referring to whenever it will be needed in the rest of this analysis.

At this point, there are three possible situations we need to distinguish:

1. $\lfloor(\bar{h}_S(\mathbf{b}_q,\mathbf{a}))\rfloor = \lfloor(\bar{h}_S(\mathbf{b}_{q+1},\mathbf{a}))\rfloor$
2. $\lfloor(\bar{h}_S(\mathbf{b}_q,\mathbf{a}))\rfloor = \lfloor(\bar{h}_S(\mathbf{b}_{q+1},\mathbf{a}))\rfloor + 1$
3. $\lfloor(\bar{h}_S(\mathbf{b}_q,\mathbf{a}))\rfloor + 1 = \lfloor(\bar{h}_S(\mathbf{b}_{q+1},\mathbf{a}))\rfloor$

notice that no other situation is possible, since $\mathbf{b}_q$ and $\mathbf{b}_{q+1}$ are two successive intersections in the ordering of all L-crossing segments. Let us analyze separately these three situations:

1) if $\lfloor(\bar{h}_S(\mathbf{b}_q, \mathbf{a}))\rfloor = \lfloor(\bar{h}_S(\mathbf{b}_{q+1}, \mathbf{a}))\rfloor$ then the $\mathbf{b}_q$ portion of the curve f does not spin around the point $\mathbf{a}$. To see that this statement is true, just notice that the approximation Γ_n closely follows the border line described by f. Moreover, in this situation we will have $\sum_{\{i_v \leq i_u | i_v \in \mathbf{b}_q c.s.\}}(sngV_{f_n}(i_v)+2) = E$, where the abbreviation *c.s* represents the fact that we are summing over all the segments belonging to the $\mathbf{b}_q$ collection of segments. To see that this statement is true, notice that, since the $\mathbf{b}_q$ portion of the curve f does not spin around the point $\mathbf{a}$, in the $\mathbf{b}_q$ collection of segments we can only have pairs of V-crossing segments, with opposite sign according to the definition of the function $sngV_{f_n}$. Therefore, summing all these terms, the resulting sum will produce an even quantity E.

2) if $\lfloor(\bar{h}_S(\mathbf{b}_q, \mathbf{a}))\rfloor + 1 = \lfloor(\bar{h}_S(\mathbf{b}_{q+1}, \mathbf{a}))\rfloor$ then the $\mathbf{b}_q$ portion of the curve f does spin in one direction around the point $\mathbf{a}$ (whether this spinning direction is clockwise or anticlockwise is irrelevant to the algorithm). To see that this statement is true, just notice that the approximation Γ_n closely follows the border line described by f. Moreover, in this situation we will have $\sum_{\{i_v \leq i_u | i_v \in \mathbf{b}_q c.s.\}}(sngV_{f_n}(i_v)+2) = E+1$, where we are summing over all the segments belonging to the $\mathbf{b}_q$ collection of segments. To see that this statement is true, notice that, since the $\mathbf{b}_q$ portion of the curve f does spin around the point $\mathbf{a}$, in the $\mathbf{b}_q$ collection of segments we can only have one V-crossing segment such that its $sngV_{f_n}$ function equals 1, plus pairs of V-crossing segments with opposite sign according to the definition of the function $sngV_{f_n}$. Therefore, summing all these terms, the resulting sum will produce in the final term $+1$ plus an even quantity E.

3) if $\lfloor(\bar{h}_S(\mathbf{b}_q, \mathbf{a}))\rfloor = \lfloor(\bar{h}_S(\mathbf{b}_{q+1}, \mathbf{a}))\rfloor + 1$ then the $\mathbf{b}_q$ portion of the curve f does spin around the point $\mathbf{a}$ in the direction opposite to the one of case 2) (whether this spinning direction is clockwise or anticlockwise is irrelevant to the algorithm). To see that this statement is true, just notice that the approximation Γ_n closely follows the border line described by f. Moreover, in this situation we will have $\sum_{\{i_v \leq i_u | i_v \in \mathbf{b}_q c.s.\}}(sngV_{f_n}(i_v)+2) = E-1$, where we are summing over all the segments belonging to the $\mathbf{b}_q$ collection of segments. To see that this statement is true, notice that, since the $\mathbf{b}_q$ portion of the curve f does spin around the point $\mathbf{a}$, in the $\mathbf{b}_q$ collection of segments we can only have one V-crossing segment such that its $sngV_{f_n}$ function equals -1, plus pairs of V-crossing segments with opposite sign according to the definition of the function $sngV_{f_n}$. Therefore, summing all these terms, the resulting sum will produce in the final term -1 plus an even quantity E.

We can now describe the meaning of the total sum in (7.3), in a more precise way. Let us consider two generic, successive L-crossing segments (excluding the first one i^*), say the segment related to the point $\mathbf{b}_q$ and the segment related to the point $\mathbf{b}_{q+1}$. These segments will produce two nonzero terms in the total sum. Notice that, if we are in case 1) described above, then summing these two terms will result to an even quantity E. Indeed, as we just discussed above, the two terms we are summing only differ by a quantity E. This consideration allows us to ignore all the pairs of L-crossing segments of this type. Therefore, we are left with the contributions coming from the pairs of L-crossing segments related to cases 2) and 3). If we are in the case 2), the two nonzero terms corresponding to the segment related to $\mathbf{b}_q$ and the segment related to $\mathbf{b}_{q+1}$

will differ by a quantity $E+1$. Therefore, summing both to the total sum will generate an increment of 1, up to an even quantity E. In the same way, if we are in the case 3), the two nonzero terms corresponding to the segment related to $\mathbf{b}_q$ and the segment related to $\mathbf{b}_{q+1}$ will differ by a quantity $E-1$. Therefore, summing both to the total sum will generate an increment of -1, up to an even quantity E.

Notice now that, as we already discussed above, case 2) and case 3) describe portions of the curve starting from a point close to L and ending in a point close to L that in between make a full spin around the point $\mathbf{a}$, in one direction or in the opposite, respectively. Now, since the curve ∂S is a closed curve, the number of situations we will encounter belonging to case 2) has to be the same of the one belonging to the case 3). This means that for each pair of intersection points $\mathbf{b}_q$ and $\mathbf{b}_{q+1}$ preceding $\mathbf{b}_{k^*}$ and representing a case 2) in the list described above that contributes to the sum for a factor $E+1$, there exists another pair of intersection points, say $\mathbf{b}_h$ and $\mathbf{b}_{h+1}$, for some $h \in \{2,..,k-1\}$, following $\mathbf{b}_{k^*}$ and representing a case 3) that contributes to the sum for a factor $E-1$. In the same way, for each pair of intersection points $\mathbf{b}_q$ and $\mathbf{b}_{q+1}$ preceding $\mathbf{b}_{k^*}$ and representing a case 3) in the list described above with that contributes to the sum for a factor $E-1$, there exists another pair of intersection points, say $\mathbf{b}_h$ and $\mathbf{b}_{h+1}$, for some $h \in \{2,..,k-1\}$, following $\mathbf{b}_{k^*}$ and representing a case 2) that contributes to the sum for a factor $E+1$. Moreover, notice also that, because ∂S is a closed curve, the last nonzero term of the sum in (7.3), the one related to the last L-crossing segment with intersection $\mathbf{b}_k$ produces an even quantity E. Indeed, this quantity corresponds to summing the sign of all the V-crossing segments between the first L-crossing segment (our starting point for the counting) and last L-crossing segment.

Therefore, based on what we discussed above, we can organize our total sum in (7.3) in pairs of terms producing even contributions E in the following way: the first pair producing an E contribution is the one formed by the first and the last L-crossing segment; then all the pairs of L-crossing segments related to case 1) above naturally produce E contributions, and finally each pair of L-crossing segment belonging to case 2) above, producing a $+1$ contribution to the sum, is canceled out (up to an even quantity E) by a pair of L-crossing segments belonging to case 3) above, producing a -1 contribution to the sum, and vice-versa. But as we already pointed out previously in this section, the total number of L-crossing segments has to be an odd number. There is then one contribution that has not been counted by this organization by pairs, and it is easy to see that is contribution is precisely the one related to the point $\mathbf{b}_{k^*}$.

Let us now indicate with the symbol C the number of situations belonging to the case 2) encountered in the group of segments preceding the segment related to $\mathbf{b}_{k^*}$, and with the symbol A the number of situations belonging to the case 3) encountered in the group of segments preceding the segment related to $\mathbf{b}_{k^*}$.

Then, because of we discussed until now, computing the contribution to the total sum of the L-crossing segment with intersection $\mathbf{b}_{k^*}$, we are correctly summing to an even constant E only the $+1$'s and -1's that are related to spins of the ∂S border around $\mathbf{a}$. This operation will produce a total of $C-A+E$. So, we will have that $\lfloor (h_S(\mathbf{z},\mathbf{a}))\rfloor \bmod 2 = \lfloor (\bar{h}_S(\mathbf{b}_{k^*},\mathbf{a}))\rfloor \bmod 2 = C-A+E \bmod 2$, and this concludes our analysis of the correct behavior of the algorithm and proves the theorem. ■

7.4 Example

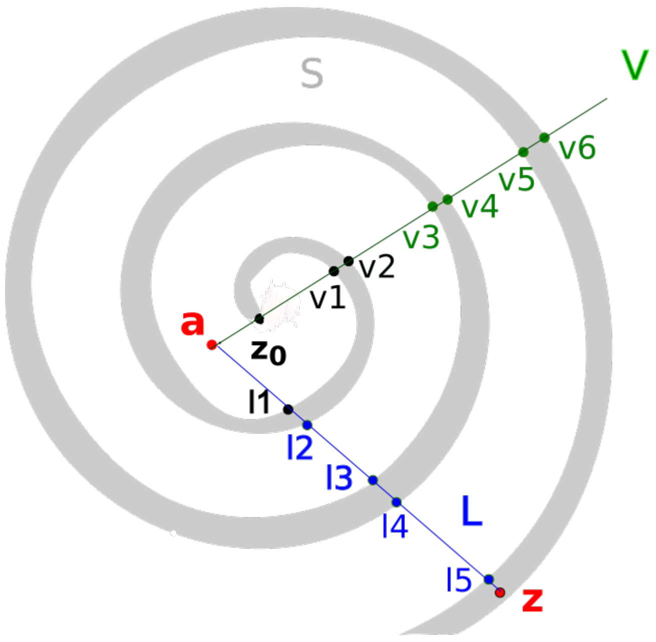


Figure 7.1: Computing the complex square root

In this section we present a simple example to clarify the functioning of our algorithm. Let us consider the case illustrated by figure 7.1 above. The letters $\mathbf{z}$, $\mathbf{a}$, $\mathbf{z_0}$, L and V represent the quantities presented in the previous section for the description of the algorithm. Specifically, $\mathbf{z_0}$ represents the origin. Notice that, in the particular case of this example, the gray domain S spins 2 times around the point $\mathbf{a}$ before reaching the point $\mathbf{z}$. In other words, the winding number of $\mathbf{z}$ with respect to $\mathbf{a}$ is 2, i.e $\lfloor (h_S(\mathbf{z},\mathbf{a}))\rfloor = 2$. Therefore, since our algorithm aims to compute the quantity $\lfloor (h_S(\mathbf{z},\mathbf{a}))\rfloor \bmod 2$, the correct output of the algorithm summation in the case of this example should be E, where, as usual, we indicate with E a generic even quantity. Hence, we have to verify that

$$\begin{aligned} G(\mathbf{z},\mathbf{a},0^{2p(n)}) &= \sum_{\{|u|=p(n), i_u > i^*\}} \sum_{\{|v|=p(n), i^* < i_v \le i_u\}} F(\mathbf{z},\mathbf{a},uv) \\ &= \sum_{i_u > i^*} |sngL_{f_n}(i_u)| \cdot \sum_{i^* < i_v \le i_u} (sngV_{f_n}(i_v) + 2) \\ &= E \end{aligned} \tag{7.4}$$

In the figure we indicated with the letters $l1, l2, l3, l4, l5$ the L intersection points, and with $v1, v2, v3, v4, v5, v6$ the V intersection points. Notice that $l1$ is the first intersection with L, and hence corresponds with the starting point of our algorithm. This is why this point is colored in black in the figure above. Therefore its contribution to the total sum is zero, i.e. $C_{l1} = 0$. The points $v1$ and $v2$ are also colored in black because their contribution is also not relevant, as we will show immediately. Let us now proceed with computing the sum above. To count the contributions to the sum we start from the starting point $l1$ as explained and follow the border ∂S in clockwise direction. Accordingly, for the signs of the intersection with L and V, we consider a positive sign for clockwise intersections and negative sign for anticlockwise. This is just a convention, and the outcome would be the same in case of the opposite choice. Moreover, since in each term of the above sum we are summing a $+2$ quantity, the calculations that follows will be considered up to an even quantity E, as usual. So, the first contribution to the sum comes from the first L intersection encountered, which is point $l3$. Notice that the only relevant V intersection preceding $l3$ is $v3$ (indeed the V intersection corresponding to point $v1$ is not relevant, since it precedes our starting point $l1$), with positive sign, and hence produces a contribution corresponding to 1. We indicate this contribution with the notation $c_{v3} = 1$. So we have $C_{l3} = c_{v3} = 1$, where we use a capital C to express the contribution coming from the L intersections, and a little c for the terms related to V intersections. Notice that at this point the portion of the border ∂S that we are considering has already completed a full spin around the point $\mathbf{a}$. Now, if we keep following the border, the second L intersection point encountered is $l5$. It is easy to see from the figure above that this point corresponds to the point we have called $\mathbf{b}_{k^*}$ in the description of the algorithm above, such that: $\lfloor(h_S(\mathbf{z}, \mathbf{a}))\rfloor \bmod 2 = \lfloor(\bar{h}_S(\mathbf{b}_{k^*}, \mathbf{a}))\rfloor \bmod 2$. This point is preceded by two relevant V intersections, the points $v3$ and $v5$, both contributing with positive sign. Hence, $C_{l5} = c_{v3} + c_{v5} = 2$. Notice that this contribution indeed corresponds to the winding number of $\mathbf{z}$ with respect to point $\mathbf{a}$.

The two remaining L intersections are $l4$ and $l2$. First, we observe that the relevant V intersections preceding $l4$ are the points $v3$, $v5$, and $v6$ where $v3$ and $v5$ have positive sign, while $v6$ has negative sign. Hence, $C_{l4} = c_{v3} + c_{v5} - c_{v6} = 2 - 1 = 1$. Finally, notice that the point $l2$ is preceded by the following relevant V intersections: $v3$, $v5$ with positive sign, and $v6$, $v4$ with negative sign. Hence $C_{l2} = c_{v3} + c_{v5} - c_{v6} - c_{v4} = 2 - 2 = 0$. Notice at this point that the V intersection $v2$ is not relevant in this analysis, since there is no L intersection that follows it.

We can then sum all these contributions, restoring the presence of a generic even quantity E, to obtain as final result of the total sum

$$\begin{aligned} G(\mathbf{z}, \mathbf{a}, 0^{2p(n)}) &= \sum_{\{|u|=p(n), i_u > i^*\}} \sum_{\{|v|=p(n), i^* < i_v \leq i_u\}} F(\mathbf{z}, \mathbf{a}, uv) \\ &= \sum_{i_u > i^*} |sngL_{f_n}(i_u)| \cdot \sum_{i^* < i_v \leq i_u} (sngV_{f_n}(i_v) + 2) \qquad (7.5) \\ &= C_{l1} + C_{l3} + C_{l5} + C_{l4} + C_{l2} \\ &= 0 + 1 + 2 + 1 + 0 + E = 2 + E = E \end{aligned}$$

which is exactly the result we wanted to verify.

Chapter 8

Conclusions

To conclude this thesis work we quickly summarize in the following list the topics discussed in each one of the previous chapters.

In chapter one we introduced the main topics related to this work. In chapter two we presented the main analog classes previously used in [20] to obtain a characterization of polynomial standard complexity classes by means of systems of ODEs. We also explained how to build a specific dynamical system composed only of ODEs in order to successfully simulate the computation of Turing machines and how exactly the equivalence with FP has to be interpreted. In chapter three we extended the results of [20] and showed that the process presented in the previous chapter can be performed also in the case of Turing machines working for an exponential amount of steps, therefore leading to a complete characterization of the class FEXPTIME. In chapter four we tackled the case of functions computable in polynomial space. In order to do so, we had to abandon the encoding previously used for the simulation of Turing machines and to introduce a new notion of emulability, which we called space-emulability. In this way we established a complete characterization of the standard complexity class FPSPACE. In chapter five we demonstrated how it is possible to adapt the solutions found for the previous characterizations of classes of functions to classes of sets as well, including therefore the standard complexity classes of EXPTIME and PSPACE in the description. In chapter six we showed how to generalize the construction illustrated in chapter three for exponentials to other classes of functions, applying those guidelines to the concrete case of the functions belonging to the Grzegorczyk hierarchy. This also implied a characterization of the class of elementary functions and the class of primitive recursive functions. In chapter seven we studied the problem of computing single-valued, analytic branches of the square root function over simply connected complex domains whose boundary is given by a polynomial time computable Jordan curve. Specifically, we improved the upper complexity bound of this problem from the class P^{MP} to the class $\mathrm{P}^{\oplus \mathrm{P}}$.
Finally, we would like to mention some possible directions of research that arise as consequences of the work conducted for this thesis. The following questions are still open and constitute an enlargement as well as a possible application of this thesis work.

1. Taking as an inspiration the model already described in [20] for the case of polynomial complexity, it is not clear whether it is possible to modify our exponential class ATSE in order to achieve a set of alternative equivalent definitions similar to the definitions of classes AOP , ALP for the ATSP case, as described in [51]. This possibility is particularly relevant, since the resources provided to the polynomial characterization by these alternative equivalent definitions are the key element to the heart of a full characterization of computable analysis. Instead, a description by means of polynomial ODEs of functions computable in exponential time in the sense of computable analysis is still missing in our approach.

2. One of the natural consequences of our polynomial space characterization is the idea of discovering which features could be modified in order to continuously simulate nondeterministic polynomial computations. This is due to the well known fact that P $\subseteq$ NP $\subseteq$ PSPACE. The most promising route to follow for achieving this goal seems to be to exploit the definition of NP that makes use of the existential quantifier, nevertheless, to the present date, there is no exact clue on how to successfully introduce this element into the picture without spoiling the continuity requirements that are necessary for dynamical systems of ODEs.

3. One of the implications of the algorithm we described for the problem of computing the complex square root is the possibility of computing and summing quantities defined over the boundary of simply connected domains on the complex plane. Specifically, we are able to compute and sum these quantities even when they are defined on particularly problematic non computable points of the boundary. Since a branch of computable analysis studies the computability and complexity of complex sets whose boundaries are polynomial time computable, it is likely that other complexity problems in the context of computable analysis could benefit from our result. Therefore, one possible direction of research could be to discover the right context where to make use of the complexity improvement achieved by our theoretical algorithm.

The work related to the characterization of space complexity classes FPSPACE and PSPACE, presented in chapters four and five respectively, has been included in an article recently submitted to a scientific journal. The material related to complexity classes FEXPTIME, EXPTIME and the generalization to the Grzegorczyk hierarchy is included in an article soon to be submitted to a scientific journal. The content of chapter seven related to the complex square root represents ongoing work.

Bibliography

[1] A. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 116, 1939.

[2] S.C. Kleene. Recursive predicates and quantifiers. *Transactions of the American Mathematical Society, JSTOR*, 1943.

[3] R. Gandy. *Studies in Logic and the Foundations of Mathematics.* Elsevier, 1980.

[4] K. Gödel and S. Feferman. *Kurt Gödel: Collected Works: Publications 1929-1936*, volume I. Oxford University Press on Demand, 1986.

[5] W. Sieg. Calculations by man and machine: Mathematical presentation, synthese library. 2002.

[6] R.P. Feynman. Simulating physics with computers. *Int. J. Theor. Phys*, 1982.

[7] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:1484–1509, 1997.

[8] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67–100, 1995.

[9] E.B. Davies. Building infinite machines. *The British Journal for the Philosophy of Science*, 52, 2001.

[10] G. Etesi and I. Németi. Non-Turing computations via Malament-Hogarth space-times. *International Journal of Theoretical Physics*, 41(2):341–370, 2002.

[11] O. Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1):21–71, 1999.

[12] J. Copeland. Even Turing machines can compute uncomputable functions. In J. Casti, C. Calude, and M. Dinneen, editors, *Proc. International Conference on Unconventional Models of Computation (UMC'98)*, pages 150–164. Springer, 1998.

[13] B.J. Copeland. Accelerating turing machines. *Minds and Machines*, 12, 2002.

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

[15] C. E. Shannon. Mathematical theory of the differential analyzer. *Journal of Mathematics and Physics*, 20:337–354, 1941.

[16] V. Bush. The differential analyzer. A new machine for solving differential equations. *Journal of the Franklin Institute*, 212:447–488, 1931.

[17] K. Weihrauch. *Computable Analysis: an Introduction.* Springer, 2000.

[18] O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, 23(3):317–335, 2007.

[19] O. Bournez and A. Pouly. A universal ordinary differential equation. *Logical Methods in Computer Science*, 16, 2020.

[20] O. Bournez, D. S. Graça, and A. Pouly. Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. *Journal of the ACM*, 64(6):38:1–38:76, 2017.

[21] M. Hofmann. An application of category-theoretic semantics to the characterisation of complexity classes using higher-order function algebras. *Bulletin of Symbolic Logic*, 3.4:469–486, 1997.

[22] E. De Benedetti P. Baillot and S. Ronchi Della Rocca. Characterizing polynomial and exponential complexity classes in elementary lambda-calculus. *Information and Computation*, 261:55–77, 2018.

[23] H. T. Siegelmann and E. D. Sontag. On the computational power of neural networks. *Journal of Computer and System Sciences*, 50(1):132–150, 1995.

[24] L. Arnold. *Random dynamical systems.* Springer, 1987.

[25] M. W. Hirsch, S. Smale, and R. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos.* Academic Press, 2004.

[26] E. D. Sontag. *Mathematical Control Theory.* Springer, 2nd edition, 1998.

[27] R.M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261, 1976.

[28] S. Smale. Differentiable dynamical systems. *Bulletin of the American Mathematical Society*, 73:747–817, 1967.

[29] R. Alur and D. L. Dill. Are timed automata updatable? In E. A. Emerson and A. P. Sistla, editors, *Proc. 12th International Conference Computed Aided Verification (CAV 2000)*, volume 1855 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2000.

[30] P. Collins and J. H. van Schuppen. Observability of piecewise-affine hybrid systems. In R. Alur and G. J. Pappas, editors, *Proc. 7th International Workshop Hybrid Systems: Computation and Control (HSCC 2004)*, volume 2993 of *Lecture Notes in Computer Science*, pages 265–279. Springer, 2004.

[31] E. N. Lorenz. Deterministic non-periodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[32] S. Smale. Mathematical problems for the next century. *The Mathematical Intelligencer*, 20:7–15, 1998.

[33] W. Tucker. A rigorous ODE solver and Smale's 14th problem. *Foundations of Computational Mathematics*, 2(1):53–117, 2002.

[34] J.C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: RungeKutta and General Linear Methods.* Wiley-Interscience, New York, 1987.

[35] E. Lindelöf. Sur l'application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 116, 1894.

[36] Amaury Pouly and Daniel S. Graça. Computational complexity of solving polynomial differential equations over unbounded domains. *Theoretical Computer Science*, 626(2):67–82, 2016.

[37] A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010.

[38] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.

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

[40] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. *Comptes rendus hebdomadaires des séances de l'Académie des Sciences (Paris)*, 241:151–153, 1955.

[41] K.-I Ko. *Complexity Theory of Real Functions.* Birkhäuser, 1991.

[42] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

[43] A. Kawamura. Type-2 computability and moore's recursive functions. *Electronic Notes in Theoretical Computer Science*, 120, 2004.

[44] M. Braverman. On the complexity of real functions. In *In 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, pages 155–164, 2005.

[45] A. Mostowski. On computable sequences. *Fundamenta Mathematicae*, 44(37-51), 1957.

[46] A. Kawamura and S. Cook. Complexity theory for operators in analysis. *ACM Transactions on Computation Theory*, 42(2):Article No. 5, 2012.

[47] A. Kawamura and H. Ota. *Small complexity classes for computable analysis.* International Symposium on Mathematical Foundations of Computer Science. Springer, 2014.

[48] W. Thomson. Mechanical integration of the general linear differential equation of any order with variable coefficients. *Proceedings of the Royal Society*, 24(164-170), 1876.

[49] M. B. Pour-El. Abstract computability and its relations to the general purpose analog computer. *Transactions of the American Mathematical Society*, 199:1–28, 1974.

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

[51] O. Bournez, D. S. Graça, and A. Pouly. On the functions generated by the general purpose analog computer. *Information and Computation*, 257:34 – 57, 2017.

[52] D. S. Graça, J. Buescu, and M. L. Campagnolo. Boundedness of the domain of definition is undecidable for polynomial ODEs. In R. Dillhage, T. Grubba, A. Sorbi, K. Weihrauch, and N. Zhong, editors, *Proc. 4th International Conference on Computability and Complexity in Analysis (CCA 2007)*, volume 202 of *Electronic Notes in Theoretical Computer Science*, pages 49–57. Elsevier, 2007.

[53] D. S. Graça, J. Buescu, and M. L. Campagnolo. Computational bounds on polynomial differential equations. *Applied Mathematics and Computation*, 215(4):1375–1385, 2009.

[54] D. S. Graça, N. Zhong, and J. Buescu. Computability, noncomputability and undecidability of maximal intervals of IVPs. *Transactions of the American Mathematical Society*, 361(6):2913–2927, 2009.

[55] A. Kawamura, H. Ota, C. Rösnick, and M. Ziegler. Computational complexity of smooth differential equations. *Logical Methods in Computer Science*, 10(1:6):1–15, 2014.

[56] K.-I Ko. On the computational complexity of ordinary differential equations. *Information and control*, 58:157–194, 1983.

[57] O. Bournez, D. S. Graça, and A. Pouly. On the complexity of solving initial value problems. In *Proc. 37h International Symposium on Symbolic and Algebraic Computation (ISSAC 2012)*, volume abs/1202.4407, 2012.

[58] O. Bournez and M. L. Campagnolo. *Handbook of Computability and Complexity in Analysis*, chapter A survey on analog models of computation, pages 173–226. Springer, 2021.

[59] J. Durand-Lose. Abstract geometrical computation and computable analysis. In C. S. Calude, J. F. Costa, N. Dershowitz, E. Freire, and G. Rozenberg, editors, *Proc. 8th International Conference on Unconventional Computation (UC 2009)*, volume 5715 of *Lecture Notes in Computer Science*, pages 158–167. Springer, 2009.

[60] A. Ben-Hur, H. T. Siegelmann, and S. Fishman. A theory of complexity for continuous time systems. *Journal of Complexity*, 18(1):51–86, 2002.

[61] M. Campagnolo and C. Moore. Upper and lower bounds on continuous-time computation. In I. Antoniou, C. Calude, and M. Dinneen, editors, *Proc. 2nd International Conference on Unconventional Models of Computation - UMC'2K*, pages 135–153. Springer, 2001.

[62] O. Bournez, D. S. Graça, and A. Pouly. Turing machines can be efficiently simulated by the general purpose analog computer. In FT-H. H. Chan, L. C. Lau, and L. Trevisan, editors, *Proc. 10th annual conference on Theory and Applications of Models of Computation (TAMC 2013)*, LNCS 7876, pages 169–180. Springer, 2013.

[63] O. Bournez, D. S. Graça, and A. Pouly. Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length – the general purpose analog computer and computable analysis are two efficiently equivalent models of computations. In I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, editors, *Proc. 43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 109:1–109:15. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[64] J. Buescu, D. S. Graça, and N. Zhong. Computability and dynamical systems. In M. Peixoto, A. Pinto, and D. Rand, editors, *Dynamics and Games in Science I*, pages 169–181. Springer, 2011.

[65] D. S. Graça, M. L. Campagnolo, and J. Buescu. Robust simulations of Turing machines with analytic maps and flows. In S. B. Cooper, B. Löwe, and L. Torenvliet, editors, *Proc. CiE 2005: New Computational Paradigms*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179. Springer, 2005.

[66] D. S. Graça, M. L. Campagnolo, and J. Buescu. Computability with polynomial differential equations. *Advances in Applied Mathematics*, 40(3):330–349, 2008.

[67] O. Bournez and E. Hainry. An analog characterization of elementarily computable functions over the real numbers. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2004.

[68] O. Bournez and E. Hainry. Real recursive functions and real extentions of recursive functions. In M. Margenstern, editor, *Proc. International Conference on Machines, Computations and Universality (MCU 2004)*, volume 3354 of *Lecture Notes in Computer Science*, pages 116–127, 2004.

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

[70] K.I. Ko and F. Yu. On the complexity of computing the logarithm and square root functions on a complex domain. *J. Complexity*, 23, 2007.

[71] A.W. Chou and K.I. Ko. Computational complexity of two-dimensional regions. *SIAM Journal on Computing*, 24(5), 1995.

[72] M.Braverman. Hyperbolic julia sets are poly-time computable. In *Proc. of the sixth workshop on computability and complexity in analysis*, 2005.

[73] R. Rettinger and K. Weihrauch. The computational complexity of some Julia sets. In *Proc. 35th annual ACM symposium on Theory of computing*, pages 154–185. ACM, 2003.

[74] K. Ko. A polynomial time computable curve whose interior has a nonrecursive measure. *Theoret. Comput. Sci.*, 145:241–270, 1995.

[75] K. Ko. On the computability of fractal dimensions and hausdorff measure. *Ann. Pure Appl. Logic*, 93:195–216, 1998.

[76] A.W. Chou and K. Ko. On the complexity of finding paths in a two-dimensional domain i: shortest path. *Math. Logic Quart.*, 50:551–572, 2004.

[77] A.W. Chou and K. Ko. On the complexity of finding paths in a two-dimensional domain ii: piecewise straight-line paths. In *Proc. of the sixth workshop on computabilty and complexity in analysis*, volume 120, pages 45–57, 2005.

[78] M. Braverman. Computational complexity of euclidean sets: hyperbolic Julia sets are poly-time computable. In V. Brattka, L. Staiger, and K. Weihrauch, editors, *Proc. 6th Workshop on Computability and Complexity in Analysis (CCA 2004)*, volume 120 of *Electronic Notes in Theoretical Computer Science*, pages 17–30. Elsevier, 2005.

[79] M. L. Campagnolo, C. Moore, and J. F. Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4):642–660, 2000.

[80] J. H. Hubbard and B. H. West. *Differential Equations: A Dynamical Systems Approach — Higher-Dimensional Systems.* Springer, 1995.

[81] M. L. Campagnolo. *Computational Complexity of Real Valued Recursive Functions and Analog Circuits.* PhD thesis, Instituto Superior Técnico/Universidade Técnica de Lisboa, 2002.

[82] O. Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, revised first edition, 2008.

[83] A. Grzegorczyk. Some classes of recursive functions. *Rozpr. Mat.*, 4:1–45, 1953.

[84] P. Odifreddi. *Classical Recursion Theory*, volume 2. Elsevier, 1999.

[85] P. Henrici. *Applied and Computational Complex Analysis*, volume 1. Wiley, New York, 1974.

[86] F. Green and al. The power of the middle bit of a # P function. *Journal of Computer and System Sciences*, 50(3), 1995.

[87] D.Z. Du and K.I. Ko. *On the complexity of computing the logarithm and square root functions on a complex domain*, volume 58. John Wiley & Sons, 2011.

Index