

PSPACE-Completeness of the Reachability Relation of Robust Dynamical Systems

Manon Blanc

ITU i København

Abstract

Reachability for dynamical systems over real domains is undecidable in the general case: it is only computably enumerable, even under reasonable hypotheses, e.g. for a computable relation. It can be proven by embedding a Turing machine into a dynamical system; the machine stops if and only if the reachability relation holds. However, with a proper notion of *robustness*, i.e. allowing a small perturbation in the dynamics, reachability in a dynamical system becomes decidable. Also, by properly quantifying the allowed perturbation in the dynamics, it is possible to prove the **PSPACE**-membership for the reachability relation, for **PSPACE** computing over real numbers in the sense of computable analysis ([5]). In this paper, we focus on completeness results for **PSPACE** over the reals, regarding various *robust* dynamical systems. We prove that, under some additional and rather reasonable hypotheses, the reachability relation becomes even **PSPACE**-complete in *second-order* complexity, for an adapted notion of reduction. As a side-effect, we also prove an improvement of an algebraic ODE-based characterisation of **PSPACE**, removing one of the base functions of the algebra. By doing so, we link results from dynamical systems and complexity over the reals, including the models of computable analysis and second-order complexity.

1 Introduction

Dynamical systems are widely used in many areas of fundamental and applied science. Their interactions with computations over the reals have various motivations (generalised shifts [36], low-dimensional dynamical systems [35], control theory [7], analogue computing [15]). A major problem about dynamical systems is the problem of reachability (**Reach**): is there a path in the dynamical system reaching some given point y starting from a given x ? This decision problem is well-known to be computably enumerable (c.e.), even when the underlying relation is computable, but not decidable in the general case. However, with a proper notion of *robustness*, as defined in [1] (Definition 3), **Reach** becomes decidable. Intuitively, the involved notion can be understood as insensitivity to an arbitrarily small perturbation. Instead of sending one point to another point, the perturbed dynamic function sends one point to an open set, in the topological sense. A system is robust if and only if each of those open sets intersects the non-perturbed solution point. For example, in game theory or in verification, even if **Reach** is not decidable in the general case, we can still

certify that a subclass of dynamical systems can reach or not reach some subset of states. The authors of [1] and [22] apply this concept mainly to decidability questions. Considering dynamical systems stable under an infinitesimal perturbation is an old idea, rediscovered several times under different names. Among them, we mention the concept of “chain reachability” ([20], [19]). In verification, the concept of infinitesimally perturbed dynamics has been explored to provide alternative semantics for certain models of computation; e.g. [39] for timed automata. The authors of [5] proved it is also possible to deal with complexity questions, and not only computability. With the proper quantification over the allowed perturbation, namely the radius of the open sets ($2^{-p(n)}$, p a polynomial and $n \in \mathbb{N}$ related to some notion of size), they prove that **Reach** is in **PSPACE**. It involves computability and complexity theory over the reals, since they handle general dynamical systems described by Ordinary Differential Equations (ODEs). Thus, as in [5], we mainly use the *computable analysis*’ model.

Many different models of computation exist for real numbers. For discrete-time models of computations over the reals, the most famous approaches are computable analysis, based on Turing machines (TMs) ([45], [49]) and algebraic models such as the Blum Shub Smale (BSS) model ([9], [8]). An extensive list of decision problems in BSS has been recently described: see [40]. Both models were tailored for very different applications, and we cannot unify existing models with the equivalent of a Church-Turing thesis. For example, computable functions in a computable analysis model are necessarily continuous (see e.g. [49]), while the BSS model intends to consider functions and problems over the polynomials that are not necessarily continuous. It is also explained by the fact that some models have not been introduced with the idea of corresponding to actual physical machines, but also to discuss abstract complexity (lower and upper bounds) for the associated problems. Using TMs, in a discrete-time framework, the authors of [4] prove it is possible to have algebraic characterisations of **PSPACE**, relying on discrete ODEs (also called finite difference ODEs).

For continuous-time models, one of the first machines was the Differential Analysers [47], mathematically formalised by the General Purpose Analog Computer (GPAC) ([41]). There are also various recent approaches and models from deep learning, such as Neural ODEs [18, 33]. Each model corresponds to a particular class of ODEs. E.g., the GPAC corresponds to polynomial ODEs [24] and Neural ODEs are made by selecting the best solution among a parameterised class of ODEs (e.g. [33]). With TMs, the authors of [6] prove it is possible to have algebraic characterisations of **PSPACE**, relying on continuous ODEs. In [29], the authors have **PSPACE**-completeness results by restricting themselves to compact domains.

We prove in this paper that, under the proper assumptions, reachability for polynomially robust dynamical systems (Subsection 2.2) is **PSPACE**-hard on a rational domain and also in second-order complexity (written **PSPACE**²- _{\leq_m^2} -hard in this context) for unbounded-time domains, which implies that it is **PSPACE**-complete.

Our contributions In the general case, the problem of reachability is not even decidable.

- We prove in Section 3 that **PSPACE**-completeness, for a robust dy-

namical system of a rational domain (included in \mathbb{Q}) (see Definition 3) (**Proposition 2** and **Corollary 2**). We consider here the framework of [4], which gives a simple description of polynomial space with discrete ODEs. Here we use the usual reduction between decision problems for this: $A \leq B$ if there is a computable in polynomial time function f such that on every w , $w \in \text{Dom}(A)$ is equivalent to $f(w) \in B$, as defined in [43]. It is sufficient, since rationals can be represented by tuples of integers.

- We also show that, with robust Type-2 TMs (Definition 13), and second-order reductions over real functions (Definition 7), we have the \mathbf{PSPACE}^2 - \leq_m^2 -hardness (see Definition 8) of **Reach**, for both discrete-time (**Theorem 7** and **Corollary 3**) and continuous-time dynamics (**Theorem 9** and **Corollary 5**). With the membership results of [5], it implies that both problems are \mathbf{PSPACE}^2 - \leq_m^2 -complete. We are no longer in the previous framework, allowing us to deal with somewhat more general dynamic functions.
- As a side-effect, we prove we can remove the cosine as a base function from the algebra of [6], which gives a characterisation of **FSPACE** over the reals (**Theorem 8**). It comes from an improvement of the error analysis in the iteration trick we use (**Lemma 4**): we express the cosine as the solution of a system of *polynomial* ODEs.

Organisation of the paper In Section 2, we introduce several tools and existing results that we rely on later. It first includes some notion of dynamical systems and their computability properties, and the notion of robustness we consider here. Then we introduce *second-order* complexity and previous results regarding complexity classes over the reals and hardness results. Then, In Section 3, we prove **PSPACE**-completeness of the reachability relation, considering the framework of [4], with the usual reduction between decision problems for this. In Section 4, we prove our main result about \mathbf{PSPACE}^2 - \leq_m^2 -completeness of the reachability relation for robust discrete- and continuous-time dynamical systems, with a stronger notion of reduction, dealing with functions over the reals in second-order complexity ([29]).

Notations In the rest of the paper, for $a, b \in \mathbb{R}$, $a =_\epsilon b$ means $\|a - b\| < \epsilon$. For A, B two sets, f some function, $f \in B^A$ means $f : A \rightarrow B$. We define an algebra $[f_1, \dots, f_n; o_1, \dots, o_k]$ as the set of functions containing the functions f_1, \dots, f_n , stable under the operators o_1, \dots, o_k .

2 Mathematical Background and state of the art

We assume the reader is familiar with the basic concepts of *classical* computability, including TMs, and complexity. We also consider notions of computable analysis, see [34], [49] for a more comprehensive study. The main notions can be found in Appendix A.

2.1 Discrete-Time Dynamical Systems

As dynamical systems are widely used in fundamental and applied science, the relations between dynamical systems over the reals and computations have been the source of many works, such as [36], [2], [15]. For now, we are interested in discrete-time dynamical systems:

Definition 1 (Discrete-Time Dynamical System [27]). *A discrete-time dynamical system \mathcal{H} is a tuple (X, f) , with X a set, called the domain of \mathcal{H} , and $f : X \rightarrow X$ such that for all $\mathbf{x} \in X$, the dynamic starting from x is defined by the sequence $\mathbf{x}_0 = \mathbf{x}$ and $\mathbf{x}_{n+1} = f(\mathbf{x}_n)$. In other words, $(\mathbf{x}_n)_{n \in \mathbb{N}} = (f^n(\mathbf{x}))_{n \in \mathbb{N}}$.*

The main problems we consider here are the problems of *point-to-point reachability* and *point-to-set reachability* in a discrete-time dynamical system. The reachability problem determines whether there exists $n \in \mathbb{N}$ such that $y = f^n(\mathbf{x})$ or $f^n(\mathbf{x}) \in Y$. For natural classes of dynamical systems, this problem is well-known to be c.e. (for f computable): starting from x , we apply f until y or Y are reached. We write $\mathfrak{R}^{\mathcal{H}}(\mathbf{x}, \mathbf{y})$ and $\mathfrak{R}^{\mathcal{H}}(\mathbf{x}, Y)$ for the corresponding relations in the dynamical system \mathcal{H} .

Reach(\mathbf{x}, \mathbf{y}):

Input: A dynamical system $\mathcal{H} = (X, f)$, $\mathbf{x}, \mathbf{y} \in X$
Output: $\mathfrak{R}^{\mathcal{H}}(\mathbf{x}, \mathbf{y})$

Reach(\mathbf{x}, Y):

Input: A dynamical system $\mathcal{H} = (X, f)$, $\mathbf{x} \in X$, $Y \subseteq X$
Output: $\mathfrak{R}^{\mathcal{H}}(\mathbf{x}, Y)$

We notice that restricting f to be a PAM, instead of just computable, does not improve the computability, where PAM stands for “Piecewise Affine Map”, a map of the form $f(\mathbf{x}) = \mathbf{A}_i \mathbf{x} + \mathbf{b}_i$, $\mathbf{x} \in P_i$, $i = 1 \dots N$, where A_i are rational $d \times d$ -matrices, $b_i \in \mathbb{Q}^d$ and P_i are convex rational polyhedral sets. We also have some *classical* complexity results for time- and space-bounded dynamical systems. See Appendix B.

2.2 Robustness for dynamical systems

We give more details on the notion of robustness for a dynamical system of [1], so the reachability problem becomes decidable. The idea is to apply the paradigm of small perturbations. Let $\mathcal{H} = (X, \mathbf{f})$ be discrete-time dynamical system. For any $\epsilon > 0$, \mathcal{H}_ϵ defines the ϵ -perturbed system. Its trajectories are defined as sequences \mathbf{x}_t satisfying $d(\mathbf{x}_{t+1}, \mathbf{f}(\mathbf{x}_t)) < \epsilon$ for all t , and d the sup-norm distance. This non-deterministic system is considered as \mathcal{H} subjected to a noise of magnitude ϵ . For convenience, we write $\mathbf{y} \in \mathbf{f}_\epsilon(\mathbf{x})$ as a synonym for $d(\mathbf{f}(\mathbf{x}), \mathbf{y}) < \epsilon$. The reachability in the system \mathcal{H}_ϵ is denoted by $\mathfrak{R}_\epsilon^{\mathcal{H}}(\cdot, \cdot)$ and $\bar{B}(\mathbf{y}, \epsilon)$ the (open) ball of radius ϵ centered on \mathbf{y} . Thus, we give the definition of the *robust* reachability problem:

Definition 2 (Robust Reach [5]). *With a discrete-time dynamical system \mathcal{H} , we associate the relation $\mathfrak{R}_\epsilon^{\mathcal{H}}(\mathbf{x}, \mathbf{y})$, such that for two points \mathbf{x} and \mathbf{y} in \mathbb{Q} , $\epsilon = 2^{-p}$ for $p \in \mathbb{N}$, if there exists a finite trajectory in \mathcal{H} from \mathbf{x} to $\bar{B}(\mathbf{y}, \epsilon)$.*

Definition 3 (Discrete-Time Robust Dynamical System [5]). *We consider a discrete-time dynamical system \mathcal{H} robust iff for all $\epsilon = 2^{-p}$, $p \in \mathbb{N}$, $\mathbf{x}, \mathbf{y} \in \text{Dom}(\mathcal{H})$, $\mathfrak{R}_\epsilon^{\mathcal{H}}(\mathbf{x}, \mathbf{y}) = \mathfrak{R}^{\mathcal{H}}(\mathbf{x}, \mathbf{y})$*

All trajectories of a non-perturbed system \mathcal{H} are also trajectories of the ϵ -perturbed system \mathcal{H}_ϵ . If $\epsilon_1 < \epsilon_2$ then any trajectory of the ϵ_1 -perturbed system is also a trajectory of the ϵ_2 -perturbed system. By [5], **Reach** for robust discrete-time dynamical systems is co-c.e. We deduce that **Reach** for robust discrete-time dynamical systems is decidable. We denote by $\mathfrak{R}_{\{p\}}^{\mathcal{H}}$, for $p \in \mathbb{N}^{\mathbb{N}}$

a polynomial, the perturbed reachability relation where the perturbation, the ϵ in Definition 2, is $2^{-p(n)}$, for $n \in \mathbb{N}$ given as an input. Such systems are called “polynomially robust”.

Theorem 1 ([5]). *Let $\mathcal{H} = (X, f)$, with $\mathbf{f} \in \mathbf{FPTIME}$ locally Lipschitz and X a closed rational box ($X = \prod_{i=1}^d [a_i, b_i] \subseteq \mathbb{R}$, $a_i, b_i \in \mathbb{Q}$). Then $\mathfrak{R}_{\{p\}}^{\mathcal{H}} \subseteq \mathbb{Q}^d \times \mathbb{Q}^d \times \mathbb{N} \in \mathbf{PSPACE}$, for p a polynomial.*

	$X \subseteq \mathbb{R}$, \mathcal{H} non-robust	\mathcal{H} robust	\mathcal{H} poly. robust, $f \in \mathbf{FPTIME}$ Lips.
Reach (x, y)	Undecidable	Decidable [1]	$\in \mathbf{PSPACE}$ [5]

In Section 3, we prove the **PSPACE**-hardness in the case where the system \mathcal{H} is robust and $f \in \mathbf{FPTIME}$ Lipschitz.

2.3 Reductions in continuous settings and second-order

Computability and complexity classes over the reals depend on the model of computation we consider, and also the representations of the reals: some complexity results hold for some representation (of the reals) but not for another (see [49]): a *representation* of a set S is a surjective function $\delta : \mathbf{Reg} \rightarrow S$ ([49], [29]). Here, we consider only equivalent representations (see Definition 20 in Appendix A). We could represent real numbers with infinite sequences (in Σ^ω). The problem with such representation is that there is no robust measure of size for space complexity. Following [29], we use **Reg** (Definition 4) as a naming system for real numbers, sets and functions. Hence, the complexity is defined as the length of the word representing the real number, plus the size of the underlying oracle, also represented by a word. The size of a regular function ϕ , also denoted by $|\phi|$, is $|\phi|(|u|) = |\phi(u)|$.

Definition 4 (Regular Functions [29]). *A function $f : \Sigma^* \rightarrow \Sigma^*$ is regular iff, for all $u, v \in \text{Dom}(f)$ such that $|u| \leq |v|$, then $|f(u)| \leq |f(v)|$.*

The usual notion of reduction only deals with sets of languages as inputs. Here, we have functions as inputs; thus, we do reductions for the complexity with *pseudofunctions*. In this context, we say that complexity classes are defined in *second-order*. More precisely, second-order *polynomials* ([29]) are defined inductively as follows: a positive integer and a variable are a second-order polynomial. If P, Q are second-order polynomials, then so are $P + Q, P \cdot Q, P(L)$ where $L \in \mathbb{N}^{\mathbb{N}}$ is a type-1 (a total or partial map from \mathbb{N} to \mathbb{N}) variable. A second-order polynomial P is in $(\mathbb{N}^{\mathbb{N}})^{\mathbb{N}^{\mathbb{N}}}$.

Definition 5 (Second-order polynomials [29]). *A ((non-)deterministic) TM M runs in second-order polynomial time if there is a second-order polynomial P such that, given any oracle $\phi \in \mathbf{Reg}$, any input $u \in \Sigma^*$, M halts within $P(|\phi|)(|u|)$ steps. Define second-order polynomial space analogously by counting the number of visited cells on all tapes.*

Definition 6 (Type-2 Complexity Classes). **PSPACE**² (resp. **PTIME**²) is the class of languages decidable in second-order polynomial space (resp. time) by a Type-2 TM. **FSPACE**² (resp. **FPTIME**²) is the class of functions computable in second-order polynomial space (resp. time) by a Type-2 TM.

We now define an adapted notion of reduction, with respect to Type-2 TMs (Definition 13). Following [29], the reductions work on the set of regular functions **Reg**. For decision problems, we assume that the represented set S is $\{0, 1\}^{\Sigma^*}$. We denote by **Pred** the set of $\{0, 1\}$ -valued regular functions, as in [29]:

Definition 7 (Type-2 reductions [29]). *Let A and B be decision problems, or multi-functions from **Reg** to **Pred**. We write $A \leq_m^2 B$ if there exists $\mathfrak{s}, \mathfrak{t} \in \mathbf{FPTIME}^2$ such that for any $\phi \in \text{Dom}(A)$, we have $\mathfrak{s}(\phi) \in \text{Dom}(B)$ and for $\theta = B(\mathfrak{s}(\phi))$ the function $\theta \circ \mathfrak{t}(\phi) = A(\phi)$.*

Intuitively, for A and B two problems, we have that $\text{Dom}(A)$ is the set of functions, which are inputs of the problem A , and $A(\phi)(x) = (B(\mathfrak{s}(\phi)) \circ \mathfrak{t}(\phi))(x)$.

Definition 8 ($\mathbf{PSPACE}^2\text{-}\leq_m^2$ -completeness). *A problem P is $\mathbf{PSPACE}^2\text{-}\leq_m^2$ -complete iff $P \in \mathbf{PSPACE}^2$ and P is $\mathbf{PSPACE}^2\text{-}\leq_m^2$ -hard: for all problem $P' \in \mathbf{PSPACE}^2$, $P' \leq_m^2 P$.*

This construction seems somewhat artificial. The point is, it is strong enough to make various classical problems $\mathbf{PSPACE}^2\text{-}\leq_m^2$ -complete ([29]), including the following, and it is invariant under replacing the representations with equivalent ones:

Basic \mathbf{PSPACE}^2 :

Input: $\langle M, \bar{\mu}, \phi \rangle$, with M a Type-2 TM, $\bar{\mu} \in \mathbf{Reg}$ such that $\bar{\mu}(u) = 0^{\mu(|u|)}$ where $\mu : \mathbb{N} \rightarrow \mathbb{N}$ a non-decreasing polynomial bounding the space of M , $\phi \in \mathbf{Reg}$; a string u .

Output: Does M , with an oracle ϕ , accept u ?

where “ $g^{[k]}$ ” denotes the fact that g is iterated k times. If g is a function, then $g^{[k]}$ is the function corresponding to g composed k times with itself. If g is a string, then $g^{[k]}$ is the string with g repeated k times. We denote by $|s|$ the length of the string s .

Proposition 1 ([29]). *Basic \mathbf{PSPACE}^2 is $\mathbf{PSPACE}^2\text{-}\leq_m^2$ -complete.*

2.4 Previous results on complexity over the reals

The discrete derivation is denoted by δ , defined as $\frac{\delta f}{\delta n}(n, \mathbf{x}) = f(n+1, \mathbf{x}) - f(n, \mathbf{x})$, and the (continuous) derivation with ∂ . A *discrete ODE* is an ODE using a discrete derivation. We introduce algebraic characterisations of **FPTIME** and **FPSPACE**, given in [4]. They allow us to have a proper equational representation for the execution of a TM. The main model of discrete ODEs to characterise **FPTIME** are Linear Length ODEs. A length ODE is a particular case of a discrete ODE, where instead of having a derivative step of size 1, the derivation steps is of size the *binary* length of n . The length function $\ell : \mathbb{N} \rightarrow \mathbb{N}$ maps some integer to the length of its binary representation. Thus, the derivative steps are of size the successive powers of 2. A more detailed definition can be found in Appendix C. The size of the solution of a linear length ODE can never exceed a polynomial, as proven in [3], ensuring the **FPTIME**-membership of solving such ODEs. Also, **FPTIME** and **FPSPACE** are not stable under the general notion of limit. It means that the limit of a sequence of functions

computable in polynomial time (resp. space) is not necessarily computable in polynomial time (resp. space). Thus, the notion of limit must be restricted by imposing that the constructed approximations construct converge *exponentially* quickly to a solution:

Definition 9 (Effective Limit operator $ELim$, [3]). *Given $\tilde{\mathbf{f}} : \mathbb{N}^{d+1} \rightarrow \mathbb{R}^{d'} \in (\mathbb{L}\mathbb{D}\mathbb{L}^\circ)$ such that for all $\mathbf{m} \in \mathbb{N}^d$, $n \in \mathbb{N}$, $\|\tilde{\mathbf{f}}(\mathbf{m}, 2^n) - \mathbf{f}(\mathbf{m})\| \leq 2^{-n}$, then $ELim(\tilde{\mathbf{f}})$ is the (clearly uniquely defined) corresponding function $\mathbf{f} : \mathbb{N}^d \rightarrow \mathbb{R}^{d'}$.*

The effective limit of a sequence of polynomial-time (resp. -space) functions is computable in polynomial time (resp. space) ([3], [4]). Thus:

Theorem 2 (Characterisation of **FPTIME**, [4]). *Let $\overline{\mathbb{L}\mathbb{D}\mathbb{L}^\circ} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \tanh, \frac{x}{2}, \frac{x}{3};$ composition, Linear Length ODE, $ELim$], such that we have $\overline{\mathbb{L}\mathbb{D}\mathbb{L}^\circ} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}} = \mathbf{FPTIME} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}}$, where π_i^k outputs the i^{th} coordinate of a vector of size k , \tanh the hyperbolic tangent function, $\frac{x}{2} : \mathbb{R} \rightarrow \mathbb{R}$ is the function dividing by 2 (similarly for $\frac{x}{3}$).*

All other basic functions are defined as usual, considered here as functions in $\mathbb{R}^{\mathbb{R}}$. With similar ideas, there is also an algebraic characterisation of **FPSPACE** over the reals where we consider “classical” numerically stable essentially linear discrete ODEs, called *robust linear discrete ODEs* ([10, 11]).

Theorem 3 (Characterisation of **FPSPACE**, [4]). $\overline{\mathbb{R}\mathbb{L}\mathbb{D}^\circ} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}} = \mathbf{FPSPACE} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}}$, where $\overline{\mathbb{R}\mathbb{L}\mathbb{D}^\circ} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \tanh, \frac{x}{2}, \frac{x}{3};$ composition, Robust Linear Discrete ODE, $ELim$].

A characterisation of **FPSPACE** with *continuous* ODEs also exists:

Definition 10 (Robust (continuous) ODE, [6]). *A function $\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}$ is robustly ODE definable (from \mathbf{g} , and dynamic \mathbf{u}) if:*

- *it corresponds to the solution of: $\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x})$ and $\frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), \mathbf{h}(t, \mathbf{x}), t, \mathbf{x})$;*
- *there is some rational $\Delta > 0$, and some polynomial p such that the previous schema is (polynomially) numerically stable on $[0, \Delta]$: for all integer n , considering $\epsilon(n) = p(n + \ell(\mathbf{x}))$ we can compute $\mathbf{f}(t, \mathbf{x})$ by working with precision $\epsilon(n)$. Considering any solution of $\tilde{\mathbf{x}} =_{\epsilon(n)} \mathbf{x}$ and $\tilde{\mathbf{h}}(t, \tilde{\mathbf{x}}) =_{\epsilon(n)} \mathbf{h}(t, \tilde{\mathbf{x}})$, and $\tilde{\mathbf{f}}(0, \tilde{\mathbf{x}}) =_{\epsilon(n)} \mathbf{g}(\mathbf{x})$ and $\frac{\partial \tilde{\mathbf{f}}(t, \tilde{\mathbf{x}})}{\partial t} =_{\epsilon(n)} \mathbf{u}(\tilde{\mathbf{f}}(t, \tilde{\mathbf{x}}), \tilde{\mathbf{h}}(t, \tilde{\mathbf{x}}), t, \tilde{\mathbf{x}})$ then $\tilde{\mathbf{f}}(t, \tilde{\mathbf{x}}) =_{\epsilon(n)} \mathbf{f}(t, \mathbf{x})$ when $0 \leq t \leq \Delta$;*
- *for $t \geq \Delta$, we can compute $\mathbf{f}(t, \mathbf{x})$ by computing some approximation $\widetilde{\mathbf{f}}(t/2, \mathbf{x})$ of $\mathbf{f}(t/2, \mathbf{x})$ at precision $\epsilon(n)$, i.e. we can compute $\Phi(\mathbf{g}(\mathbf{x}), t/2)$, by computing some approximation of $\Phi(\widetilde{\mathbf{f}}(t/2, \mathbf{y}), t/2)$, working at precision $\epsilon(n)$.*

Theorem 4 (**FPSPACE** [6]). $\overline{\mathbb{R}\mathbb{C}\mathbb{D}} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}} = \mathbf{FPSPACE} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}}$, where $\overline{\mathbb{R}\mathbb{C}\mathbb{D}} = [0, 1, \pi_i^k, +, -, \times, \tanh, \cos, \pi, \frac{x}{2}, \frac{x}{3};$ composition, robust continuous ODE, $ELim$].

2.5 Properly encoding real numbers into a Type-2 machine

A configuration $C(q, l, r)$ of a Type-2 TM is such that $l, r \in \Sigma^\omega$ are words over $\Sigma = \{0, 1, 3\}$, $q \in Q$ denotes the internal state of M . The configuration of a bi-infinite tape TM M is encoded by real numbers using radix 4 encoding, but using only digits 1,3. Write: $\gamma_{word} : \Sigma^\omega \rightarrow \mathbb{R}$ for the function mapping a word $w = w_0w_1w_2\dots$ to the dyadic $\gamma_{word}(w) = \sum_{n \geq 0} w_n 4^{-(n+1)}$. The proof of the following lemma is direct from the definition of γ_{word} :

Lemma 1. *For $w = w_0w_1w_2\dots \in \Sigma^\omega$, for $n \in \mathbb{N}$, $\gamma_{word}(w_0w_1\dots w_n)$ is computable in polynomial time in n .*

Lemma 2 ([4]). *We can construct some function \overline{Next} in $\mathbb{L}DL^\circ$ simulating one step of M : given a configuration C , writing C' for the next one, for all integer m , $\|\overline{Next}(2^m, \overline{C}) - \overline{C}'\| \leq 2^{-m}$, for $\overline{C} = \gamma_{config}(C) = (q, \bar{l}, \bar{r}) \in \mathbb{N} \times \mathbb{R}^2$, where $\bar{r} = \gamma_{word}(r)$ and $\bar{l} = \gamma_{word}(l)$.*

In the construction of \overline{Next} , there are discontinuous functions such as the floor function ($x \in \mathbb{R} \mapsto \lfloor x \rfloor = n \in \mathbb{N}$ such that $n \leq x < n + 1$). But it must deal only with continuous functions. The main motivation behind using only symbols 1 and 3 is that it guarantees an expression like $\lfloor 4\bar{r} \rfloor$, either equals 0 (the specific case where there remain only blanks in r) or $4\bar{r}$ is in the interval $[1, 2]$ or in $[3, 4]$. A proof of Lemma 2 is in Appendix D. From Theorem 2 and Lemma 2, we can deduce:

Corollary 1. *The transition function of a (Type-2) TM is computable in polynomial time.*

3 PSPACE-completeness for the robust Reach with “classical” TMs

Reach in the case where the underlying dynamical system is robust and the dynamic function is Lipschitz polynomial-time computable, is in **PSPACE**. Now, it is interesting to prove that this decidability problem is even **PSPACE**-complete, hence has a lower-bound result. The algebra $\mathbb{L}DL^\circ = \overline{\mathbb{L}DL}^\circ \setminus \{ELim\}$ allows us to talk about **PTIME** and **PSPACE** with *smooth*, even analytic, functions. We consider dynamic functions preserving the rationals on a rational domain. As a rational number can be described by a couple of integers, we can use the classical notion of reduction in this section. Let us define $\mathbb{L}DL_{light}^\circ = \overline{\mathbb{L}DL}^\circ \setminus \{\ell(x); \text{linear length ODE}, ELim\} = [\mathbf{0}, \mathbf{1}, \pi_i^k, +, -, \tanh, \frac{x}{2}, \frac{x}{3}; \text{composition}]$

We first prove the **PSPACE**-completeness of the following problem:

Reach Robust $\mathbb{L}DL_{light}^\circ$:

Input: A discrete-time polynomially robust dynamical system $\mathcal{H} = (X, g)$, \overline{X} a closed rational box, $g \in \mathbb{L}DL_{light}^\circ$, $Y \subseteq X$ a closed rational box, $\mathbf{x} \in X$

Output: $\mathfrak{R}_\epsilon^{\mathcal{H}}(\mathbf{x}, Y)$

This problem is in **PSPACE**, as proven in [5].

Proposition 2 (Main result I). **Reach Robust $\mathbb{L}DL_{light}^\circ$ is **PSPACE**-hard.**

We reduce from the problem of knowing whether a polynomial space TM reach an accepting state. The reduction maps the set of possible configurations to the domain of the dynamical system and the transition function into the dynamic. The proof can be found in Appendix E.

Corollary 2. Reach Robust LDL_{light}° is **PSPACE**-complete.

We have a first **PSPACE**-completeness result, using the classical notion of reduction. If we had considered $\text{LDL}_{light}^\circ \setminus \{\tanh\}$, we would be in the situation where the dynamic function is a PAM. The proof of Proposition 2 also covers this case. We now prove this property but with a stronger notion of reduction and for more general domains, using Type-2 TMs.

4 $\text{PSPACE}^2\text{-}\leq_m^2$ -completeness for the robust Reach with Type-2 TMs

We impose our dynamic functions to be Lipschitz. We are going to use the notion of reduction given in Definition 7. For $\phi \in \mathbf{Reg}$ an oracle, we define the function $\Phi : \gamma(w) \rightarrow \gamma(\phi(w))$. We write **PAM** for the algebra of PAMs and the operators under which it is closed. We denote by $\mathbf{S} + \Phi$ the algebra where \mathbf{S} is an algebra and Φ is added as a base function. It gives us a strong stability result, avoiding too *chaotic* systems. We define the problem **DRRL**:

Discrete Reach Robust Lipschitz (DRRL):

Input: A discrete-time polynomially robust dynamical system $\mathcal{H} = (X, g)$,
 X a closed rational box, $g : X \rightarrow X \in \text{LDL}^\circ + \Phi$ Lipschitz, $p \in \mathbb{N}$, $\mathbf{x}, \mathbf{y} \in X$
Output: $\mathfrak{R}_{2-p}^{\mathcal{H}}(\mathbf{x}, \mathbf{y})$

The main difference with the problems considered in [29] is that we do not bound the time: there is no fixed number of iterations of g in \mathcal{H} in the input.

4.1 Transition functions for Type-2 machines

Many authors have embedded TMs in various classes of dynamical systems to get undecidability or several hardness results. We present in this section how this is usually done, to point out where (intuitively) (non-)robustness issues appear.

Theorem 5 ([3, 4]). *Let M be a TM. Then, we can construct a dynamical system $(\mathbb{N} \times \Sigma^\omega \times \Sigma^\omega, \mathbf{f})$, \mathbf{f} a PAM with rational coefficients, simulating one transition of M .*

A proof can be found in Appendix F. The transition function of a TM can be seen as a PAM (by Theorem 5). We extend Theorem 5 to deal with real numbers. We prove we can simulate an oracle since we are dealing with an oracle TM.

Proposition 3. *Let $M = (Q, \Sigma, \Delta_M, q_{init}, Q_{accept}, Q_{reject})$ be a TM. We denote by \mathbf{f} the transition function of M with oracle ϕ . We have that \mathbf{f} is in $\mathbf{PAM} + \Phi$.*

Since $g \in \text{LDL}^\circ + \Phi$, it is computable in second-order polynomial time. Hence, with Theorem 1 and Corollary 1, we have that **DRRL** \in **PSPACE**².

The proofs of hardness rely on dealing with Lipschitz functions. The transition function in a (Type-2) TM is Lipschitz as, according to our encoding, applying one transition implies the encoded real is at most multiplied by 4 and at least divided by $\frac{1}{4}$.

4.2 Discrete-time systems

As in [5], we restrain our dynamic functions to be Lipschitz.

Theorem 6 ([5], Theorem 45). **DRRL** \in **PSPACE**².

The proof of Theorem 6 uses the same idea as the proof of Savitch’s theorem [42], where we reason on the reachability in a graph. We properly discretise the domain of the dynamical system, using the error we allow, and see the “cubes” as the vertices of a graph.

Main theorem: Completeness result

Theorem 7 (Main Theorem I). **DRRL** is **PSPACE**²– \leq_m^2 -hard.

The proof consists in reducing (Definition 7) **Basic PSPACE**² to **DRRL**: The function \mathfrak{s} maps each configuration of M^ϕ to its successor, after one step of M^ϕ , in \mathbb{R} . And the function \mathfrak{t} encodes an input string into a representation of a dyadic number (approximating some real number). A detailed proof can be found in Appendix G.

Corollary 3. **DRRL** is **PSPACE**²– \leq_m^2 -complete.

4.3 Continuous-time systems

Definition 11 (Continuous-time Dynamical Systems). A *continuous-time dynamical system* is a tuple (X, g_h) , with X a set and $g_h : \mathbb{R} \times X \rightarrow X$ such that g_h is the solution of $g_h(0, \mathbf{x}) = y(\mathbf{x})$ and $\frac{\partial g_h(t, \mathbf{x})}{\partial t} = h(t, g_h(t, \mathbf{x}))$.

When $g_h : [0, 1] \rightarrow [-1, 1]$ and $h : [0, 1] \times [-1, 1] \rightarrow \mathbb{R}$ are polynomial-time computable and Lipschitz, then computing g_h is **PSPACE**-complete ([28]). It can be generalised to any compact set. We are interested here in not being restricted to compact sets: we use a principle of dichotomy, as described in [6] and recalled in Definition 10.

We can extend our previous result to *continuous-time* dynamical systems:

Continuous Reach Robust Lipschitz (CRRL):

Input: A continuous-time polynomially robust dynamical system $\mathcal{H}_C = (X_C, g_C)$, with $X_C \subseteq \mathbb{R}$ a closed rational box, $g_C : \mathbb{R} \times X_C \rightarrow X_C \in \text{LDL}^\circ$ Lipschitz, $p \in \mathbb{N}$, $\mathbf{x}, \mathbf{y} \in X_C$

Output: $\mathfrak{R}_{2^{-p}}^{\mathcal{H}_C}(\mathbf{x}, \mathbf{y})$

The Simulation Trick: a Polynomial Improvement For proving the **PSPACE**²-hardness of this problem, we reduce from **DRRL**. We first prove that it is possible to simulate a discrete ODE with a continuous ODE. We rely on a simulation of a discrete dynamic by a continuous ODE attributed to Branicky ([16]). The core idea of this method is, starting from a discrete dynamic,

to construct a continuous ODE whose (smooth) dynamic goes through all the points of the discrete dynamic. We improve this method for our context (Lemma 4).

Definition 12 (Ideal iteration trick [16]). *Consider the following discrete ODE, given by functions \mathbf{g} and \mathbf{u} : $\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x})$; $\frac{\delta \mathbf{f}}{\delta t}(t, \mathbf{x}) = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x})$. Then, let $\mathbf{G}(\mathbf{v}, t, \mathbf{x}) = \mathbf{u}(\mathbf{v}, t, \mathbf{x}) + \mathbf{v}$, and consider the (continuous) ODE:*

$$\begin{cases} \mathbf{y}_1(0, \mathbf{x}) = \mathbf{y}_2(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \\ \mathbf{y}'_1 = c \cdot (\mathbf{G}(r(\mathbf{y}_2), r(t), \mathbf{x}) - \mathbf{y}_1)^3 \theta(\sin(2\pi t)) \\ \mathbf{y}'_2 = c \cdot (r(\mathbf{y}_1) - \mathbf{y}_2)^3 \theta(-\sin(2\pi t)) \end{cases} \quad (1)$$

where c is a constant, $\theta(x) = 0$ if $x \leq 0$ and $\theta(x) > 0$ if $x > 0$. Here, r is a rounding function: we mean, by construction, G preserves the integers, and r is a function that maps a real value close to some integer to this integer: assume, say, that for $z \in [n - \frac{1}{4}, n + \frac{1}{2}]$, $r(z) = n$, for any integer $n \in \mathbb{Z}$.

We write $r(\mathbf{v})$ for applying the function $r : \mathbb{R} \rightarrow \mathbb{R}$ componentwise on the vector \mathbf{v} . We do not need to specify what $r(z)$ values for a z outside such interval: the following reasoning remains correct, whatever it is.

Proposition 4 ([6]). *The solution of the continuous ODE in Definition 12 emulates continuously the associated discrete ODE.*

Intuitively, for f the solution of the discrete ODE and g the solution of the constructed continuous ODE, for any $\mathbf{x}, \mathbf{z} \in \mathbb{R}$, if $f(\mathbf{x}) = \mathbf{z}$ then $g(\mathbf{x}) = \mathbf{z}$. More details about this proposition can be found in Appendix H. To implement such continuous ODEs, we must fix a function $\theta(x)$ with the above property. Taking $\text{ReLU}(x) = \max(0, x)$ would satisfy it, but it is not differentiable, and hence would not lead to an ODE. We could take $\theta(x) = 0$ for $x \leq 0$, $\exp(-\frac{1}{x})$ for $x > 0$. The point is that such a function is not real analytic. The base functions considered in \mathbb{RCD} are all real analytic, and real analytic functions are preserved by composition, so we cannot get such a function by composition from our base functions. Thus, we consider the following continuous approximation on ReLU ([4, Lemma 19]):

Lemma 3 ([4]). *We set $Y(x, 2^{m+2}) = \frac{1 + \tanh(\frac{2^{m+2}x}{2})}{2}$. For all integer m , for all $x \in \mathbb{R}$, $|\text{ReLU}(x) - xY(x, 2^{m+2})| \leq 2^{-m}$, where $\text{ReLU}(x) = \max(0, x)$.*

Having $Y(x, z) = \frac{1 + \tanh(\frac{4xz}{2})}{2}$ yields a function in $\mathbb{RCD}_* = \overline{\mathbb{RCD}} \setminus \{\text{robust continuous ODE, ELim}\}$ with the same property: we avoid the computation of 2^m by a substitution of a variable, and using a multiplication. We write $\text{ReLU-}\mathfrak{s}(Y, x)$ for $xY(x, z)$: we have $|\text{ReLU-}\mathfrak{s}(2^m, x) - \text{ReLU}(x)| \leq 2^{-m}$. Furthermore, a real analytic function that is constant on some interval (we assumed it is 0 for $x \leq 0$) is constant. Hence, the above-considered function $\theta(x)$ cannot be real analytic. So, implementing this trick cannot be done directly using our base functions and only compositions. In Proposition 5, we do a similar construction, but we improve it to deal with errors and not exact functions $\theta(z)$ and $r(x)$. Furthermore, the purpose of the function r is to correct errors around integers, i.e. around \mathbb{Z} : it is possibly around other $\mathbb{Z}\delta = \{n\delta | n \in \mathbb{Z}\}$ for some $\delta > 0$. Then, we assume that, for $z \in [n\delta - \frac{1}{4}\delta, n\delta + \frac{1}{2}\delta]$, $r(z) = n$, for any integer $n \in \mathbb{Z}$, to have a similar reasoning as the one above, where $\delta = 1$.

Proposition 5 (Simulating a discrete ODE by a continuous ODE [6]). *Suppose that, in (1), we replace function $\theta(z)$ and function $r(z)$ by some suitable approximations: we take $\theta(x) = \text{ReLU}(x)$, $\theta_{\epsilon'}(x)$, $r_{\epsilon'}(z)$ such that $\theta(z) =_{\epsilon'} \theta_{\epsilon'}(z)$ and $r(x) =_{\epsilon'} r_{\epsilon'}(x)$, and take constant c big enough. Then the solution of the obtained ODE continuously simulates the discrete ODE, with error at most ϵ if ϵ' is taken sufficiently small. To guarantee $\epsilon = 2^{-n}$, it is sufficient to take $\epsilon' = 2^{-p(n)}$ and $\theta_{\epsilon'}(x) = \text{ReLU}_{\mathfrak{s}}(2^{p(n)}, x)$ for some polynomial p .*

The main idea is that the constructions always replace every function with another not changing much locally (i.e. changes with an error we control). This is the key that provides a robust ODE as in Definition 10, leading to polynomial space complexity. Whenever there is a discrete ODE $\frac{\partial \mathbf{f}}{\partial t}(t, \mathbf{x}) = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x})$ defining some function $\mathbf{f}(t, \mathbf{x})$, one can construct some continuous ODE, using only functions from \mathbb{RCD}_* , such that one of its projections provides $\mathbf{f}(z, t, \mathbf{x})$, where $\mathbf{f}(2^n, t, \mathbf{x}) =_{2^{-n}} \mathbf{f}(n, \mathbf{x})$ whenever $t = \frac{1}{4} n$, for some $n \in \mathbb{N}$. The simulation of Proposition 5 is computable in polynomial time. The sine function is computable in polynomial time (adapting [28]), so is π (consider the Taylor expansion) and the other functions of \mathbb{RCD}_* . **FPTIME** is closed under composition.

We actually improve this result, by expressing the system (1) without using a cosine. The point is to replace $t \mapsto \sin(2\pi t)$ by function \mathbf{y} such that:

$$\begin{cases} \frac{\partial \mathbf{y}}{\partial t}(t) = 2\pi \cdot \mathbf{z}(2\pi t) \\ \frac{\partial \mathbf{z}}{\partial t}(t) = -2\pi \cdot \mathbf{y}(2\pi t) \end{cases} \quad (2)$$

where $\mathbf{y}(0) = 0$ and $\mathbf{z}(0) = 1$. This is a *polynomial* system of ODEs, there solution are 1-periodic. It enables us to avoid the (direct) use of sine and cosine functions in the simulation, and still have an expression for periodic functions. It also simplifies the algebra \mathbb{RCD} : the sine as a base function is no longer necessary. The main point is to make sure the errors remain additive, so the simulation stays in polynomial time.

Lemma 4 (Main result II). *Under the hypotheses of Proposition 5, and replacing $t \mapsto \sin(2\pi t)$ by Equation 2, the solution of the obtained ODE continuously simulates the associated discrete ODE, with error at most ϵ if ϵ' is taken sufficiently small.*

A complete proof can be found in Appendix I: it is done by induction on the intervals of the domain. Then, since polynomial is stable under composition, we have:

Lemma 5. *The simulation of Lemma 4 is computable in polynomial time.*

Corollary 4. $\text{CRRL} \in \text{PSPACE}^2$.

A side-effect of Lemma 4 is that the cosine function is no longer necessary in the algebra \mathbb{RCD} , as it can be expressed with the other components of the algebra:

Theorem 8 (Main theorem II: improvement of [6]). $\overline{\mathbb{RCD}} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}} = \text{FPSPACE} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^d}$, where $\overline{\mathbb{RCD}} = [0, 1, \pi_i^k, +, -, \times, \tanh, \pi, \frac{\pi}{2}, \frac{\pi}{3}; \text{composition, robust continuous ODE, ELim}]$.

Main theorem: Completeness result We can now state and prove the following theorem:

Theorem 9 (Main theorem III). **CRRL** is $\mathbf{PSPACE}^{2-\leq_m^2}$ -hard.

We reduce **DRRL** to **CRRL**, where \mathfrak{t} encode the simulation of Lemma 4 and \mathfrak{s} the successor function (for the discrete ODE). A more formal proof can be found in Appendix J.

Corollary 5. **CRRL** is $\mathbf{PSPACE}^{2-\leq_m^2}$ -complete.

5 Conclusion

We proved **PSPACE**-completeness for **Reach**, under the framework of existing algebraic characterisations of **PSPACE**, defined in [4]. The proof only required the classic notion of reduction and is about classical complexity.

We then proposed a stronger property: we managed to prove that the reachability relation is $\mathbf{PSPACE}^{2-\leq_m^2}$ -complete for robust discrete-time dynamical systems in second-order complexity. We provide those completeness results in a more general framework than in [29], as we use a stronger notion of reduction on more general domains. So now, we have:

	Reach Robust $\mathbb{L}\mathbb{D}\mathbb{L}_{light}^\circ$	DRRL	CRRL
-Completeness	\mathbf{PSPACE} (Cor. 2)	\mathbf{PSPACE}^2 (Cor. 3)	\mathbf{PSPACE}^2 (Cor. 5)

Having such completeness results allows us to have a better understanding of those complexity classes over the reals.

An interesting side-effect is also that we managed to remove the cosine function from the algebra $\overline{\mathbb{RCD}}$, characterising polynomial space for functions over the reals (Theorem 8), getting us closer to a minimal algebra.

Many further works are possible. For example, reasoning on broader definitions of perturbations and *noisy* dynamical systems to include real-world systems would be interesting. More precisely, having completeness results for classical properties (reachability, existence of an attractor, etc.) for more general types of systems would allow a deeper understanding of the upper and lower complexity bounds.

References

- [1] Eugene Asarin and Ahmed Bouajjani. Perturbed Turing machines and hybrid systems. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS-01)*, pages 269–278, Los Alamitos, CA, June 16–19 2001. IEEE Computer Society Press.
- [2] Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65, February 1995.
- [3] Manon Blanc and Olivier Bournez. A characterization of polynomial time computable functions from the integers to the reals using discrete ordinary differential equations. In Jérôme Durand-Lose and György Vaszil, editors, *Machines, Computations, and Universality - 9th International Conference, MCU 2022, Debrecen, Hungary, August 31 - September 2, 2022, Proceedings*, volume 13419 of *Lecture Notes in Computer Science*, pages 58–74. Springer, 2022. URL: https://doi.org/10.1007/978-3-031-13502-6_4, doi:10.1007/978-3-031-13502-6_4.
- [4] Manon Blanc and Olivier Bournez. A characterisation of functions computable in polynomial time and space over the reals with discrete ordinary differential equations: Simulation of turing machines with analytic discrete odes. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPICs*, pages 21:1–21:15. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.MFCS.2023.21.
- [5] Manon Blanc and Olivier Bournez. Quantifying the Robustness of Dynamical Systems. Relating Time and Space to Length and Precision. In Aniello Murano and Alexandra Silva, editors, *32nd EACSL Annual Conference on Computer Science Logic (CSL 2024)*, volume 288 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 17:1–17:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPICs.CSL.2024.17>, doi:10.4230/LIPICs.CSL.2024.17.
- [6] Manon Blanc and Olivier Bournez. The Complexity of Computing in Continuous Time: Space Complexity Is Precision. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 129:1–129:22, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPICs.ICALP.2024.129>, doi:10.4230/LIPICs.ICALP.2024.129.
- [7] Vincent Blondel and John Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

- [8] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer, 1998. URL: <https://www.worldcat.org/oclc/37004484>.
- [9] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers; NP completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, jul 1989.
- [10] Olivier Bournez and Arnaud Durand. Recursion schemes, discrete differential equations and characterization of polynomial time computation. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th Int Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 138 of *LIPICs*, pages 23:1–23:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.MFCS.2019.23.
- [11] Olivier Bournez and Arnaud Durand. A characterization of functions over the integers computable in polynomial time using discrete ordinary differential equations. *Computational Complexity*, 32(2):7, 2023. doi:10.1007/s00037-023-00240-1.
- [12] Olivier Bournez, Riccardo Gozzi, Daniel S Graça, and Amaury Pouly. A continuous characterization of PSPACE using polynomial ordinary differential equations. *Journal of Complexity*, 77:101755, august 2023. URL: <https://www.sciencedirect.com/science/article/pii/S0885064X23000249?dgcid=author>, doi:10.1016/j.jco.2023.101755.
- [13] Olivier Bournez, Daniel Silva Graça, and Amaury 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 Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 109:1–109:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.109.
- [14] Olivier Bournez and Amaury Pouly. A universal ordinary differential equation. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPICs*, pages 116:1–116:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.ICALP.2017.116.
- [15] Olivier Bournez and Amaury Pouly. A survey on analog models of computation. In *Handbook of Computability and Complexity in Analysis*, pages 173–226. Springer, 2021.
- [16] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67–100, 6 feb 1995. doi:10.1016/0304-3975(94)00147-B.

- [17] Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In *New computational paradigms*, pages 425–491. Springer, 2008.
- [18] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html>.
- [19] Pieter Collins. Continuity and computability of reachable sets. *Theoretical Computer Science*, 341(1):162–195, 2005. URL: <https://www.sciencedirect.com/science/article/pii/S0304397505003038>, doi: <https://doi.org/10.1016/j.tcs.2005.05.001>.
- [20] C.C. Conley and Conference Board of the Mathematical Sciences. *Isolated Invariant Sets and the Morse Index*. Regional conference series in mathematics. Conference Board of the Mathematical Sciences, 1978. URL: <https://books.google.fr/books?id=7u9KAQAAAJ>.
- [21] François Fages, Guillaume Le Guludec, Olivier Bournez, and Amaury Pouly. Strong turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In Jérôme Feret and Heinz Koepl, editors, *Computational Methods in Systems Biology - 15th International Conference, CMSB 2017, Darmstadt, Germany, September 27-29, 2017, Proceedings*, volume 10545 of *Lecture Notes in Computer Science*, pages 108–127. Springer, 2017. CMSB’2017 Best Paper Award. URL: https://doi.org/10.1007/978-3-319-67471-1_7, doi:10.1007/978-3-319-67471-1_7.
- [22] Sicun Gao, Jeremy Avigad, and Edmund M Clarke. Delta-decidability over the reals. In *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*, pages 305–314. IEEE, 2012.
- [23] Riccardo Gozzi. *Analog Characterization of Complexity Classes*. PhD thesis, Instituto Superior Técnico, Lisbon, Portugal and University of Algarve, Faro, Portugal, 2022.
- [24] Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. *jcomp*, 19(5):644–664, 2003. doi:10.1016/S0885-064X(03)00034-7.
- [25] Daniel S. Graça and Ning Zhong. *Handbook of Computability and Complexity in Analysis*, chapter Computability of Differential Equations. Springer., 2018.
- [26] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What’s decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, August 1998.
- [27] Morris W. Hirsch, Stephen Smale, and Robert Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Elsevier Academic Press, 2003.

- [28] Akitoshi Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 149–160. IEEE, 2009. doi:10.1109/CCC.2009.34.
- [29] Akitoshi Kawamura and Stephen A. Cook. Complexity theory for operators in analysis. *ACM Trans. Comput. Theory*, 4(2):5:1–5:24, 2012. doi:10.1145/2189778.2189780.
- [30] Akitoshi Kawamura, Norbert Müller, Carsten Rösnick, and Martin Ziegler. Computational benefit of smoothness: Parameterized bit-complexity of numerical operators on analytic functions and gevrey’s hierarchy. *Journal of Complexity*, 31(5):689–714, 2015.
- [31] Akitoshi Kawamura, Hiroyuki Ota, Carsten Rösnick, and Martin Ziegler. Computational complexity of smooth differential equations. *Logical Methods in Computer Science*, 10, 2014. doi:10.2168/LMCS-10(1:6)2014.
- [32] Akitoshi Kawamura, Florian Steinberg, and Holger Thies. Parameterized complexity for uniform operators on multidimensional analytic functions and ode solving. In *International Workshop on Logic, Language, Information, and Computation*, pages 223–236. Springer, 2018.
- [33] Patrick Kidger. On neural differential equations. *CoRR*, abs/2202.02435, 2022. URL: <https://arxiv.org/abs/2202.02435>, arXiv:2202.02435, doi:10.48550/arXiv.2202.02435.
- [34] Ker-I Ko. *Complexity theory of real functions*, volume 3 of *Progress in theoretical computer science*. Birkhäuser, Boston, 1991.
- [35] Pascal Koiran, Michel Cosnard, and Max Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1-2):113–128, September 1994.
- [36] Cristopher Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(3):199–230, 1991.
- [37] Amaury Pouly. *Continuous models of computation: from computability to complexity*. PhD thesis, Ecole Polytechnique and Unidersidade Do Algarve, Defended on July 6, 2015. 2015. <https://pastel.archives-ouvertes.fr/tel-01223284>, Prix de Thèse de l’Ecole Polytechnique 2016, Ackermann Award 2017.
- [38] Amaury Pouly. *Continuous models of computation: from computability to complexity*. Theses, Ecole Doctorale Polytechnique ; Universidad do Algarve, July 2015. URL: <https://pastel.hal.science/tel-01223284>.
- [39] Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10:87–113, 2000.
- [40] Marcus Schaefer, Jean Cardinal, and Tillmann Miltzow. The existential theory of the reals as a complexity class: A compendium. *CoRR*, abs/2407.18006, 2024. URL: <https://doi.org/10.48550/arXiv.2407.18006>, arXiv:2407.18006, doi:10.48550/ARXIV.2407.18006.

- [41] Claude E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.
- [42] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.
- [43] Michael Sipser. *Introduction to the Theory of Computation*. Introduction to the Theory of Computation. Cengage Learning, 2012. URL: <https://books.google.fr/books?id=4J1ZMAEACAAJ>.
- [44] Izumi Takeuti. Effective fixed point theorem over a non-computably separable metric space. In Jens Blanck, Vasco Brattka, and Peter Hertling, editors, *Computability and Complexity in Analysis*, pages 310–322, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [45] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. Reprinted in Martin Davis. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvability Problems and Computable Functions*. Raven Press, 1965. doi:10.1112/plms/s2-42.1.230.
- [46] Bernd Ulmann. *Analog computing*. Walter de Gruyter, 2013.
- [47] Bernd Ulmann. *Analog and hybrid computer programming*. De Gruyter Oldenbourg, 2020.
- [48] Veritasum. Future computers will be radically different (analog computing). Youtube video, 2022. URL: <https://www.youtube.com/watch?v=GVsU0uSjvcg>.
- [49] Klaus Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.

A Little course of computable analysis

A.1 Notions of computable analysis

In classical models of computation, we usually consider an input of finite size, e.g. integers or graphs, and work on some representation on a finite alphabet. Such objects are said to be *computable* when there exists some Turing machine producing their representations.

The main idea behind computable analysis is to be able to talk about objects which usually do not have a finite representation, such as real numbers, functions over the reals, compact sets, etc. A reason why we cannot use classical models is that the set of words on a finite alphabet is countable, while \mathbb{R} is not.

However, they can be represented by some infinite words over a finite alphabet and the idea is to fix such representations for these various objects, called *names*, with suitable computable properties. In particular, in all the following proposed representations, it was proved that an object is computable iff it has some computable representation. We denote by Σ the machine alphabet, Σ^* denotes the finite words on this alphabet (they can be arbitrarily long) and Σ^ω the infinite words on Σ .

For discrete-time models of computations over the reals, the most famous models are computable analysis, based on the Turing machine model (see [45], [49]), and algebraic models such as the Blum Shub Smale (BSS) model of computation [9], [8]. The class $\exists\mathbb{R}$ corresponds to the (constant-free, equivalently uniform) non-deterministic time of the BSS model of computation. Numerous decision problems have been proved recently in this class [40]. Both models were tailored for different applications, and it is well known that we cannot unify existing models with the equivalent of a Church-Turing thesis. For example, computable functions in a computable analysis model must be continuous, whereas the BSS model aims to consider functions and problems over polynomials that are not. The former model has not been introduced to be associated with actual physical machines. Among models of computation over the reals, we can highlight continuous-time models. This includes models of old, first-ever-built computers, such as the Differential Analysers [47]. A famous mathematical model of such machines is the General Purpose Analog Computer model of Claude Shannon [41]. It covers many historical machines and today's analogue devices [46, 48] too. It also includes various recent approaches and models from deep learning, such as Neural ODEs [18, 33] with many variants.

In the context of continuous-time, the situation is clearer than with discrete-time models, as there is a unifying way to describe these models, provided by Ordinary Differential Equations (ODEs). Each model corresponds to a particular class of ODEs. For example, the GPAC corresponds to polynomial ODEs [24], and Neural ODEs are made by selecting the best solution among a parameterised class of ODEs: see, e.g. [33].

When we say that a function $f : S_1 \times \dots \times S_d \rightarrow \mathbb{R}^d$, $S_i \subseteq \mathbb{R}$ for all $i \in [1, d]$, is (respectively: polynomial-time) computable, it will always be in the sense of computable analysis. We recall here the basic concepts and definitions, mostly following the book [34] on complexity theory in computable analysis. Alternative presentations include [17, 49].

NB 1. *The monograph [34] formulates explicitly most of the statements only for functions over the reals. Since we talk about functions in $\mathbb{R}^{\mathbb{N}}$, we need to*

mix complexity issues dealing with integer and real arguments. Hence, we extend some of the statements to these more general settings (in the spirit of [34, 28, 31, 30, 32]).

We present here a formalisation equivalent to [44] but with our own words. This will be used to fix the framework and the notations.

To talk about computable analysis, we consider a variant of Turing machines in this context, namely Type-2 machines:

Definition 13 (Type-2 Turing machine [49]). *A Type-2 Turing machine is a Turing machine with k input and working tapes together with a type specification (Y_1, \dots, Y_k, Y_0) with $Y_i \in \{\Sigma^*, \Sigma^\omega\}$, giving the type (finite or infinite) for each input tape and output tape.*

This machine contains some read-only input tapes, containing the inputs, which can be either finite or infinite words, a read-write working tape and one write-only output tape. It looks like a classical Turing machine, the only difference is that we allow it to output an infinite word written forever on its (or one of its) write-only infinite output tape(s).

Definition 14 (Computable String Function f [49]). *The initial tape configuration for input $(y_1, \dots, y_k) \in Y_1 \times \dots \times Y_k$ is as follows: for each input tape i , the (finite or infinite) sequence $y_i \in Y_i$ is placed on the tape immediately to the right of the head, all other tape cells contain the symbol \mathbf{B} . For all $y_0 \in Y_0, y_1 \in Y_1, \dots, y_k \in Y_k$ we define:*

1. Case $Y_0 = \Sigma^*$: $f(y_1, \dots, y_k) := y_0 \in \Sigma^*$ iff the associated Type-2 Turing machine M halts on input (y_1, \dots, y_k) with y_0 written on the output tape.
2. Case $Y_0 = \Sigma^\omega$: $f(y_1, \dots, y_k) := y_0 \in \Sigma^\omega$ iff M computes forever on input (y_1, \dots, y_k) and writes y_0 on the output tape.

We can now define a proper notion of computable enumerability in that context:

Definition 15 (Computably enumerable sets [49]). *A closed set $A \in \mathbb{R}^n$ is computably enumerable if there exists a Turing machine that enumerates all the open rational cubes intersecting A .*

We can also consider oracle Turing machines to represent functions in $\mathbb{R}^{\mathbb{R}}$, as we will see in Definition 19.

We approximate real numbers with dyadics. A dyadic number d is a rational number with a finite binary expansion. That is to say $d = m/2^n$ for some integers $m \in \mathbb{Z}, n \in \mathbb{N}, n \geq 0$. Let \mathbb{D} be the set of all dyadic rational numbers.

We denote by \mathbb{D}_n the set of all dyadic rationals d with a representation s of precision $\text{prec}(s) = n$: that is, $\mathbb{D}_n = \{m \cdot 2^{-n} \mid m \in \mathbb{Z}\}$.

NB 2. \mathbb{D} is dense in \mathbb{R} .

Thus, we can approximate any real number x with a sequence of dyadics converging to x .

CF_x is a set of Cauchy sequences converging to x :

Definition 16 (Cauchy representation of a real number). *Define the Cauchy representation $\rho_C : \Sigma^\omega \rightarrow \mathbb{R}$ by $\rho_C(p) = x$ iff there are words $w_0, w_1, \dots \in \text{Dom}(\nu_{\mathbb{Q}})$ such that $p = \iota(w_0)\iota(w_1)\dots$, $|\overline{w_i} - \overline{w_k}| \leq 2^{-i}$ for $i < k$ and $x = \lim_{i \rightarrow \infty} \overline{w_i}$.*

It leads us to the definition of computable real numbers:

Definition 17 ([34]). *For each real number x , a function $\phi : \mathbb{N} \rightarrow \mathbb{D}$ is said to binary converge to x if for all $n \in \mathbb{N}$, $\text{prec}(\phi(n)) = n$ and $|\phi(n) - x| \leq 2^{-n}$.*

Thus, we deduce:

Definition 18 (Computable real number [34]). *A real number x is computable iff CF_x contains a computable function: there exists a TM such that, on input $n \in \mathbb{N}$, computes the n first digits of x .*

There exist many more non-computable than computable real numbers. One intuition is to notice that, according to the previous definition, there is an injection between the set of Turing machines and computable reals: the set of computable real numbers is countable, whereas \mathbb{R} is not. However, the set of computable reals being dense in \mathbb{R} , any non-computable real can be approximated by a sequence of computable reals. A *name* of $x \in \mathbb{R}^d$ is a sequence $(I_n)_{n \in \mathbb{N}}$ of nested open rational balls with $I_{n+1} \subseteq I_n$ for all $n \in \mathbb{N}$ and $\{x\} = \bigcap_{n \in \mathbb{N}} I_n$. Such a name can be encoded as an infinite sequence of symbols.

In [4] and [6], the authors are interested in having characterisations of functions computable in **FPTIME** and **FPSPACE**. So, we have to formally define what a computable function is in the framework of computable analysis:

Definition 19 ([34]). *A real function $f : \mathbb{R} \rightarrow \mathbb{R}$ is computable if there is a function-oracle TM 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 the function f is computable on the interval $[a, b]$ if the above condition holds for all $x \in [a, b]$.*

We call a real function $f : \subseteq \mathbb{R} \rightarrow \mathbb{R}$ computable, iff some Type-2 Turing machine maps any name of any $x \in \text{Dom}(f)$ to a name of $f(x)$. For real functions $\mathbf{f} : \subseteq \mathbb{R}^k \rightarrow \mathbb{R}$ we consider machines reading k names in parallel. A computable function is necessarily continuous: see [49] for all details.

The intuition is that an oracle Turing machine M computes a function f over the real, if we have, for any real x in the domain of f , for any $\phi \in CF_x$, M with oracle ϕ , the following: given the output precision 2^{-n} , given in the form of integer n , M eventually output $d \in \mathbb{D}$ with $\|d - f(x)\| \leq 2^{-n}$. During this computation, M asks necessarily finitely many questions to its oracle ϕ . From the considered oracles, any such question can be seen as follows: for some queried precision 2^{-m} , M asks the oracle ϕ to provide some $\phi(m)$, such that $\|\phi(m) - x\| \leq 2^{-m}$.

We notice that, in computable analysis, we only deal with continuous functions:

Theorem 10 ([49]). *Every computable real function is continuous.*

NB 3. *From the model of computable analysis, given the name of \mathbf{f} , $\mathbf{x}, \mathbf{y} \in \mathbb{Q}$, it is impossible in general to tell effectively if $\mathbf{f}(\mathbf{x}) = \mathbf{y}$.*

For closed sets, the notion of computability can be interpreted as the possibility of being plotted with an arbitrarily chosen precision: $\mathbf{z}/2^n$ corresponds to a pixel at precision 2^{-n} , 1 is black (the pixel is plotted black), 0 is white (the pixel is plotted white).

Theorem 11 ([17]). *For a closed set $A \subseteq \mathbb{R}^k$, A is computable iff it can be plotted: there exists a computable function $f : \mathbb{N} \times \mathbb{Z}^k \rightarrow \{0, 1\}$ and such that for all $n \in \mathbb{N}$ and $\mathbf{z} \in \mathbb{Z}^k$*

$$f(n, \mathbf{z}) = \begin{cases} 1 & \text{if } B\left(\frac{\mathbf{z}}{2^n}, 2^{-n}\right) \cap A \neq \emptyset, \\ 0 & \text{if } B\left(\frac{\mathbf{z}}{2^n}, 2 \cdot 2^{-n}\right) \cap A = \emptyset, \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases}$$

Computability and complexity classes over the reals depend on the model of computation we consider, and also the representations of the reals: some complexity results hold for some representation (of the reals) but not for another (see [49]). Here, we consider only equivalent, with respect to the following definition, representations.

Definition 20 (Translation and equivalence, from [49]). *For arbitrary functions $\gamma : Y \rightarrow S$, $\gamma' : Y' \rightarrow S'$ with $Y, Y' \in \{\Sigma^*, \Sigma^\omega\}$ we have: $f : S^Y \rightarrow S'^{Y'}$, f computable, translates γ to γ' iff, for all $y \in \text{Dom}(\gamma)$, $\gamma(y) = \gamma'(f(y))$; and γ is equivalent to γ' iff some computable function translates γ to γ' and some computable function translates γ' to γ .*

A.2 On computable analysis: Complexity

Assume that M is an oracle machine which computes f on the domain $\text{Dom}(f)$.

For simplicity, we may assume in our discussion that $\text{Dom}(f)$ is some closed interval or \mathbb{R} . For any oracle $\phi \in CF_x$, with $x \in \text{Dom}(f)$, let $T_M(\phi, n)$ be the number of steps for M to halt on input n with oracle ϕ , and $T'_M(x, n) = \max\{T_M(\phi, n) \mid \phi \in CF_x\}$.

We define the time complexity of f as follows:

Definition 21 (From [34]). *Let $\text{Dom}(f)$ be bounded closed interval $[a, b]$. Let $f : \text{Dom}(f) \rightarrow \mathbb{R}$ be a computable function. Then, we say that the time complexity of f on $\text{Dom}(f)$ is bounded by a function $t : \text{Dom}(f) \times \mathbb{N} \rightarrow \mathbb{N}$ if there exists an oracle TM M which computes f such that for all $x \in \text{Dom}(f)$ and all $n > 0$, $T'_M(x, n) \leq t(x, n)$.*

In other words, as mentioned in [34], the idea is to measure the time complexity of a function over the reals based on the input and the output precision 2^{-n} . However, it is usually more convenient to simplify the complexity measure and make it depend only on the output precision. They define the uniform time complexity of f on G with a bound function $t' : \mathbb{N} \rightarrow \mathbb{N}$.

However, if we do so, it is critical to realise that if we took $\text{Dom}(f) = \mathbb{R}$ in previous definition, for unbounded functions f , [34] observes that “the uniform time complexity does not exist, because the number of moves required to write down the integral part of $f(x)$ grows as x approaches $+\infty$ or $-\infty$ ”.

Therefore, the approach of [34] is to do as follows (the bounds -2^X and 2^X are somewhat arbitrary, but are chosen here because the binary expansion of any $x \in (-2^n, 2^n)$ has n bits in the integral part):

Definition 22 (Adapted from [34]). *For functions $f(x)$ whose domain is \mathbb{R} , we say that the (non-uniform) time complexity of f is bounded by a function $t' : \mathbb{N}^2 \rightarrow \mathbb{N}$ if the time complexity of f on $[-2^X, 2^X]$ is bounded by a function $t : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that $t(x, n) \leq t'(X, n)$ for all $x \in [-2^X, 2^X]$.*

We extend the approach to more general functions. When $\mathbf{x} = (x_1, \dots, x_p)$ and $\mathbf{X} = (X_1, \dots, X_p)$, we write $\mathbf{x} \in [-2^{\mathbf{X}}, 2^{\mathbf{X}}]$ as short for $x_1 \in [-2^{X_1}, 2^{X_1}]$, \dots , $x_p \in [-2^{X_p}, 2^{X_p}]$.

For $n \in \mathbb{N}$, we denote by $\ell(n)$ the number of bits of the binary representation of n .

Definition 23 (Complexity for real functions: general case, from [34]). *Consider a function $f(x_1, \dots, x_p, n_1, \dots, n_q)$ whose domain is $\mathbb{R}^p \times \mathbb{N}^q$. We say that the (non-uniform) time complexity of f is bounded by a function $t' : \mathbb{N}^{p+q+1} \rightarrow \mathbb{N}$ if the time complexity of $f(\cdot, \dots, \cdot, \ell(n_1), \dots, \ell(n_q))$ on $[-2^{X_1}, 2^{X_1}] \times \dots \times [-2^{X_p}, 2^{X_p}]$ is bounded by a function:*

$$t(\cdot, \dots, \cdot, \ell(n_1), \dots, \ell(n_q), \cdot) : \mathbb{N}^p \times \mathbb{N} \rightarrow \mathbb{N}$$

such that $t(\mathbf{x}, \ell(n_1), \dots, \ell(n_q), n) \leq t'(\mathbf{X}, \ell(n_1), \dots, \ell(n_q), n)$ whenever $\mathbf{x} \in [-2^{\mathbf{X}}, 2^{\mathbf{X}}]$. We say that f is polynomial time computable if t' can be chosen as a polynomial. We say that a vectorial function is polynomial time computable iff all its components are.

The motivation behind those previous definitions is similar to the one in [37] and [13]: we want this measure of complexity to extend the usual complexity for functions over the integers, where the complexity of integers is measured with respect to their lengths, and over the reals, where complexity is measured with respect to their approximation.

In particular, in the specific case of a function $f : \mathbb{N}^d \rightarrow \mathbb{R}^d$, it means there is some polynomial $t' : \mathbb{N}^{d+1} \rightarrow \mathbb{N}$ such that the time complexity of producing some dyadic approximating $f(\mathbf{m})$ at precision 2^{-n} is bounded by $t'(\ell(m_1), \dots, \ell(m_d), n)$, writing $\mathbf{m} = (m_1, \dots, m_d)$.

A key observation is the stability of polynomial time under composition:

Lemma 6 ([34]). *The class of polynomial time computable functions is stable under composition.*

There exist various characterisations of functions computable in polynomial time over the reals. For example, the following is proved in [34] for functions from $[a, b] \rightarrow \mathbb{R}$. It could be extended to more general functions.

Theorem 12 (Alternative characterisation [34]). *A function $f : [a, b] \rightarrow \mathbb{R}$ is computable in polynomial time iff there exist polynomial functions m and q and a function $\psi : (\mathbb{D} \cap [a, b]) \times \mathbb{N} \rightarrow \mathbb{D}$ such that*

1. *m is a polynomial modulus function for f on $[a, b]$: $|x - y| \leq 2^{-m(n)}$ implies $|f(x) - f(y)| \leq 2^{-n}$ for every $x, y \in [a, b]$ and every integer n ;*
2. *for any $d \in \mathbb{D} \cap [a, b]$ and all $n \in \mathbb{N}$, $|\psi(d, n) - f(d)| \leq 2^{-n}$;*
3. *$\psi(d, n)$ is computable in time $q(\ell(d) + n)$.*

B Classical Computability and Complexity for Dynamical Systems

Assuming f preserves the rationals ($f(\mathbb{Q}) \subseteq \mathbb{Q}$), we have:

	$X \subseteq \mathbb{Q}$, f computable	$X \subseteq \mathbb{Q}$, f PAM
Reach (\mathbf{x}, y)	c.e.-complete	c.e.-complete
Reach (\mathbf{x}, Y)	c.e.-complete	c.e.-complete

To prove those hardness results, it is sufficient to reduce from the Halting problem. See e.g. [36, 26, 38] for the proofs.

Notice that the complexity of the problem crucially depends on the dynamical systems we consider:

	$X \subseteq \mathbb{Q}$, $f \in \mathbf{FPTIME}$	$X \subseteq \mathbb{Q}$, $f \in \mathbf{FPSPACE}$
Reach (\mathbf{x}, y)	PTIME -complete	PSPACE -complete
Reach (\mathbf{x}, Y)	PTIME -complete	PSPACE -complete

where we assume time to be given in unary and **Reach** to be time-bounded, meaning the computation time does not exceed some given $t \in \mathbb{N}$. To prove those results, it is sufficient to reduce from the time-bounded Halting Problem.

C Length ODEs

Length ODEs are a scheme of discrete ODEs, where the derivative is not with steps of 1, as for classical discrete ODEs, but of the successive powers of 2. More comprehensive definitions can also be found in [10].

Definition 24 (Length-ODE [10, 11]). *A function $\mathbf{f} : \mathbb{N} \times \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$, $d, d' \in \mathbb{N}$, is length-ODE definable if it corresponds to the solution of*

$$\mathbf{f}(0, \mathbf{y}) = \mathbf{g}(\mathbf{y}) \quad \text{and} \quad \frac{\delta \mathbf{f}(x, \mathbf{y})}{\delta \ell(x)} = \mathbf{u}(\mathbf{f}(x, \mathbf{y}), \mathbf{h}(x, \mathbf{y}), x, \mathbf{y}) \quad (3)$$

So $\frac{\delta \mathbf{f}}{\delta \ell(x)}(n, \mathbf{y}) \neq 0$ when $n = 2^m - 1$ for some $m \in \mathbb{N}$. The latter is a change of variable on the former: we differentiate with respect to the binary length of an integer x (its number of bits), instead of x itself. It crucially links the size of a given representation for real numbers and the complexity class its computation is in.

For polynomial time we consider *Linear* length-ODE, for the following notion of linearity:

Definition 25 (Essential linearity). *$\mathbf{g}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y})$ is essentially linear in $\mathbf{f}(x, \mathbf{y})$ if it is of the form: $\mathbf{g}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y}) = \mathbf{A}[\mathbf{f}(x, \mathbf{y}), x, \mathbf{y}] \cdot \mathbf{f}(x, \mathbf{y}) + \mathbf{B}[\mathbf{f}(x, \mathbf{y}), x, \mathbf{y}]$ where \mathbf{A} and \mathbf{B} depend on some sigmoid on $\mathbf{f}(x, \mathbf{y})$.*

Thus, a length ODE is linear when \mathbf{u} (in Equation 3) is *essentially linear* in \mathbf{f} , $\mathbf{g} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ and $\mathbf{h} : \mathbb{N} \times \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$.

A fundamental fact is that the derivation with respect to the length provides a way to make a change of variables:

Lemma 7 ([10, 11]). *Assume that (3) holds. Then $\mathbf{f}(x, \mathbf{y})$ is given by $\mathbf{f}(x, \mathbf{y}) = \mathbf{F}(\ell(x), \mathbf{y})$ where \mathbf{F} is the solution of the initial value problem*

$$\mathbf{F}(1, \mathbf{y}) = \mathbf{g}(\mathbf{y}), \quad \text{and} \quad \frac{\delta \mathbf{F}(t, \mathbf{y})}{\delta t} = \mathbf{u}(\mathbf{F}(t, \mathbf{y}), \mathbf{h}(2^t - 1, \mathbf{y}), 2^t - 1, \mathbf{y}) \quad (4)$$

This means $\mathbf{f}(x, \mathbf{y})$ depends only on the length of its first argument: $\mathbf{f}(x, \mathbf{y}) = \mathbf{f}(2^{\ell(x)}, \mathbf{y})$. Then (4) can be seen as defining a function (with this latter property) by a recurrence of type

$$\mathbf{f}(2^0, \mathbf{y}) = \mathbf{g}(\mathbf{y}), \quad \text{and} \quad \mathbf{f}(2^{t+1}, \mathbf{y}) = \bar{\mathbf{u}}(\mathbf{f}(2^t, \mathbf{y}), \mathbf{h}(2^t - 1, \mathbf{y}), 2^t, \mathbf{y}). \quad (5)$$

for $\bar{\mathbf{u}}$ essentially linear in $\mathbf{f}(2^t, \mathbf{y})$. As recurrence (4) is equivalent to (3):

Corollary 6 (Linear length ODE presented with powers of 2). *A function \mathbf{f} is linear length ODE definable iff the value of $\mathbf{f}(x, \mathbf{y})$ depends only on the length of its first argument and satisfies (5), for some \mathbf{g} and \mathbf{h} , and $\bar{\mathbf{u}}$, essentially linear in $\mathbf{f}(2^t, \mathbf{y})$.*

D Proof of Lemma 2, from [4]

Before giving the proof of the lemma, we admit the following:

Lemma 8 ([4]). *Let $N \in \mathbb{N}$, $\alpha_1, \alpha_2, \dots, \alpha_n \in \mathbb{N}^n$, pairwise distinct, and, for $1 \leq i \leq n$ and $1 \leq j \leq N$, let $V_{i,j}$ be some constants. Then there exists some function*

$$\mathcal{C}\text{-send}(2^m, (\alpha_i, j) \mapsto V_{i,j})_{i \in \{1, \dots, n\}, j \in \{1, \dots, N\}}$$

in $\mathbb{L}\mathbb{D}\mathbb{L}^\circ$, such as, for all $i \in \{1, \dots, n\}$, $j \in \{1, \dots, N\}$, it maps any $x \in [\alpha_i - 1/4, \alpha_i + 1/4]$ and any $y \in [j - 1/4, j + 1/4]$ to a real number at a distance at most 2^{-m} of $V_{i,j}$.

Proof of Lemma 2 from [4]. We can write $l = l_0 l^\bullet$ and $r = r_0 r^\bullet$, respectively the left part of the tape with respect to the head and the right part of the tape, where l_0 and r_0 are the first letters of l and r , and l^\bullet and r^\bullet corresponding to the (possibly infinite) word $l_{-1} l_{-2} \dots$ and $r_1 r_2 \dots$ respectively. Notice that l is written in reverse order on the tape.

$$\begin{array}{cccccc} \dots & l^\bullet & l_0 & r_0 & r^\bullet & \dots \\ \hline & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & & \\ & l & & r & & \end{array}$$

Consider the transition function working over the representation of the configurations as reals, considering $r_0 = \lfloor 4\bar{r} \rfloor$:

$$\begin{aligned} \overline{\text{Next}}(q, \bar{l}, \bar{r}) &= \overline{\text{Next}}(q, \bar{l}^\bullet l_0, \bar{r}_0 r^\bullet) = (q', \bar{l}', \bar{r}') \\ &= \begin{cases} (q', \bar{l}^\bullet l_0 x, \bar{r}^\bullet) & \text{whenever } \delta(q, r_0) = (q', x, \rightarrow) \\ (q', \bar{l}^\bullet, \bar{l}_0 x r^\bullet) & \text{whenever } \delta(q, r_0) = (q', x, \leftarrow) \end{cases} \end{aligned}$$

- in the first case “ \rightarrow ” : $\bar{l}' = 4^{-1} \bar{l} + 4^{-1} x$ and $\bar{r}' = \bar{r}^\bullet = \{4\bar{r}\}$
- in the second case “ \leftarrow ” : $\bar{l}' = \bar{l}^\bullet = \{4\bar{l}\}$ and $\bar{r}' = 4^{-2} \{4\bar{r}\} + 4^{-2} x + \lfloor 4\bar{l} \rfloor / 4$

(6)

We introduce the following functions, corresponding respectively to the head moves to the right and to the head moves to the left:

$$\begin{aligned} \rightarrow: Q \times \{0, 1, 3\} &\mapsto \{0, 1\} & \leftarrow: Q \times \{0, 1, 3\} &\mapsto \{0, 1\} \\ (q, a) &\mapsto 1 \quad \text{if } \delta(q, a) = (\cdot, \cdot, \rightarrow) & (q, a) &\mapsto 1 \quad \text{if } \delta(q, a) = (\cdot, \cdot, \leftarrow) \\ (q, a) &\mapsto 0 \quad \text{otherwise} & (q, a) &\mapsto 0 \quad \text{otherwise} \end{aligned}$$

We can rewrite $\overline{Next}(q, \bar{l}, \bar{r}) = (q', \bar{l}', \bar{r}')$ as $\bar{l}' = \sum_{q, r_0} \left[\rightarrow (q, r_0) \left(\frac{\bar{l}}{4} + \frac{x}{4} \right) + \leftarrow (q, r_0) \{4\bar{l}\} \right]$

and $\bar{r}' = \sum_{q, r_0} \left[\rightarrow (q, r_0) \{4\bar{r}\} + \leftarrow (q, r_0) \left(\frac{\{4r\}}{4^2} + \frac{x}{4^2} + \frac{\lfloor 4\bar{l} \rfloor}{4} \right) \right]$, and, using notation of Lemma 8, $q' = \text{send}((q, r) \mapsto \text{next}q_r^q)_{q \in Q, r \in \{0, 1, 3\}}(q, \lfloor 4\bar{r} \rfloor)$.

Then, following the intuition of having continuous approximations of discontinuous functions, we can replace $\lfloor 4\bar{r} \rfloor$ by $\sigma(4\bar{r})$ if we take σ as some continuous function that would be affine and values respectively 0, 1 and 3 on $\{0\} \cup [1, 2] \cup [3, 4]$ (that is to say matches $\lfloor 4\bar{r} \rfloor$ on this domain). A possible candidate is $\sigma(x) = \mathfrak{s}(1/4, 3/4, x) + \mathfrak{s}(9/4, 11/4, x)$, with $\mathfrak{s}(a, b, \cdot)$ the sigmoid valuing 0 before a and 1 after b . Then considering $\xi(x) = x - \sigma(x)$, then $\xi(4\bar{r})$ would be the same as $\{4\bar{r}\}$: that is, considering $r_0 = \sigma(4\bar{r})$, replacing in the above expression every $\{4\cdot\}$ by $\xi(\cdot)$, and every $\lfloor \cdot \rfloor$ by $\sigma(\cdot)$, and get something that would still work the same, but using only continuous functions.

Now, we would like to use analytic functions, and not only continuous functions. It is well-known that an analytic function that equals some affine function on some interval (e.g. on $[1, 2]$) must be affine, and hence cannot be 3 on $[3, 4]$. We introduce $\mathcal{C}\text{-}\mathfrak{s}(m, a, b, x) \in \mathbb{L}\mathbb{D}\mathbb{L}^\circ$ such that for all integer m ,

$$|\mathcal{C}\text{-}\mathfrak{s}(m, a, b, x) - \mathfrak{s}(a, b, x)| \leq 2^{-m}.$$

The function $\mathcal{C}\text{-}\mathfrak{s}$ allows us to tolerate errors by replacing $\mathfrak{s}(\cdot, \cdot)$ by $\mathcal{C}\text{-}\mathfrak{s}(2^{m+c}, \cdot, \cdot)$ in the expressions above for σ and ξ , taking c such that $(3 + 1/4^2)3|Q| \leq 2^c$. This would just introduce some error at most $(3 + 1/4^2)3|Q|2^{-c}2^{-m} \leq 2^{-m}$.

We can also replace every $\rightarrow (q, r)$ in above expressions for \bar{l}' and \bar{r}' by

$$\mathcal{C}\text{-send}(k, (q, r) \mapsto \rightarrow (q, r))(q, \sigma(4\bar{r})),$$

for a suitable error bound k , and symmetrically for $\leftarrow (q, r)$. However, if we do so, we still might have some multiplications in the above expressions. To avoid that, we give a continuous approximation in $\mathcal{C}\text{-if} \in \mathbb{L}\mathbb{D}\mathbb{L}^\circ$ of the conditional sigmoid function $\text{if}(\cdot, \cdot, \cdot)$:

- if we take $|d' - 0| \leq 1/4$, then $|\mathcal{C}\text{-if}(d', \ell) - 0| \leq 2^{-m}$,
- and if we take $|d' - 1| \leq 1/4$, then $|\mathcal{C}\text{-if}(d', \ell) - \ell| \leq 2^{-m}$.

We can then also write the above expressions as

$$\begin{aligned} \bar{l}' = \sum_{q, r} &\left[\mathcal{C}\text{-if} \left(2^{m+c}, \mathcal{C}\text{-send}(2^2, (q, r) \mapsto \rightarrow (q, r))(q, \sigma(4\bar{r})), \frac{\bar{l}}{4} + \frac{x}{4} \right) \right. \\ &\left. + \mathcal{C}\text{-if} \left(2^{m+c}, \mathcal{C}\text{-send}(2^2, (q, r) \mapsto \leftarrow (q, r))(q, \sigma(4\bar{r})), \xi(4\bar{l}) \right) \right] \end{aligned}$$

$$\bar{r}' = \sum_{q,r} \left[\mathcal{C}\text{-if} \left(2^{m+c}, \mathcal{C}\text{-send}(2^2, (q, r) \mapsto \rightarrow (q, r))(q, \sigma(4\bar{r})), \xi(4\bar{r}) \right) \right. \\ \left. + \mathcal{C}\text{-if} \left(2^{m+c}, \mathcal{C}\text{-send}(2^2, (q, r) \mapsto \leftarrow (q, r))(q, \sigma(4\bar{r})), \frac{\xi(4r)}{4^2} + \frac{x}{4^2} + \frac{\sigma(4\bar{l})}{4} \right) \right]$$

and still have the same bound on the error. \square

E Proof of Proposition 2 (Main Result I)

Proof of Proposition 2. Let $M = (Q, \Sigma, \Sigma', \delta_M, q_0, Q_{accept}, Q_{reject})$ be a **PSPACE** TM, working on the rationals, and $\mathbf{x}, \mathbf{y} \in \mathbb{Q}$. We construct a discrete-time dynamical system $\mathcal{H} = (X, g)$ such that:

- $X \subseteq \mathbb{R}$ such that $X \cap \mathbb{Q}$ is the space containing all the possible rationals that can be represented by all the configurations of M . A rational is represented by a pair of integers;
- $g : X \rightarrow X$ such that $g(c) = c'$ iff $\delta_M(C) = C'$, $c, c' \in X$, for c, c' rationals encoded by configurations C and C' of M .

Some points of X may not correspond to a number representable in M . We assume that if dynamical reaches such a point z , then it is not associated with a valid configuration of M and $g(z) = \gamma_{config}(C_{reject})$ with C_{reject} the rejecting configuration on M .

By the construction of the execution of a **PSPACE** TM given in the proof of Lemma 2, we have $g \in \mathbb{L}\mathbb{D}\mathbb{L}_{\text{light}}^{\circ}$ and the reduction is computable in polynomial time. If $M(\bar{\mathbf{x}})$ reaches $\bar{\mathbf{y}}$ ($\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ encoded \mathbf{x} and \mathbf{y}), then \mathcal{H} reaches \mathbf{y} , starting from \mathbf{x} . If there exists $m \in \mathbb{N}$ such that $g^m(\mathbf{x}) = \mathbf{y}$, then, by construction, $M(\bar{\mathbf{x}})$ reaches $\bar{\mathbf{y}}$ in m steps. We conclude. \square

F Proof of Theorem 5, from [3, 4]

Proof. We consider a Turing machine $M = (Q, \{0, 1, 3\}, q_{init}, \delta, F)$. Up to renaming, we assume $Q = \{0, 1, \dots, |Q| - 1\} \subset \mathbb{N}$. Let

$$\dots l_{-k} l_{-k+1} \dots l_{-1} l_0 r_0 r_1 \dots r_n \dots$$

denote the content of the tape of the Turing machine M . In this representation, the head is in front of symbol r_0 , and $l_i, r_i \in \{0, 1, 3\}$ for all i . Furthermore, if the blank symbol $B = 0$ appears at some letter r_i then it appears at letter r_j for all $j \geq i$; if the blank symbol $B = 0$ appears at some letter l_{-i} then it appears at letter l_{-j} for all $j \geq i$.

The idea is that such a configuration C can also be encoded by some element $\bar{C} = \gamma_{config}(C) = (q, \bar{l}, \bar{r}) \in \mathbb{N} \times \mathbb{R}^2$, by considering $\bar{r} = \gamma_{word}(r)$ and $\bar{l} = \gamma_{word}(l)$.

Denoting the image of $\gamma_{word} : \Sigma^\omega \rightarrow \mathbb{R}$ by \mathcal{I} , this even lives in $\mathbb{Q} \times \mathcal{I}^2$.

NB 4. Notice that \mathcal{I} is a Cantor-like set: it corresponds to the rational numbers that can be written using only 1 and 3 in base 4. We write \mathcal{I}_S for those with at most S digits after the point (i.e. of the form $n/4^S$ for some integer n).

We denote by δ the transition function: given some control state $q \in Q$, and some symbol r_0 , $\delta(q, r_0) = (q', x, m)$ indicates the new control state $q' \in Q$, the

symbol x to be written, and some movement $m \in \{\leftarrow, \rightarrow\}$, corresponding to shifting left or right. For practical reasons, we assume that when M reaches a final state, it stays in the same configuration.

A configuration C of such TM can be denoted by $C = (q, l, r)$, where

$$\begin{aligned}\bar{r} &= r_0 r_1 \cdots r_n \cdots, \\ \bar{l} &= l_0 l_{-1} \cdots l_{-k} \cdots\end{aligned}$$

are words over alphabet $\Sigma = \{0, 1, 3\}$ and where $q \in Q$ denotes the internal state of M . Notice that r , the right part, is written from left to right (in the usual way), while l , the left part, is written from right to left (in a mirror way). We work with infinite words, but as expected, this covers the case of finite words. Namely, given a finite word $w = w_1 w_2 \dots w_n$ on the tape, in control state q , with the head in front of symbol $1 \leq i \leq n$, corresponds to the configuration $C = (q, w_{i-1} w_{i-2} \dots w_1 0^\omega, w_i w_{i+1} \dots w_n 0^\omega)$, that is to say, with eventually the blank symbol $B = 0$.

The idea is that such a configuration C can also be encoded by some element $\gamma_{\text{config}}(C) = (q, \bar{l}, \bar{r}) \in \mathbb{N} \times \mathbb{R}^2$, by considering

$$\begin{aligned}\bar{r} &= r_0 4^{-1} + r_1 4^{-2} + \cdots + r_n 4^{-(n+1)} + \cdots, \\ \bar{l} &= l_0 4^{-1} + l_{-1} 4^{-2} + \cdots + l_{-k} 4^{-(k+1)} + \cdots\end{aligned}$$

In other words, we encode the configuration of a bi-infinite tape Turing machine M by real numbers using their radix 4 encoding, but using only digits $\{0, 1, 3\}$. As digit 0 is used only at the end of the words, this corresponds to some rational numbers for a finite word, while for an infinite word, this corresponds to some real numbers, possibly irrational. If we write: $\gamma_{\text{word}} : \Sigma^\omega \rightarrow \mathbb{R}$ for the function that maps word $w = w_0 w_1 w_2 \cdots$ to

$$\gamma_{\text{word}}(w) = w_0 4^{-1} + w_1 4^{-2} + \cdots + w_n 4^{-(n+1)} + \cdots,$$

we can also write: $\gamma_{\text{config}}(C) = \gamma_{\text{config}}(q, l, r) = (q, \gamma_{\text{word}}(l), \gamma_{\text{word}}(r))$.

Here, the transition function δ leads naturally to a function $\bar{\delta} : C \mapsto C'$ that maps a configuration $C = (q, l, r)$ to a configuration $C' = (q', l', r')$ after one step in M . Via the above encoding, this can be seen as a function $\mathbf{f} : \mathbb{N} \times \mathbb{R}^2 \mapsto \mathbb{N} \times \mathbb{R}^2$.

Notice that \mathbf{f} takes its values in $Q \times [0, 1]^2$ and it is a piecewise affine map. Actually, if we denote the image of $\gamma_{\text{word}} : \Sigma^\omega \rightarrow \mathbb{R}$ by \mathcal{I} , it even takes its values in $Q \times \mathcal{I}^2$.

It remains to prove that \mathbf{f} is a PAM:

The function \mathbf{f} is basically of the form

$\mathbf{f}(q, l, r) = \mathbf{f}(q, l_0 l^\bullet, r_0 r^\bullet) = (q', l', r')$ defined as a definition by case of type:

$$(q', l', r') = \begin{cases} (q', x l_0 l^\bullet, r^\bullet) & \text{whenever } \delta(q, r_0) = (q', x, \rightarrow) \\ (q', l^\bullet, l_0 x r^\bullet) & \text{whenever } \delta(q, r_0) = (q', x, \leftarrow) \end{cases}$$

This rewrites as a function $\bar{\mathbf{f}}$ which is similar, working over the representation of the configurations as reals, considering $r_0 = \lfloor 4\bar{r} \rfloor$

$$\begin{aligned}\bar{\mathbf{f}}(q, \bar{l}, \bar{r}) &= \bar{\mathbf{f}}(q, \overline{l_0 l^\bullet}, \overline{r_0 r^\bullet}) = (q', \bar{l}', \bar{r}') \\ &= \begin{cases} (q', \overline{x l_0 l^\bullet}, \overline{r^\bullet}) & \text{whenever } \delta(q, r_0) = (q', x, \rightarrow) \\ (q', \bar{l}^\bullet, \overline{l_0 x r^\bullet}) & \text{whenever } \delta(q, r_0) = (q', x, \leftarrow) \end{cases}\end{aligned}$$

where $r_0 = \lfloor 4\bar{r} \rfloor$ and

- in the first case “ \rightarrow ” : $\bar{l}' = 4^{-1}\bar{l} + 4^{-1}x$ and $\bar{r}' = \bar{r}^\bullet = \{4\bar{r}\}$
 - in the second case “ \leftarrow ” : $\bar{l}' = \bar{l}^\bullet = \{4\bar{l}\}$ and $\bar{r}' = 4^{-2}\bar{r}^\bullet + 4^{-2}x + 4^{-1}\lfloor 4\bar{l} \rfloor$
- (7)

We introduce the following functions:

$$\begin{aligned} \rightarrow: Q \times \{0, 1, 3\} &\mapsto \{0, 1\} \\ (q, a) &\mapsto 1 \quad \text{if } \delta(q, a) = (\cdot, \cdot, \rightarrow) \\ (q, a) &\mapsto 0 \quad \text{otherwise} \end{aligned}$$

corresponding to the head moving to the right, and

$$\begin{aligned} \leftarrow: Q \times \{0, 1, 3\} &\mapsto \{0, 1\} \\ (q, a) &\mapsto 1 \quad \text{if } \delta(q, a) = (\cdot, \cdot, \leftarrow) \\ (q, a) &\mapsto 0 \quad \text{otherwise} \end{aligned}$$

corresponding to the head moving to the left.

Hence, we can rewrite, $\bar{\mathbf{f}}(q, \bar{l}, \bar{r}) = (q', \bar{l}', \bar{r}')$ as

$$\bar{l}' = \sum_{q, r_0} \left[\rightarrow (q, r_0) \left(\frac{\bar{l}}{4} + \frac{x}{4} \right) + \leftarrow (q, r_0) \{4\bar{l}\} \right]$$

and

$$\bar{r}' = \sum_{q, r_0} \left[\rightarrow (q, r_0) \{4\bar{r}\} + \leftarrow (q, r_0) \left(\frac{\{4\bar{r}\}}{4^2} + \frac{x}{4^2} + \frac{\lfloor 4\bar{l} \rfloor}{4} \right) \right].$$

And $\{\cdot\}$ and $\lfloor \cdot \rfloor$ are PAMs. □

The embedding of configurations of TM into reals is $\Upsilon(C) = \gamma_{\text{config}}(C)$. But, without loss of generality, we could have taken $\Upsilon(q, w_1, w_2) = (q, \gamma(w_1), \gamma(w_2))$ with $\gamma: \Sigma^* \rightarrow \mathbb{R}$ taken as:

- the encoding $\gamma_{\mathbb{N}}$ mapping the word $w = a_1 \dots a_n$ to the integer whose binary expression is w ,
- or $\gamma_{[0,1]}$ mapping w to the real number of $[0, 1]$ whose binary expansion is w ,
- or more generally, $\gamma_{[0,1]}^k$ or $\gamma_{\mathbb{N}}^k$, using base k instead of base 2 for $k \geq 2$,
- or $\gamma'_{[0,1]}^k$ mapping w to $(\gamma_{[0,1]}^k(w), \ell(w))$.

In any case, we can then consider a function $\mathbf{f}: X \subseteq \mathbb{R}^d \rightarrow X$ such that for any configuration C , denoting by C' the next configuration, $\mathbf{f}(\Upsilon(C)) = \Upsilon(C')$: one step of the TM corresponds to one step in the dynamical system (X, \mathbf{f}) with respect to γ . Then, we have that the diagram of the execution commutes for any number of steps:

$$\begin{array}{ccccccc}
C & \xrightarrow{\vdash} & C' & \xrightarrow{\vdash} & C'' & \xrightarrow{\vdash} & C''' \dashrightarrow \\
\Upsilon \downarrow & & \Upsilon \downarrow & & \Upsilon \downarrow & & \Upsilon \downarrow \\
\Upsilon(C) & \xrightarrow{\mathbf{f}} & \Upsilon(C') & \xrightarrow{\mathbf{f}} & \Upsilon(C'') & \xrightarrow{\mathbf{f}} & \Upsilon(C''') \dashrightarrow
\end{array}$$

We can write $l = l_0 l^\bullet$ and $r = r_0 r^\bullet$, respectively the left part of the tape with respect to the head and the right part of the tape, where l_0 and r_0 are the first letters of l and r , and l^\bullet and r^\bullet corresponding to the (possibly infinite) word $l_{-1} l_{-2} \dots$ and $r_1 r_2 \dots$ respectively. Notice that l is written in reverse order on the tape. The questions related to the existence of trajectories in the dynamical

$$\begin{array}{ccccccc}
\dots & l^\bullet & l_0 & r_0 & r^\bullet & \dots \\
\hline
& \underbrace{\hspace{2cm}}_l & & & \underbrace{\hspace{2cm}}_r & &
\end{array}$$

system (X, \mathbf{f}) associated with the TM lead to questions about the existence of suitable trajectories in the TM. Specifically, it provides c.e.-hardness of reachability for various classes of dynamical systems. Call such a situation a *step-by-step* emulation. However, encodings such as $\gamma_{[0,1]}$, whose image is compact, or on γ_{word} as above, map intrinsically distinct configurations to points arbitrarily close to each other (a sequence over a compact must have some accumulation point). Encodings like $\gamma_{\mathbb{N}}$ do not have a compact image but involve emulations with arbitrarily large integers, which is another issue. These observations led to the already mentioned (informal) conjecture that undecidability/hardness may hold only for non-robust systems. This leads to formally discussing robustness issues for general dynamical systems over \mathbb{R}^d , for some $d \in \mathbb{N}$. These statements and underlying constructions, which allow the simulation of Turing machines, led to solving several open problems: the existence of a universal ODE [14], the proof of the Turing-completeness of chemical reactions [21], or statements about the hardness of several dynamical systems problems [25].

G Proof of Theorem 7 (Main Theorem I)

Proof of Theorem 7. We reduce **Basic PSPACE**² to **DRRL**. Let M, μ, ϕ and u be the inputs of **Basic PSPACE**². Without loss of generality, we assume $M^\phi = (Q, \Sigma, \mathbf{f}, q_{init}, Q_{accept}, Q_{reject})$ has a unique accepting state q_{accept} with $0^{\mu(|u|)}$ being the content of the tape. We also assume that, if $\mathbf{f}(C)$, for a configuration C , is not defined, then M^ϕ goes to a rejecting state. \mathbf{f} remains a PAM with this assumption. We construct the inputs of **DRRL** from the inputs of **Basic PSPACE**² and define the functions \mathfrak{s} and \mathfrak{t} from Definition 7:

- For C_0 the initial configuration of M^ϕ on u , we define $\mathbf{x} = \gamma_{config}(C_0)$.
- For C_{f_a} the (unique) final accepting configuration of M , $\mathbf{y} = \gamma_{config}(C_{f_a})$.
- $\mathfrak{t} : \langle M, \mu, \phi \rangle \rightarrow X, g, p$, where X is the set of all the real numbers encoded by possible configurations of M^ϕ , $g \in X^X$ such that $\mathbf{f}(C) = C'$

iff $g(\gamma_{\text{config}}(C)) = \gamma_{\text{config}}(C')$, $p = \mu(|u|)$ is the polynomial bounding the space.

- $\mathfrak{s} : \text{Reg} \rightarrow \mathbb{R}^{\mathbb{R}}$ such that $\mathfrak{s} : \phi \rightarrow \Phi$.

The function \mathfrak{s} maps each configuration of M^ϕ to its successor, after one step of M^ϕ , in \mathbb{R} . It is Lipschitz and it is computable in polynomial time by Corollary 1. The function \mathfrak{t} encodes an input string into a representation of a dyadic number (approximating some real number). It is computable in polynomial time by Lemma 1. Thus, we have **DRRL** is $\mathbf{PSPACE}^2_{\leq m}$ -hard. \square

H Details and proofs about Proposition 4

Indeed, the idea is that \mathbf{y}_1 corresponds to the actual computation of the iterates of \mathbf{G} (and hence computes the successive values of \mathbf{f}) and \mathbf{y}_2 acts as a “memory” equation. Let us detail how it works.

NB 5. *We describe here an “ideal” computation, as $\theta(x)$ is exactly 0 when $x \leq 0$, and $r(z)$ is exactly some integer on suitable domains. Later in the paper, we will deal with a not-so-ideal θ and r .*

Initially, $\mathbf{f}(0, \mathbf{x}) = \mathbf{y}_1(0, \mathbf{x}) = \mathbf{y}_2(0, \mathbf{x}) = \mathbf{g}(\mathbf{x})$. For $t \in [0, 1/2]$, we have

$$\theta(-\sin(2\pi t)) = 0,$$

and hence $\mathbf{y}'_2 = 0$, so \mathbf{y}_2 is fixed and kept at value $\mathbf{g}(\mathbf{x})$ for $t \in [0, \frac{1}{2}]$. Consequently, for $t \in [0, \frac{1}{2}]$, $r(\mathbf{y}_2)$ is also fixed and kept at value $\mathbf{g}(\mathbf{x})$, and $r(t)$ is also fixed and kept at value 0. Consequently, on this interval, if we write $C(t) = c\theta(\sin(2\pi t))$, then the dynamics of \mathbf{y}_1 is given by

$$\mathbf{y}'_1 = C(t) \left(\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) - \mathbf{y}_1 \right)^3 \quad (8)$$

Then, a key is to analyse such an ODE:

Lemma 9 (Analysis of ODE (8) [6]). *The solution $\mathbf{y}_1(t, \mathbf{x})$ of ODE (8) is converging to $G(\mathbf{g}(\mathbf{x}), 0, \mathbf{x})$ for any initial condition. Furthermore, for any initial condition $\mathbf{y}_1(0, \mathbf{x}) \neq G(\mathbf{g}(\mathbf{x}), 0, \mathbf{x})$, we have*

$$\left\| \mathbf{y}_1\left(\frac{1}{2}, \mathbf{x}\right) - \mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) \right\| \leq \frac{\sqrt{2}}{2\sqrt{\int_0^{\frac{1}{2}} C(z) dz}}.$$

For any $m \in \mathbb{N}$, we can select constant c such that for any initial condition $\mathbf{y}_1(0, \mathbf{x})$,

$$\left\| \mathbf{y}_1\left(\frac{1}{2}, \mathbf{x}\right) - \mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) \right\| \leq 2^{-m}$$

Consequently, $\mathbf{y}_1(t, \mathbf{x})$ will approach $\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) = \mathbf{f}(1, \mathbf{x})$ on this interval. Thus, $\mathbf{y}_1(\frac{1}{2}, \mathbf{x}) =_\epsilon \mathbf{f}(1, \mathbf{x})$ and $\mathbf{y}_2(\frac{1}{2}, \mathbf{x}) = \mathbf{g}(\mathbf{x})$, for some $\epsilon > 0$, that we can consider less than $\frac{1}{4} = 2^{-2}$, by selecting a big enough constant c (just taking $m = 2$ above). At $t = \frac{1}{2}$, \mathbf{y}_1 will hence have simulated one step of the discrete ODE.

Now, for $t \in [\frac{1}{2}, 1]$ the roles of \mathbf{y}_1 and \mathbf{y}_2 are exchanged : $\mathbf{y}'_1(t, \mathbf{x}) = 0$, so \mathbf{y}_1 is kept fixed, \mathbf{y}_2 approaches $r(\mathbf{y}_1) = \mathbf{f}(1, \mathbf{x})$, thus $\mathbf{y}_1(1, \mathbf{x}) =_\epsilon \mathbf{y}_2(1, \mathbf{x}) =_\epsilon \mathbf{f}(1, \mathbf{x})$.

By induction, from the same reasoning, we obtain that, for all $n \in \mathbb{N}$, $\mathbf{y}_1(n, \mathbf{x}) =_\epsilon \mathbf{y}_2(n, \mathbf{x}) =_\epsilon \mathbf{f}(n, \mathbf{x})$, and actually, we also have, for all $t \in [n, n + \frac{1}{2}]$, for any integer n :

$$\mathbf{y}_1\left(t + \frac{1}{2}, \mathbf{x}\right) =_\epsilon \mathbf{y}_2(t, \mathbf{x}) =_\epsilon \mathbf{f}(n, \mathbf{x}).$$

I Proof of Lemma 4 (Main result II)

As in Proposition 5, to guarantee an error of $\epsilon = 2^{-n}$, it is sufficient to take $\epsilon' = 2^{-p(n)}$ and $\theta_{\epsilon'}(x) = \text{ReLU}\text{-}\mathfrak{s}(2^{p(n)}, x)$ for some polynomial p . Note that Equation (2) is a robust continuous ODE (see Definition 10). To prove this lemma, we first state the following one:

Lemma 10 (Improved error analysis of ODE (8), [23, Lemma 4.5] [12, 5.2]). *Consider a point $b \in \mathbb{R}$, some $\gamma > 0$ some reals $t_0 < t_1$, and a function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ 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 $\bar{b}, E : \mathbb{R} \rightarrow \mathbb{R}$ be functions such that that $|\bar{b}(t) - b| \leq \rho$ and $|E(t)| \leq \delta$ for all $t \geq t_0$. Then the IVP defined by*

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

with the initial condition $z(t_0) = \bar{z}_0$, where $\gamma > 0$ and $c \geq \frac{1}{2\gamma^2 \int_{t_0}^{t_1} \phi(t) dt}$ satisfies

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. The proof is similar to the proof of Proposition 5, except we have to take into account the additional error of the computation of Equation (2). The key here is also to make sure the errors propagate additively (and not multiplicatively): rounding function corrects errors of order ϵ whenever its argument is at a distance less than $\frac{1}{4}\delta$ of some $n\delta$. We introduce some error ϵ' at every step; but the latter is corrected at the next step). The involved constant c is of order 2^n . We use the notations of Definition 12.

We prove, by induction, the following property: for all $n \in \mathbb{N}$, $\mathbf{y}_1(n, \mathbf{x}) =_\epsilon \mathbf{y}_2(n, \mathbf{x}) =_\epsilon \mathbf{f}(n, \mathbf{x})$, and $\mathbf{y}_1(t + \frac{1}{2}, \mathbf{x}) =_\epsilon \mathbf{y}_2(t, \mathbf{x}) =_\epsilon \mathbf{f}(n, \mathbf{x})$ for all $t \in [n, n + \frac{1}{2}]$.

Initialisation: For $n = 0$, $\mathbf{f}(0, \mathbf{x}) = \mathbf{y}_1(0, \mathbf{x}) = \mathbf{y}_2(0, \mathbf{x}) = \mathbf{g}(\mathbf{x})$.

- For $t \in [0, 1/2]$, $r(\mathbf{y}_2)$ is kept close to a constant value $\mathbf{g}(\mathbf{x})$, when an error less than ϵ' , if we choose $\epsilon' < \frac{1}{4}\delta$. Meanwhile, $r(t)$ is also at a value close to n with an error lesser than ϵ' . We have $\theta_{\epsilon'}(-\mathbf{y}(t)) =_{\epsilon'} 0$, thus $\mathbf{y}_2(t, \mathbf{x}) =_{\epsilon'} 0$, then $y_2(t, \mathbf{x})$ remains close to $y_2(0, \mathbf{x})$ on this interval. Here the hypothesis that \mathbf{G} is almost constant around $\mathbb{N}\delta$ means that its value is guaranteed to be at ϵ' from $\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x})$ on the $[0, \frac{1}{2}]$.

By Lemma 10, $\mathbf{y}_1(t, \mathbf{x})$ approach $\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) = \mathbf{f}(1, \mathbf{x})$ on this interval, with an error of order $\epsilon' + \epsilon' + \epsilon' + \frac{1}{2}\epsilon'$. Thus, $\mathbf{y}_1(\frac{1}{2}, \mathbf{x}) =_{\epsilon/2} \mathbf{f}(1, \mathbf{x})$, if we choose $\frac{7}{2}\epsilon' < \epsilon/2$.

- At $t = \frac{1}{2}$, \mathbf{y}_1 will hence have simulated one step of the associated discrete ODE, with an error less than $\epsilon/2$, and \mathbf{y}_2 will be close to $\mathbf{g}(\mathbf{x})$ with an error less than $\epsilon' < \epsilon/2$.
- For $t \in [\frac{1}{2}, 1]$ the roles of y_1 and y_2 are inverted, and the analysis is analogous.

Induction:

- For $t \in [n, n + 1/2]$, we have $\theta(-\mathbf{y}(t)) =_{\epsilon'} 0$, and hence $\mathbf{y}'_2(t) =_{\epsilon'} 0$, so \mathbf{y}_2 is kept almost fixed (with an error less than $\frac{1}{2}\epsilon'$). In the same time \mathbf{y}_1 approaches $G(r(\mathbf{y}_2(t)), r(t), \mathbf{x}) = \mathbf{f}(n + 1, \mathbf{x})$ by Lemma 10, with some new error of order less than $\frac{7}{2}\epsilon' < \epsilon/2$.
- For $t \in [n + \frac{1}{2}, n + 1]$ the roles of \mathbf{y}_1 and \mathbf{y}_2 are exchanged: $\mathbf{y}'_1(t, \mathbf{x}) =_{\epsilon'} 0$, so \mathbf{y}_1 is kept almost fixed, with a new error less than $\frac{1}{2}\epsilon'$. In the same time \mathbf{y}_2 approaches $r(\mathbf{y}_1(t)) = \mathbf{f}(n + 1, \mathbf{x})$ by Lemma 10, with some new error of order less than $\frac{7}{2}\epsilon' < \epsilon/2$.

Consequently, we get the property at rank $n + 1$. □

J Proof of Theorem 9 (Main Theorem III)

Proof of Theorem 9. We reduce from **DRRL**:

- $\mathfrak{t} : \mathcal{H} = (X, g) \rightarrow \mathcal{H}_C = (X_C, g_C)$ doing the adaptation of the Branicky trick described by Proposition 5: with $X = X_C$ and g_C such that:

$$\left\{ \begin{array}{l} g_C(0, \mathbf{x}) = g_C^m(0, \mathbf{x}) = g(0, \mathbf{x}) = \gamma_{config}(C_0); \mathbf{y}(0) = 0; \mathbf{z}(0) = 1 \\ \frac{\partial g_C}{\partial t}(t, \mathbf{x}) = c \cdot \left(G(r_{\epsilon'}(g_C^m(t, \mathbf{x})), r_{\epsilon'}(t), \mathbf{x}) - g_C(t, \mathbf{x}) \right)^3 \cdot \theta_{\epsilon'}(\mathbf{y}(t)) \\ \frac{\partial g_C^m}{\partial t}(t, \mathbf{x}) = c \cdot \left(r_{\epsilon'}(g_C(t, \mathbf{x})) - g_C^m(t, \mathbf{x}) \right)^3 \cdot \theta_{\epsilon'}(-\mathbf{y}(t)) \\ \frac{\partial \mathbf{y}}{\partial t}(t) = 2\pi \cdot \mathbf{z}(t) \\ \frac{\partial \mathbf{z}}{\partial t}(t) = -2\pi \cdot \mathbf{y}(t) \end{array} \right.$$

where c is a constant; $G : (v, t, \mathbf{x}) \rightarrow \mathbf{u}(v, t, \mathbf{x}) + v$ and \mathbf{u} such that $\frac{\partial g}{\partial t}(t, \mathbf{x}) = \mathbf{u}(g(t, \mathbf{x}), t, \mathbf{x})$; $r_{\epsilon'}$ and $\theta_{\epsilon'}$ are defined in Proposition 5. The function g_C^m has the value of a memory equation, as y_2 in the Definition 12.

It is computable in polynomial time by Lemma 5.

- $\mathfrak{s} : \gamma(w) \rightarrow \gamma(w')$ where w' is the successor of w in the discrete ODE.

The equivalence is straightforward from Lemmas 4 and 10. □