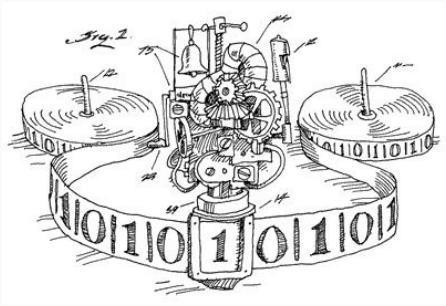
Discrete-Time and Continuous-Time Systems over the Reals: Relating Complexity with Robustness, Length and Precision

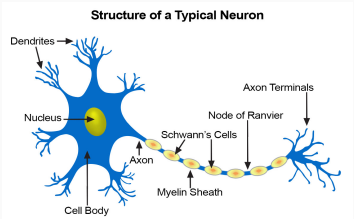
PhD Defence **Manon Blanc**

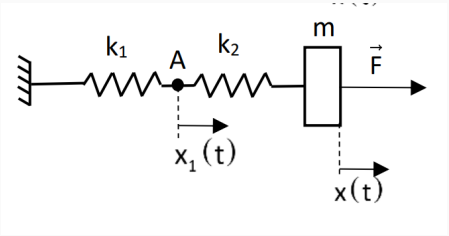
Jury: **Enrico Formenti**, **Juha Kontinen**, **Elvira Mayordomo**, **Cristóbal Rojas**, **Pierre Valarcher**

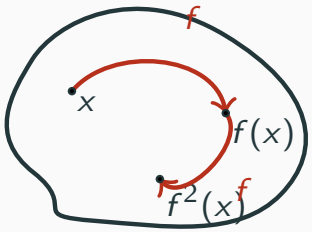
Introduction

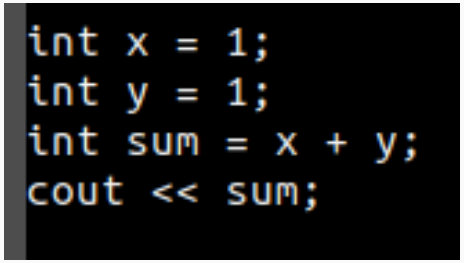
Introduction: what are we studying?

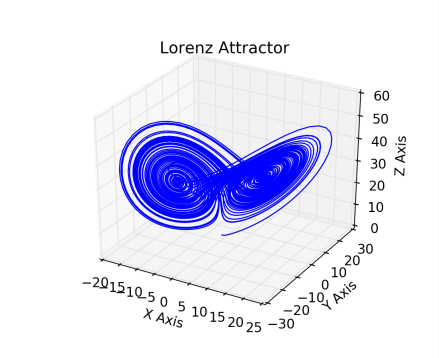


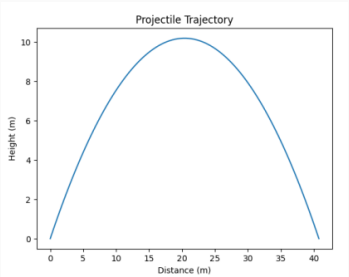






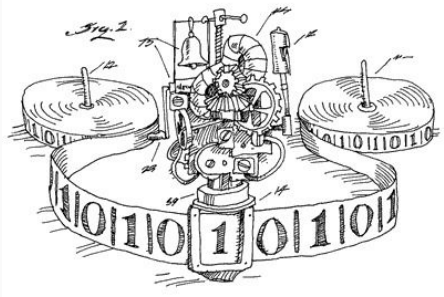


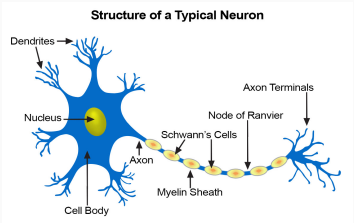


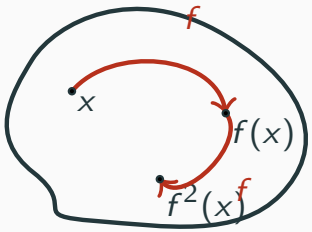


Introduction: what are we studying?

Discrete-time

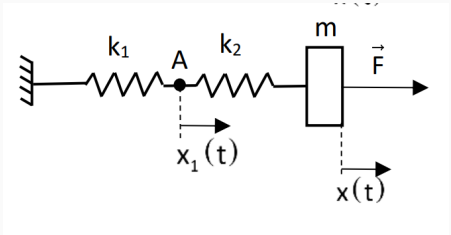


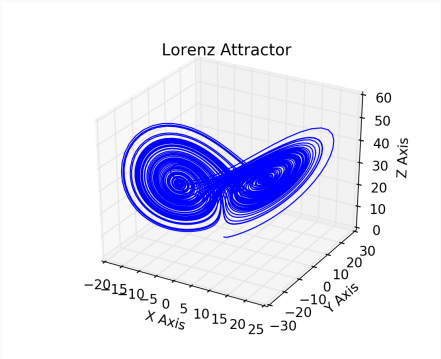


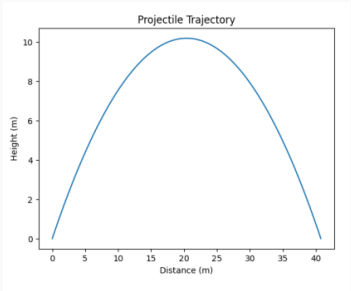



```
int x = 1;
int y = 1;
int sum = x + y;
cout << sum;
```

Continuous-time







Introduction

- Continuous-Time Systems: can be described by a (continuous) ODE

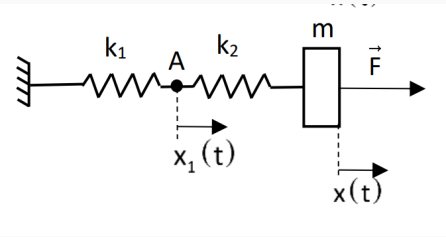
$$\frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x})$$

- Discrete-Time Systems: can be described by a discrete ODE

$$\frac{\delta \mathbf{f}(n, \mathbf{y})}{\delta n} = \mathbf{f}(n+1, \mathbf{y}) - \mathbf{f}(n, \mathbf{y}) = \mathbf{u}(\mathbf{f}(n, \mathbf{y}), n, \mathbf{y})$$

Introduction

- Continuous-Time Systems: can be described by a (continuous) ODE



$$\frac{\partial^2 x}{\partial^2 t}(t) + \frac{k}{m}x(t) = 0$$

- Discrete-Time Systems: can be described by a discrete ODE

$$f : n \to 2^n$$

$$\frac{\delta f(n, \mathbf{y})}{\delta n} = f^2(n) - f(n)$$

Content

- Characterisations of polynomial-time and polynomial-space;
- Complexity of *robust* dynamical systems;
- Robustness of tilings.

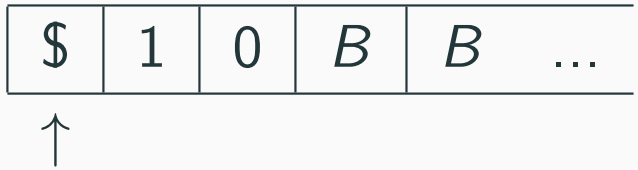
Motivations

Motivations

- Verification: can a system reach an unsafe state from an initial state (reachability)
- Models of computation: what is the computational power of a given model
- Ressources: how to measure time, memory (space) used by a computation

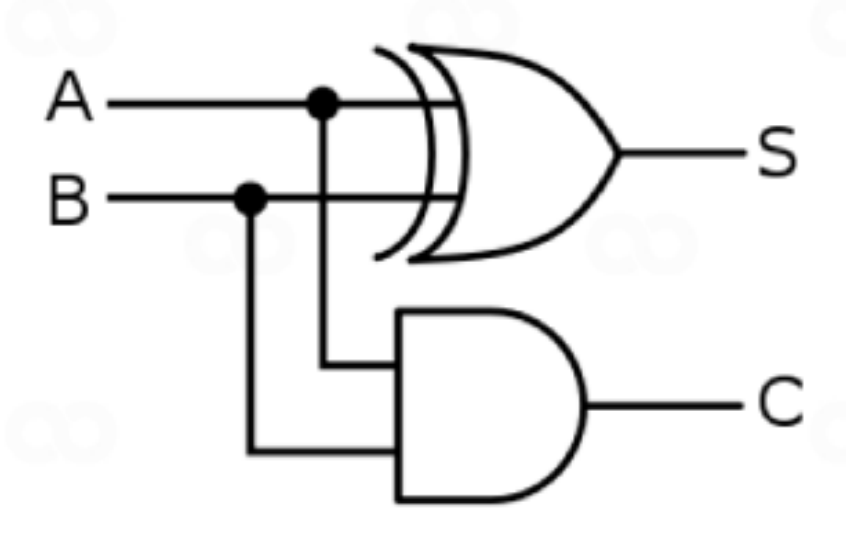
Discrete models of computation: Turing machines

The model:

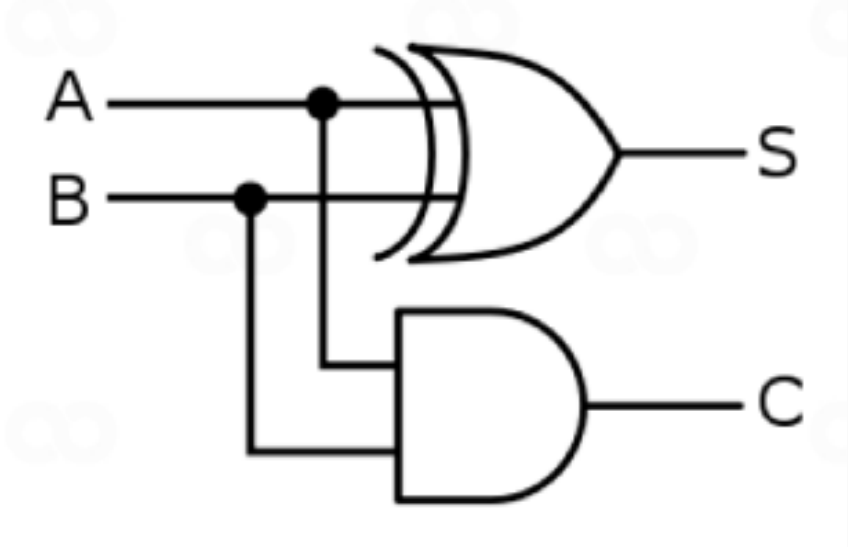


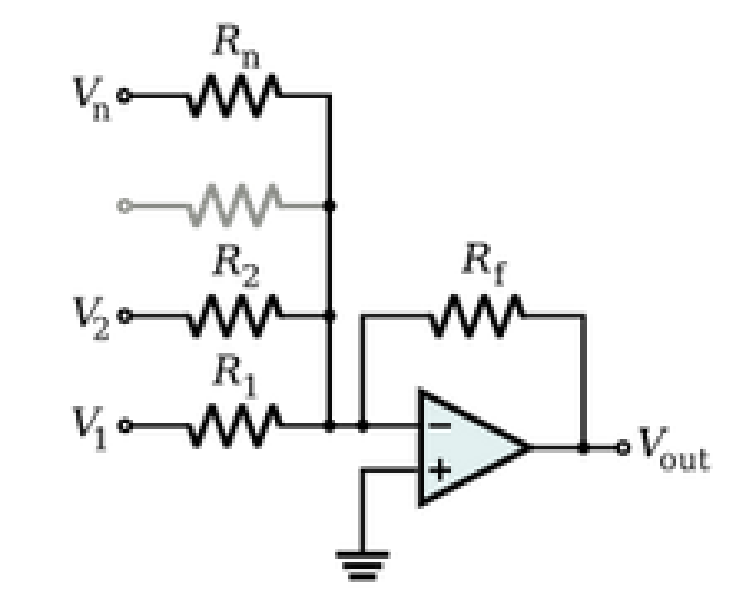
Physical World	Mathematical Model
Computer	Turing machines, boolean circuits...

Example: the addition

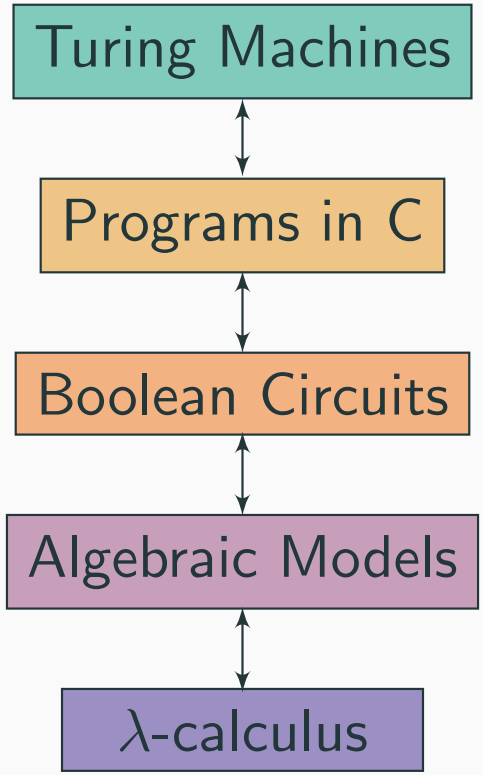


Example: the addition





Computationnal Power: model equivalence



Why algebraic characterisation?

$$[f_1, f_2, \cdots, f_k; o_1, o_2, \cdots, o_n]$$

- Allows a more denotational point of view on complexity classes

Why algebraic characterisation?

$$[f_1, f_2, \cdots, f_k; o_1, o_2, \cdots, o_n]$$

- Allows a more denotational point of view on complexity classes
- This is already done in computability:

$$[0, \mathbf{s}, \pi; \textit{composition}, \textit{minimisation}, \textit{primitive recursion}]$$

On our example

Effective Church-Turing thesis: Every effectively and *reasonable* models of computation yields to the same complexity classes

$\rightarrow$ Notions of time and space do not depend on the model, up to some polynomial

$\rightarrow$ Verification is hard

On our example

Effective Church-Turing thesis: Every effectively and *reasonable* models of computation yields to the same complexity classes

$\rightarrow$ Notions of time and space do not depend on the model, up to some polynomial

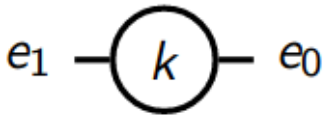
$\rightarrow$ Verification is hard

What about continuous-time models?

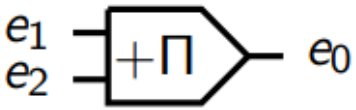
The General Purpose Analog Computer (GPAC)

- Introduced by Shannon (1941);
- Mathematical model of the *Differential Analizer*;
- Model based on circuits, with several interconnected units doing basic operations
- Corresponds to polynomial ODEs: $y' = p(y)$

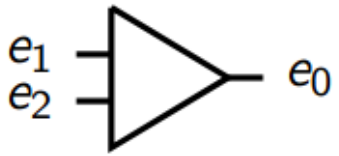
GPAC



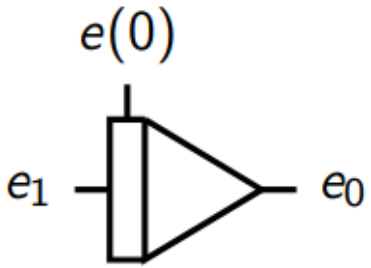
constant: $e_0 = ke_1$



product: $e_0 = e_1 e_2$



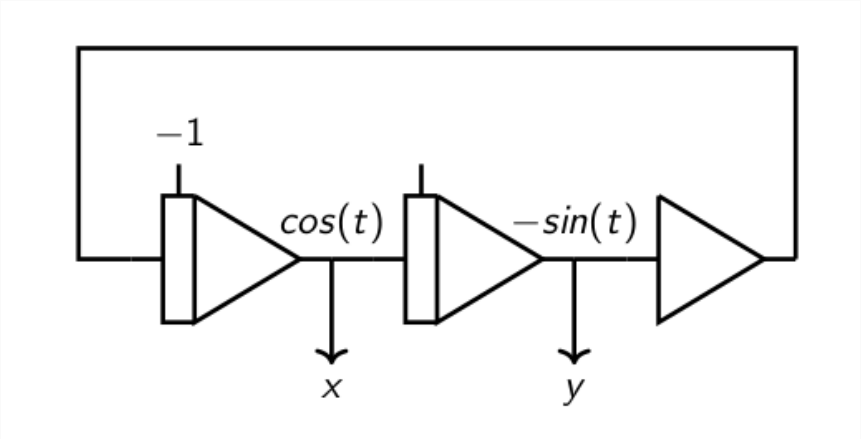
summer: $e_0 = -(e_1 + e_2)$



integrator:

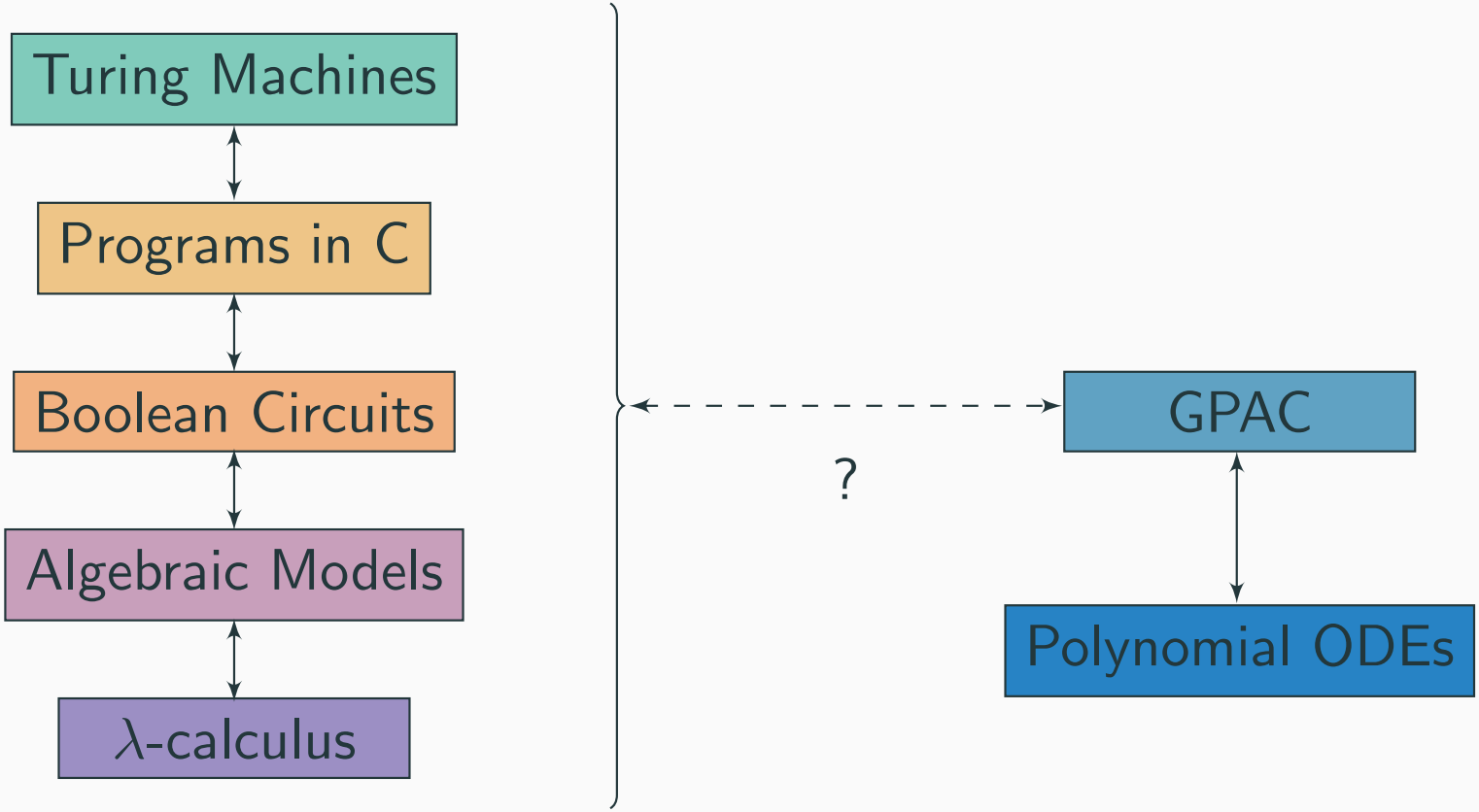
$$e_0 = -\int_0^t (e_1(u)du + e(0))$$

GPAC: how does it work?



$$\begin{cases} x'(t) &= y(t) \\ y'(t) &= -x(t) \\ x(0) &= -1 \\ y(0) &= 0 \end{cases} \Rightarrow \begin{cases} x(t) &= \cos(t) \\ y(t) &= -\sin(t) \end{cases}$$

Computability: GPAC



Is verification simpler for continuous-time systems?

- Difficulty: simulating Turing machines with continuous settings (e.g. Graça, Rojas) with
  - generalised shifts (Moore)
  - recurrent neural networks (Siegelmann, Sontag)
  - analytic functions (Graca, Buescu, Campagnolo)
  - hydrostatic equations in Beltrami fields (Cardona, Miranda, Peralta-Salas)
  - 2d continuous PAMs (Koiran, Cosnard, Garzon)
  - ...

Solving continuous ODEs

$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x}),$$

- Non-computable in the general case, for $\mathbf{u}$ computable (Pour-El, Richards)

Solving continuous ODEs

$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x}),$$

- Non-computable in the general case, for $\mathbf{u}$ computable (Pour-El, Richards)
- Computable for $\mathbf{u}$ computable and unique solution (Collins, Graça, Ruohonen)

Solving continuous ODEs

$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x}),$$

- Non-computable in the general case, for $\mathbf{u}$ computable (Pour-El, Richards)
- Computable for $\mathbf{u}$ computable and unique solution (Collins, Graça, Ruohonen)
- FPSPACE-completeness on a bounded domain for $\mathbf{u}$ computable in polynomial-time (Kawamura, Ko)

Complexity

But now, how do we define the complexity?

- Measuring the complexity so it corresponds to the digital world

Complexity

- Time?:

  $\rightarrow$ Time = Length

  - Continuous ODEs : Bournez, Graça, Pouly
  - Discrete ODEs : Bournez, Durand, Kontinen, Antonelli, ...

Complexity

- Time?:

  $\rightarrow$ Time = Length

  - Continuous ODEs : Bournez, Graça, Pouly
  - Discrete ODEs : Bournez, Durand, Kontinen, Antonelli, ...

- Space?:

  - Effective Church thesis for probabilistic systems: Rojas
  - A characterisation of PSPACE: Gozzi, Graça

  $\rightarrow$ **Our approach: Space = Precision**

Our frameworks

- Mathematical framework: Computable Analysis
- Complexity framework: Algebraic Models

Definitions (from computable analysis): intuition

- What is a computable real number x?

  $\rightarrow$ we can compute a representation of x.

Definitions (from computable analysis): intuition

- What is a computable real number x?

  $\rightarrow$ we can compute a representation of x.

Example

e, π are computable. $\sum_{i \geq 1} 2^{-BB(i)}$, where BB is the Busy Beavers function is not.

Definitions (from computable analysis): intuition

- What is a computable real number x?

  $\rightarrow$ we can compute a representation of x.

Example

e, π are computable. $\sum_{i \geq 1} 2^{-BB(i)}$, where BB is the Busy Beavers function is not.

- What is a computable function $f : \mathbb{R} \rightarrow \mathbb{R}$?

  $\rightarrow$ on a representation of $x \in \mathbb{R}$, we produce a representation of $f(x)$.

Definitions (from computable analysis): intuition

- What is a computable real number x?

  $\rightarrow$ we can compute a representation of x.

Example

e, π are computable. $\sum_{i \geq 1} 2^{-BB(i)}$, where BB is the Busy Beavers function is not.

- What is a computable function $f : \mathbb{R} \rightarrow \mathbb{R}$?

  $\rightarrow$ on a representation of $x \in \mathbb{R}$, we produce a representation of $f(x)$.
- How do we define complexity over the reals?

Algebraic Models: for complexity classes

- Characterisation of polynomial time with explicit bounds: e.g. Cobham

Algebraic Models: for complexity classes

- Characterisation of polynomial time with explicit bounds: e.g. Cobham

Here: describing complexity without explicit bounds (e.g. Bellantoni, Cook, Levant, Marion, Jacobé de Naurois, ...):

Algebraic Models: for complexity classes

- Characterisation of polynomial time with explicit bounds: e.g. Cobham

Here: describing complexity without explicit bounds (e.g. Bellantoni, Cook, Levant, Marion, Jacobé de Naurois, ...):

- With BSS model (Jacobé de Naurois, Marion, ...)
- Characterisation of FPTIME with linear length ODEs (Bournez, Durand, Kontinen, ...)

Time and Space as algebras

Algebraic characterisations: Chapters 3 and 5 in one slide

	With Discrete ODEs	With ODEs
FPTIME for $\mathbb{N}^{\mathbb{N}}$	(Bournez, Durand)	
FPTIME for $\mathbb{R}^{\mathbb{N}}$	$\overline{\mathbb{LDL}^{\bullet}}$ (thm. 3.1.6)	
FPTIME for $\mathbb{R}^{\mathbb{R}}$	$\overline{\mathbb{LDL}^{\circ}}$ (thm. 3.2.3)	(Bournez, Graça, Pouly)
FPSPACE for $\mathbb{R}^{\mathbb{R}}$	$\overline{\mathbb{RLD}^{\circ}}$ (thm. 5.1.6)	$\overline{\mathbb{RCD}}$ (thm. 5.2.2)

Algebraic characterisations: Chapters 3 and 5 in one slide

	With Discrete ODEs	With ODEs
FPTIME for $\mathbb{N}^{\mathbb{N}}$	(Bournez, Durand)	
FPTIME for $\mathbb{R}^{\mathbb{N}}$	$\overline{\mathbb{LDL}^{\bullet}}$ (thm. 3.1.6)	
FPTIME for $\mathbb{R}^{\mathbb{R}}$	$\overline{\mathbb{LDL}^{\circ}}$ (thm. 3.2.3)	(Bournez, Graça, Pouly)
FPSPACE for $\mathbb{R}^{\mathbb{R}}$	$\overline{\mathbb{RLD}^{\circ}}$ (thm. 5.1.6)	$\overline{\mathbb{RCD}}$ (thm. 5.2.2)

- $\overline{\mathbb{LDL}^{\bullet}}$, $\overline{\mathbb{LDL}^{\circ}}$ based on Linear Length ODEs;
- $\overline{\mathbb{RLD}^{\circ}}$ based on Robust Linear Discrete ODEs;
- $\overline{\mathbb{RCD}}$ based on Robust Continuous ODEs.

Robust Continuous ODEs

Definition (5.2.3)
$\mathbf{f} : \mathbb{R} \to \mathbb{R}$ is robustly ODE definable (from initial condition $\mathbf{g}$, and dynamic $\mathbf{u}$) if

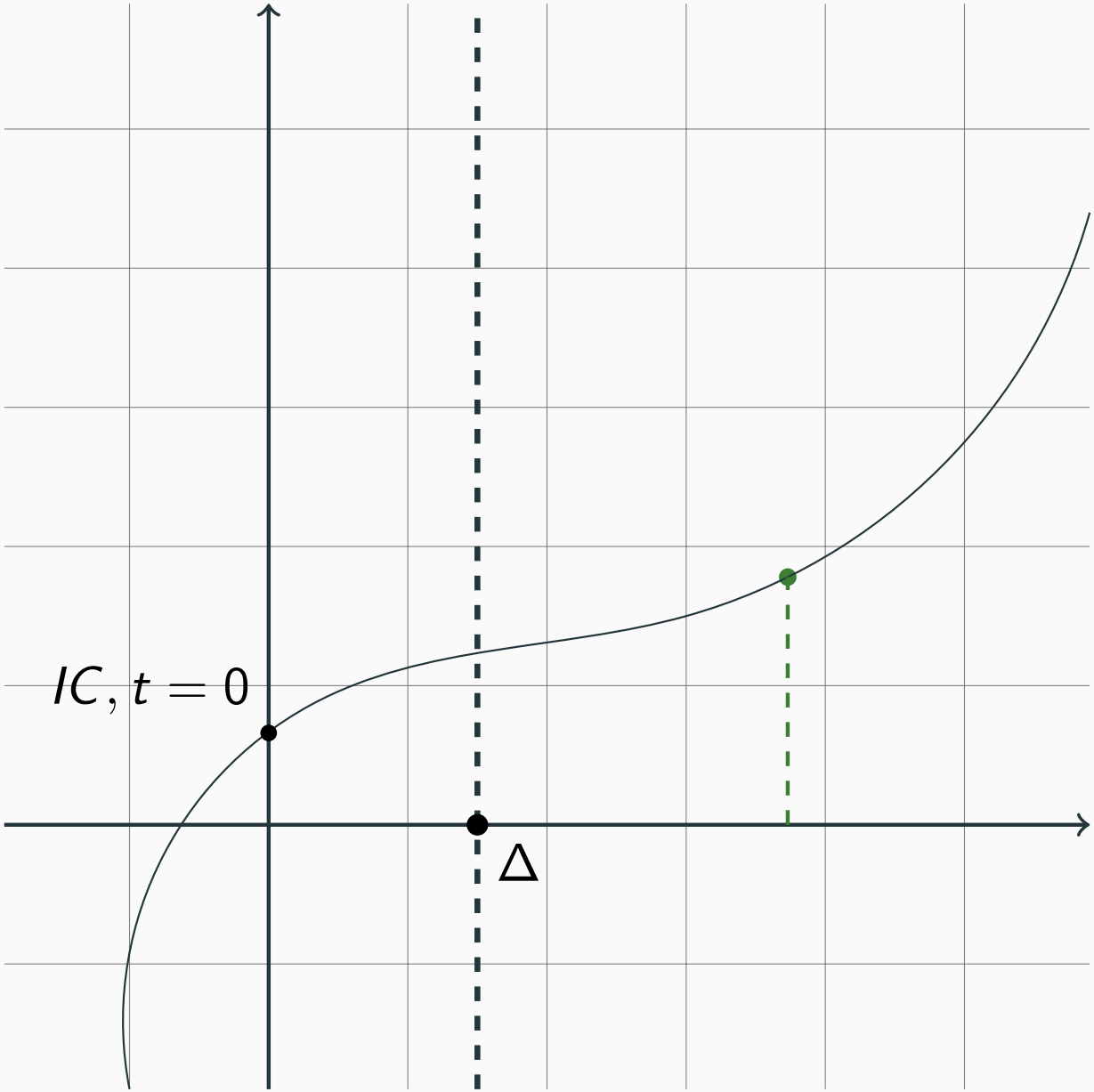
- it is the solution of the following continuous ODE:

$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x});$$

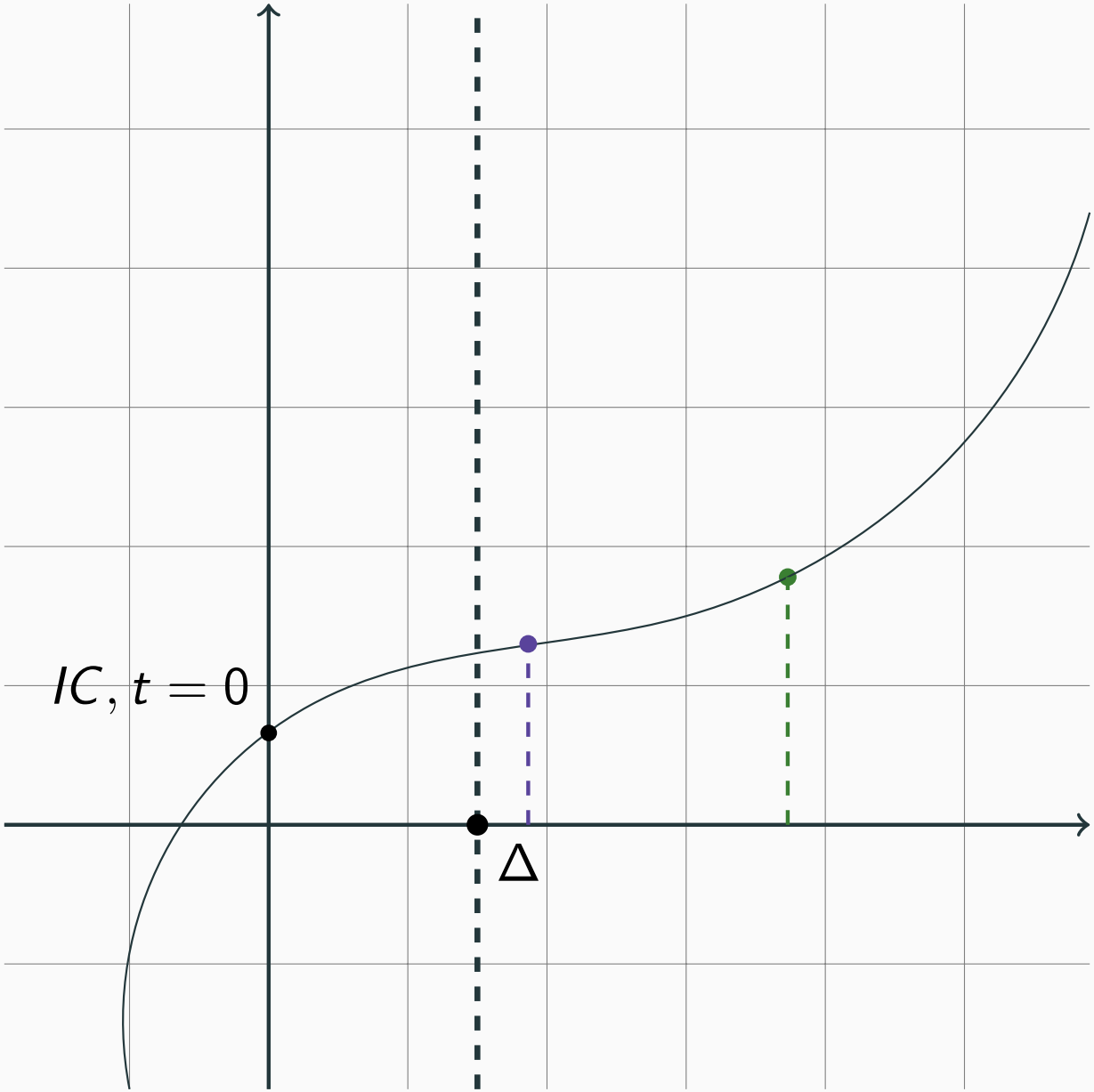
- And polynomially numerically stable (to be defined : next slides)

$\to$ this allows computation by dichotomy.

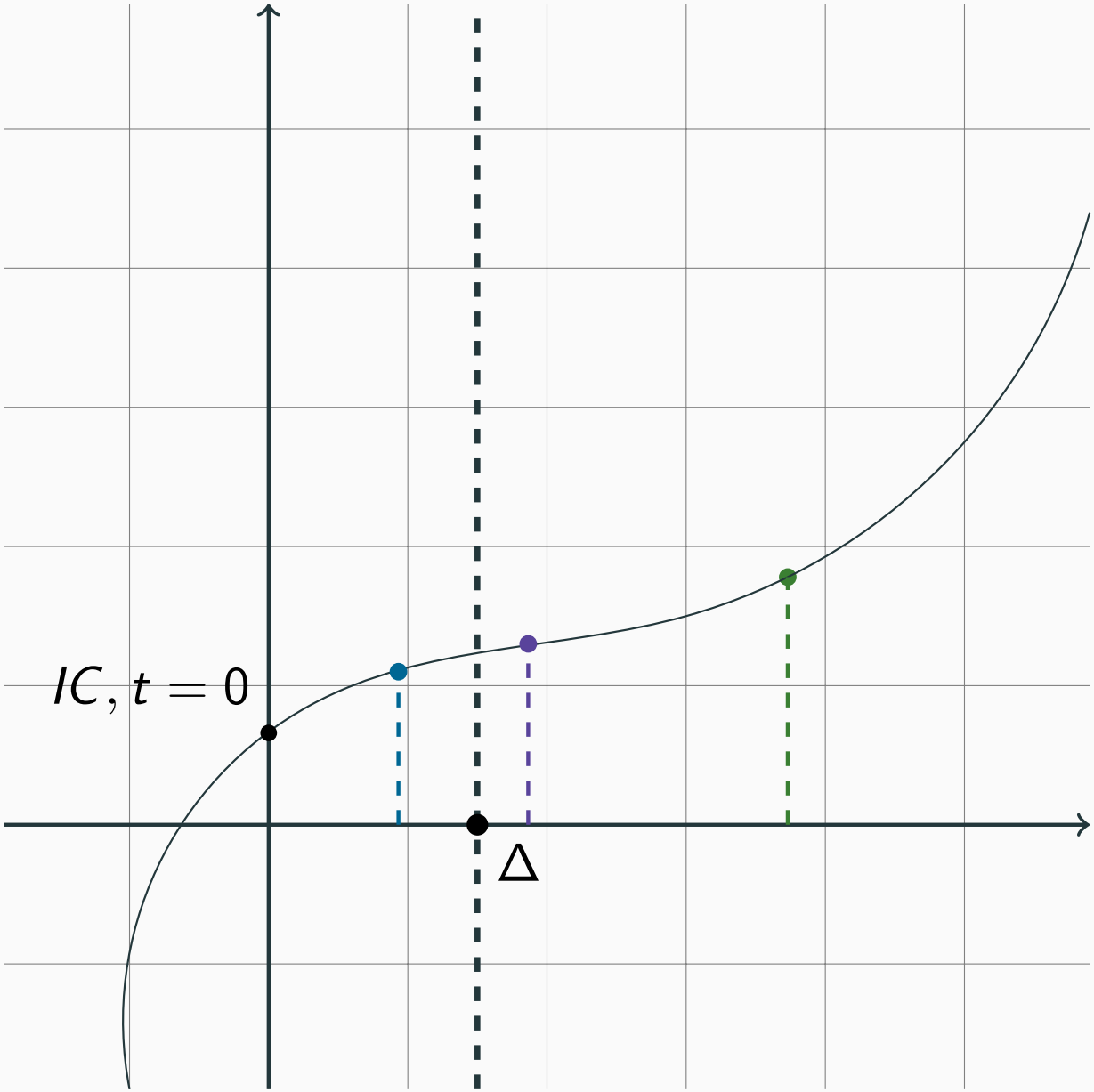
An unusual way of solving an ODE



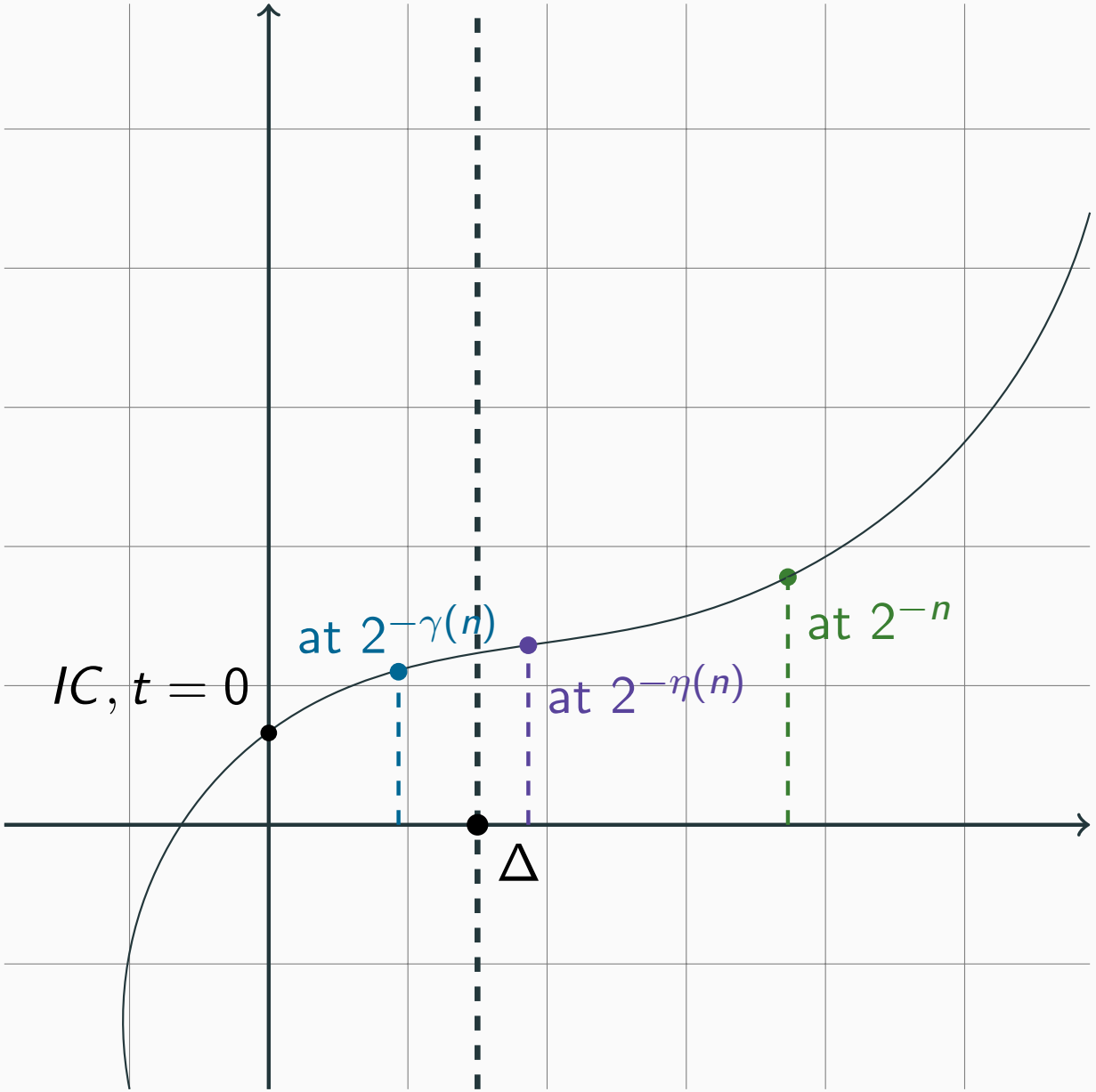
An unusual way of solving an ODE



An unusual way of solving an ODE



An unusual way of solving an ODE



Robust Continuous ODEs (2/2)

More formally:

- $\exists \Delta \in \mathbb{Q}_+^*$, such that the previous ODE is (polynomially space) solvable on $[0, \Delta]$.

- For $t \geq \Delta$, we can compute $\mathbf{f}(t, \mathbf{x})$ at 2^{-n} by computing some approximation $\widetilde{\mathbf{f}(t/2, \mathbf{x})}$ of $\mathbf{f}(t/2, \mathbf{x})$ at precision $2^{-\eta(n)}$.

Characterisation of PSPACE with continuous ODEs

We consider :

$$\overline{\mathbb{RCD}} = [\mathbf{0}, \mathbf{1}, \pi_i^k, +, -, \times, \tanh, \cos, \pi, \frac{x}{2}, \frac{x}{3};$$

$$\textit{composition}, \textit{robust continuous ODE}, \textit{ELim}]$$

Theorem (5.2.2)

$$\overline{\mathbb{RCD}} \cap \mathbb{R}^{\mathbb{R}} = \mathsf{FPSPACE} \cap \mathbb{R}^{\mathbb{R}}$$

Proof ideas ($\subseteq$)

$$\overline{\mathbb{RLD}^{\circ}} \cap \mathbb{R}^{\mathbb{R}} = \overline{\mathbb{RCD}} \cap \mathbb{R}^{\mathbb{R}}$$

- **Continuous $\Rightarrow$ Discrete settings:**
  From our definitions of robustness of ODEs

Proof ideas ($\subseteq$)

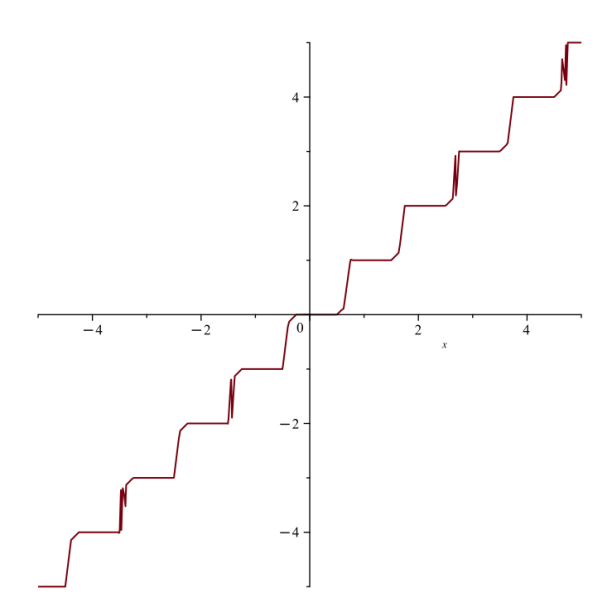
All the functions and operators in $\overline{\mathbb{RCD}}$ are computable in polynomial-space.

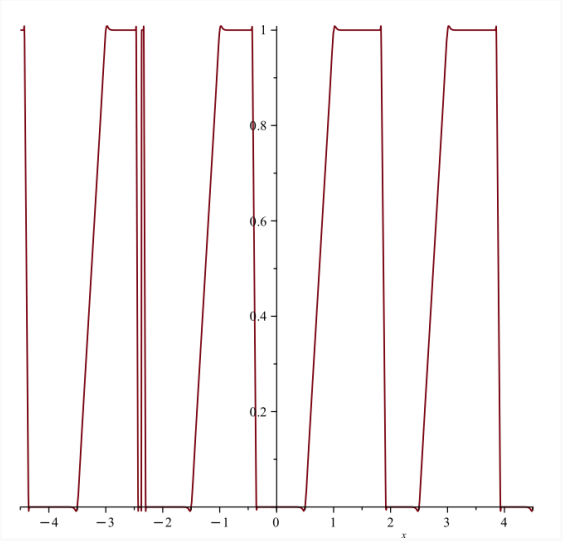
Proof ideas ($\supseteq$): with discrete ODEs

- We give *continuous* approximations of some basic functions (integer/fractional part, modulo 2, ...):

Proof ideas ($\supseteq$): with discrete ODEs

- We give *continuous* approximations of some basic functions (integer/fractional part, modulo 2, ...):





Proof ideas ($\supseteq$): with discrete ODEs

- We give *continuous* approximations of some basic functions (integer/fractional part, modulo 2, ...);

- We encode each step of the transition of the Turing machine into the algebra, using the previous functions : Next;

Proof ideas ($\supseteq$): with discrete ODEs

- We give *continuous* approximations of some basic functions (integer/fractional part, modulo 2, ...);
- We encode each step of the transition of the Turing machine into the algebra, using the previous functions : Next;
- We obtain a *robust linear discrete ODE* encoding the execution of the Turing machine:
  $State(t+1) = Next(State(t))$ so

$$\frac{\delta \overline{State}(t,y)}{\delta t} = \overline{Next}(\overline{State}(t,y))$$

.

Proof ideas ($\supseteq$): going to continuous world

$$\overline{\mathbb{R}\mathrm{LD}^{\circ}} \cap \mathbb{R}^{\mathbb{R}} = \overline{\mathbb{R}\mathrm{CD}} \cap \mathbb{R}^{\mathbb{R}}$$

Proof ideas ($\supseteq$): going to continuous world

$$\overline{\mathbb{RLD}^{\circ}} \cap \mathbb{R}^{\mathbb{R}} = \overline{\mathbb{RCD}} \cap \mathbb{R}^{\mathbb{R}}$$

- **Discrete $\Rightarrow$ Continuous settings:**
  Adaptation of the a trick due to Branicky ('95)

Proof ideas ($\supseteq$): going to continuous world

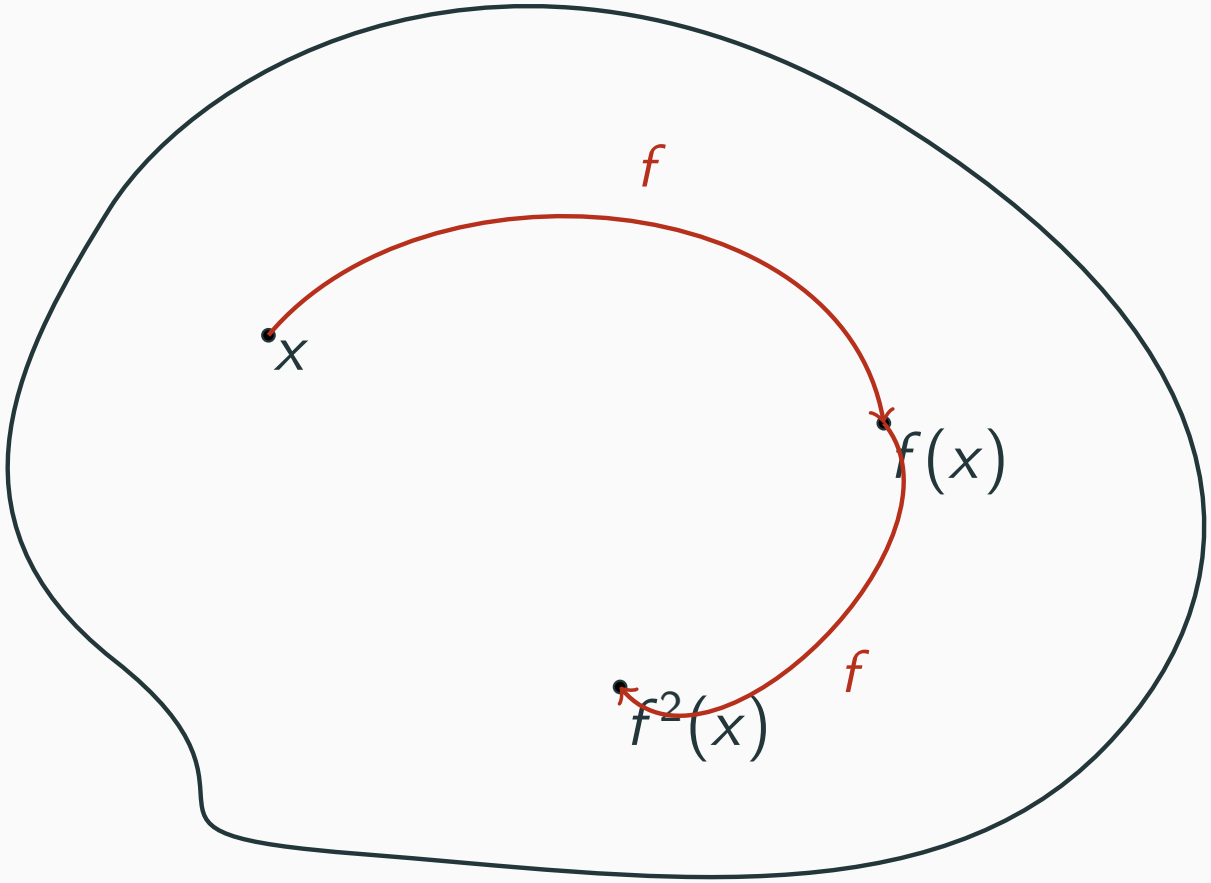
$$\overline{\mathbb{RLD}^{\circ}} \cap \mathbb{R}^{\mathbb{R}} = \overline{\mathbb{RCD}} \cap \mathbb{R}^{\mathbb{R}}$$

- **Discrete $\Rightarrow$ Continuous settings:**
  Adaptation of the a trick due to Branicky ('95)

$$\frac{\delta f}{\delta t}(t) = u(y(t), t) \qquad \begin{cases} z_1(0) = x_0 \\ z_2(0) = x_0 \\ \begin{cases} \frac{\partial z_2}{\partial t}(t) = c_2 \left(r(z_1(t)) - z_2(t)\right)^3 \theta\left(-\cos 2\pi t\right) \\ \frac{\partial z_1}{\partial t}(t) = c_1 \left(\bar{u}\left(r(z_2(t))\right) - z_1(t)\right)^3 \theta\left(\cos 2\pi t\right) \end{cases} \end{cases}$$

Robustness in Dynamical Systems

Dynamical Systems



Computability Properties

- Reachability is undecidable...

Computability Properties

- Reachability is undecidable...
- ... more precisely, it is not co-computably enumerable;

Computability Properties

- Reachability is undecidable...
- ... more precisely, it is not co-computably enumerable;
- It becomes decidable for a sub-class of systems: Robust Dynamical Systems.

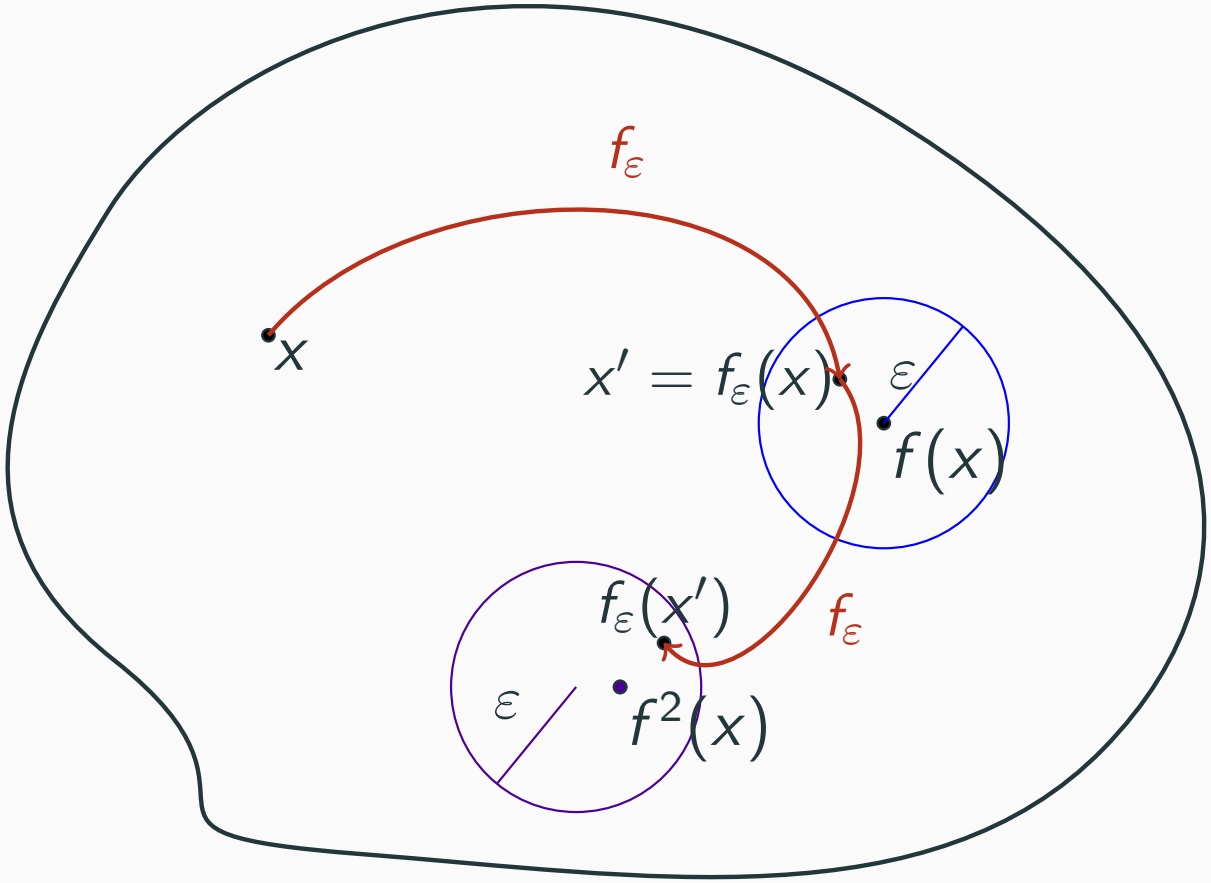
Computability

- Informal conjecture: undecidability in verification does not happen for robust systems.

- Undecidabilty is related to non-robustness of the systems (Asarin, Bouajjani)

  $\rightarrow$ with proper notion of robusteness: no sensitivity to an arbitrarly small perturbation.

Robustness in Dynamical Systems



Complexity for Reach

- go from computability to complexity,
- quantifying the robustness to characterise PSPACE,
- having a notion of robustness to characterise PTIME.

Robustness

$$\text{Consider } R_\omega^{\mathcal{H}} = \bigcap_{\epsilon>0} R_\epsilon^{\mathcal{H}}(x, y)$$

- Say $R^{\mathcal{H}}$ is robust when $R_\omega^{\mathcal{H}} = R^{\mathcal{H}}$:

$$R^{\mathcal{H}} \text{ robust} \Rightarrow R^{\mathcal{H}} \text{ computable}.$$

- Reachability in robust dynamical systems is *decidable* (Asarin & Bouajjani)

PSPACE **on robust dynamical systems**

Theorem (4.2.33)

Take a locally Lipschitz system, with $f : X \to X$ *polynomial-time computable, with* X *a closed rational box. Then, for* $p : \mathbb{N} \to \mathbb{N}$ *a polynomial,* $R_p^{\mathcal{H}} \subseteq \mathbb{Q}^d \times \mathbb{Q}^d \times \mathbb{N} \in$ PSPACE.

$\to$ works also for dynamical systems over the reals

Example
Simulation of Turing machines with continuous PAMs.

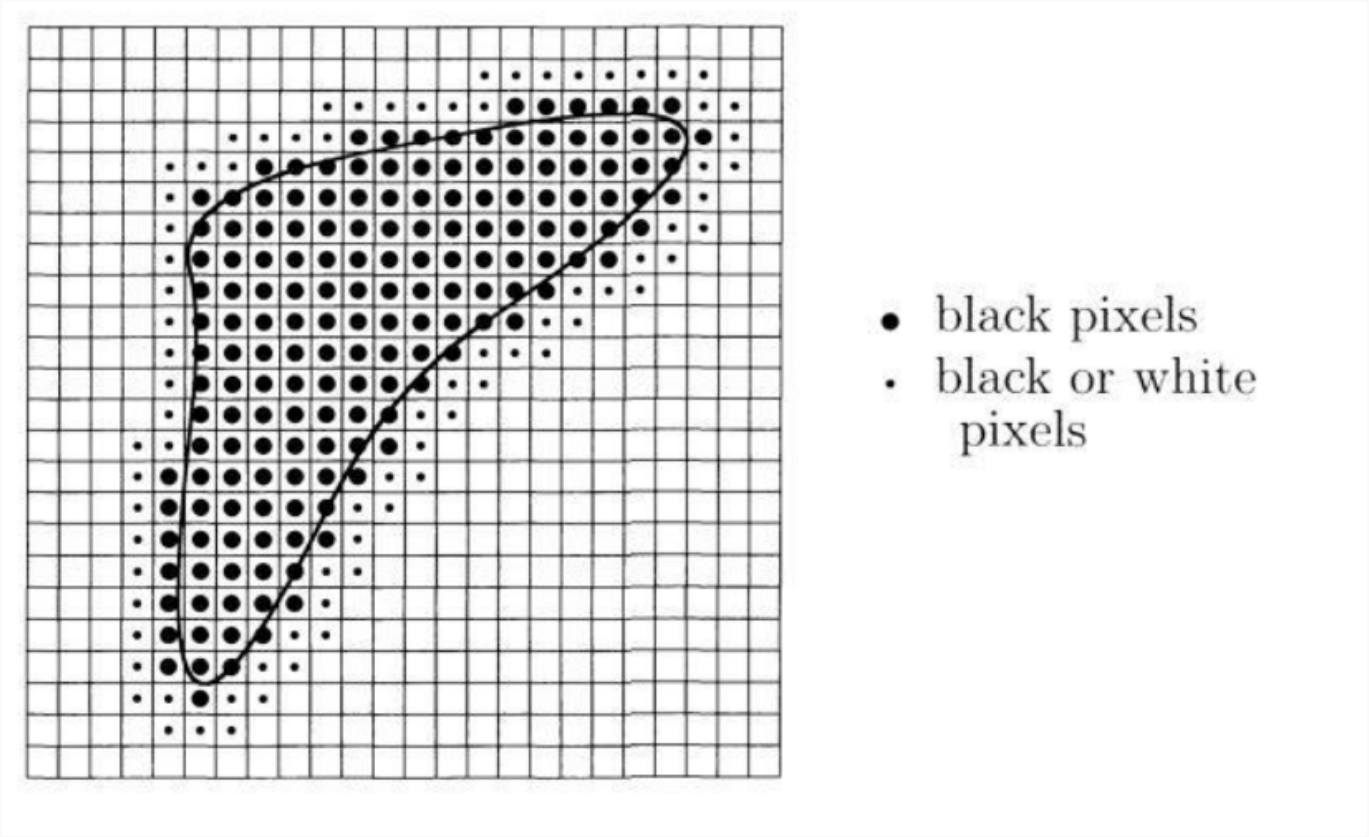
And With Reachability Properties

Theorem (4.3.2)

Consider a dynamical system, with $\mathbf{f}$ *locally Lipschitz, computable, whose domain is a computable compact, then, for all* $\mathbf{x}$, $cls(R_\omega^{\mathcal{H}}(\mathbf{x})) \subseteq \mathbb{R}^d$ *is a co-computably enumerable closed subset.*

Plot twist

From CA, Computable $\Leftrightarrow$ can be plotted.

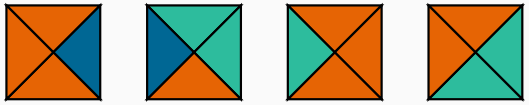


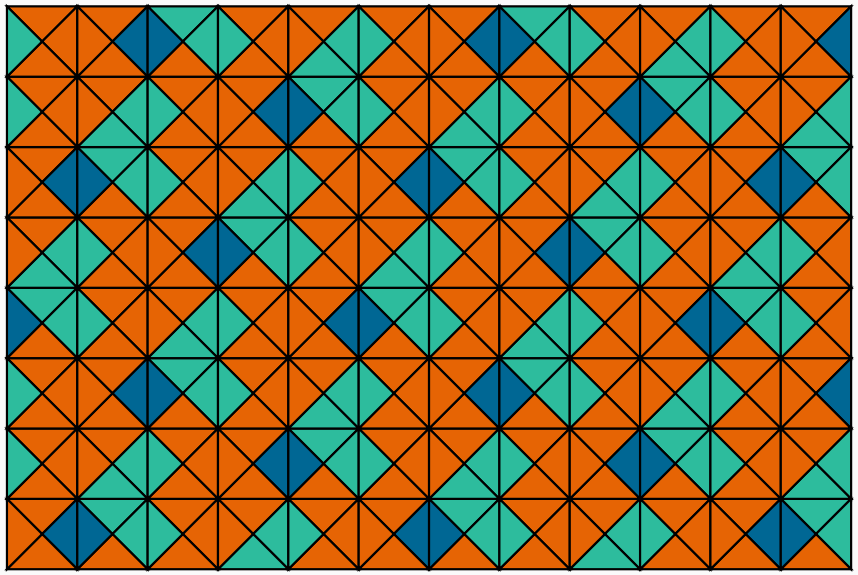
Theorem (4.3.5)

$R^{\mathcal{H}}$ closed and can be plotted in a name of $\mathbf{f}$ $\Leftrightarrow$ *the system is robust, i.e.* $R^{\mathcal{H}}_{\omega} = R^{\mathcal{H}}$.

Toward Complexity for tilings

Wang tilesets





Robinson

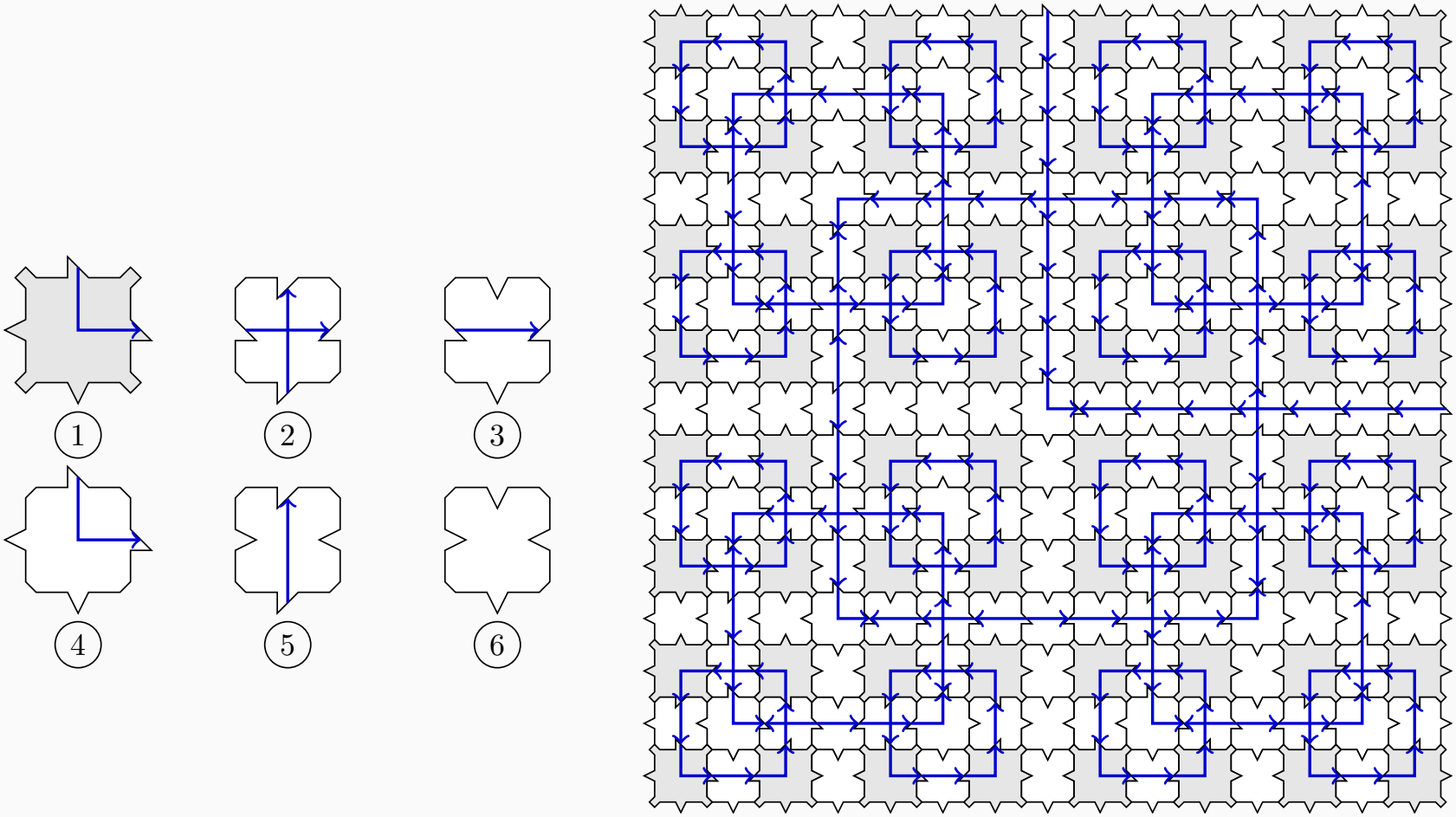


Figure 1: To the left is Robinson's tileset, where tiles can be rotated and reflected. To the right a pattern that appears in every tiling by Robinson's tileset.

Computability Properties

- The domino problem is undecidable...
- ... more precisely, it is not computably enumerable;
- It becomes decidable for a sub-class of tilesets: robust tilesets.

$\rightarrow$ Chapter 5

Conclusion

Conclusion

- We extended the motto **time = length** to discrete ODEs (chapter 3);

Conclusion

- We extended the motto **time = length** to discrete ODEs (chapter 3);
- We proved that **space = precision** (chapter 5);
  $\rightarrow$ proper notions of costs for analogue models of computation;

Conclusion

- We extended the motto **time = length** to discrete ODEs (chapter 3);
- We proved that **space = precision** (chapter 5);
  $\rightarrow$ proper notions of costs for analogue models of computation;
- We studied what makes reachability decidable for robust dynamical systems and prove that space complexity is linked to the precision (chapter 4);

Conclusion

- We extended the motto **time = length** to discrete ODEs (chapter 3);
- We proved that **space = precision** (chapter 5);
  $\rightarrow$ proper notions of costs for analogue models of computation;
- We studied what makes reachability decidable for robust dynamical systems and prove that space complexity is linked to the precision (chapter 4);
- We apply this principle to tilings (chapter 6).

Conclusion: Perspectives

- Minimising the algebras?
- Would it work for GPAC and polynomial ODEs?
- Can we relate this approch to second-order complexity?
- Can we relate more thinly our notions of robustness?

Conclusion: Perspective

- Studying non-determinism, and polynomial hierarchy;
- Algebraically characterising probabilistic classes;
- Studying the complexity for more general differential equations, e.g. distributions.

Appendix: Algebraic characterisation of PTIME for sequences

We consider:

$$\overline{\mathbb{LDL}^\bullet} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \times, \overline{\mathrm{cond}}(x), \frac{x}{2};$$

composition, *linear length ODE*, *ELim*]

Theorem (B., Bournez, MCU22 Best Student Paper Award)

$$\mathsf{FPTIME} \cap \mathbb{R}^{\mathbb{N}} = \overline{\mathbb{LDL}^\bullet} \cap \mathbb{R}^{\mathbb{N}}$$

Appendix: FPTIME for computable functions over the reals

We consider:

$$\overline{\mathbb{LDL}^\circ} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \tanh, \frac{x}{2}, \frac{x}{3};$$

$$\textit{composition}, \textit{linear length ODE}, \textit{ELim}]$$

Theorem (B., Bournez, MFCS23 Best Paper Award)

$$\overline{\mathbb{LDL}^\circ} \cap \mathbb{R}^{\mathbb{R}} = \mathsf{FPTIME} \cap \mathbb{R}^{\mathbb{R}}$$

Appendix: FPSPACE for computable functions over the reals

We consider:

$$\overline{\mathbb{RLD}^\circ} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(n), +, -, \tanh, \frac{x}{2}, \frac{x}{3};$$

$$\textit{composition}, \textit{robust linear ODE}, \textit{ELim}]$$

Theorem (B., Bournez, MFCS23 Best Paper Award)

$$\overline{\mathbb{RLD}^\circ} \cap \mathbb{R}^{\mathbb{R}} = \mathsf{FPSPACE} \cap \mathbb{R}^{\mathbb{R}}$$

Appendix: An unusual way of solving an ODE

- Computation at $t = x$ at precision 2^{-n}, with initial condition $t = 0$
- Computation at $t = t_0 + \frac{x}{2}$ at precision $2^{-\eta(n)}$, with initial condition $t_0 = 0$ and $t_0 = \frac{x}{2}$
- Computation at $t = t_0 + \frac{x}{4}$ at precision $2^{-\gamma(n)}$, with initial condition $t_0 = 0$ and $t_0 = \frac{x}{4}$ and $t_0 = \frac{x}{2}$ and $t_0 = \frac{3x}{4}$