Robust computation with dynamical systems

Olivier Bournez[1] Daniel S. Graça[2] Emmanuel Hainry[3]

[1]École Polytechnique, LIX, Palaiseau, France

[2]DM/FCT, Universidade do Algarve, Faro & SQIG/Instituto de Telecomunicações, Lisbon, Portugal

[3]LORIA & Université de Lorraine, Nancy, France

Dynamical Systems and Computability
Lyon, December 17 2013

Introduction

- Dynamical Systems consist of
  - a configuration space,
  - a dynamics function: maps a configuration to the "following".

  ▸ Definition
- Interests:
  - Describing physical phenomena
    e.g. Fluid dynamics, Gravitation, Weather...
  - Describing biological phenomena
    e.g. Population dynamics, Epidemiologics...
  - Simulating computation models

  ▸ Simulating a Turing machine

Questions on Dynamical Systems

- Reachability of a configuration
  e.g. Halting of a Turing machine.
- Verification of safety properties
  e.g. The population does not go extinct.

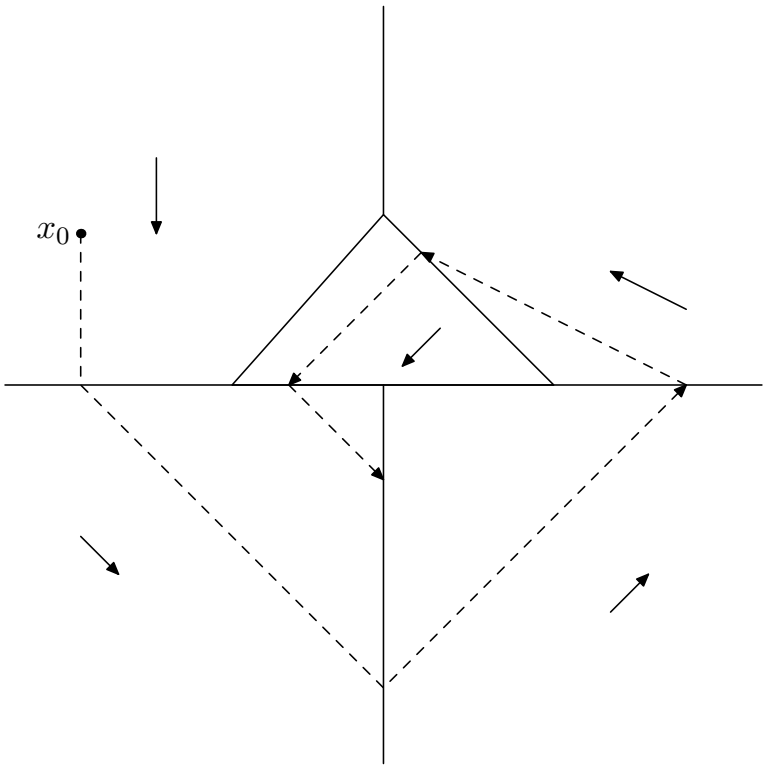
Unfortunately,

- Those problems are undecidable *(even for very restricted classes such as polynomial systems or PCD systems).*
- But decidable for "real systems"

What real systems have:

- Robustness to noise?
- Robustness to perturbations?

Piecewise Constant Derivative



Space: $\mathbb{R}^n$
Partitioned in a finite number of polyhedra

Dynamics: Constant in each polyhedron.

Theorem [Asarin Maler Pnueli, 1995]

If $n \geq 3$, Reachability is undecidable.

Robustness

Asarin and Bouajjani studied robustness to infinitesimal perturbations in the context of Piecewise Constant Derivative systems.

Theorem [Robust ⇒ decidable]

If a PCD is robust, then the language it recognizes is decidable.

Theorem [Perturbed reachability is complete in Π_0^1]

Given a Turing machine $\mathcal{M}$, one can design a PCD whose perturbed recognized language is $\Sigma^\star - L(\mathcal{M})$

Theorem [Decidable ⇒ robust]

Given a Turing machine $\mathcal{M}$, one can design a robust PCD that recognizes $L(\mathcal{M})$

Goal

PCD is not satisfactory

- Physically: not $\mathcal{C}^1$
- Mathematically: very restricted class
- Analog computation: Shannon's General Purpose Analog Computer

$\Rightarrow$ Similar results on smooth computable systems

Dynamical systems

Definition [Discrete time dynamical systems]

$\mathcal{H} = (X, f)$ with $X \subset \mathbb{R}^d$ and $f : X \to X$ defines a discrete dynamical system.
A trajectory is a sequence $\{x_0, x_1, ..., \} \in X^{\mathbb{N}}$ s.t.
$\forall n, x_{n+1} = f(x_n)$.

Definition [Continuous time dynamical systems]

$\mathcal{H} = (X, f)$ with $X \subset \mathbb{R}^d$ and $f : X \to X$ defines a continuous dynamical system.
A trajectory is a solution of $\begin{cases} x' = f(x) \\ x(0) = x_0 \end{cases}$

Dynamical Systems

We will study 2 kinds of domains:

- Bounded domain: $X = [-1, 1]^d$
- Unbounded domain: $X = \mathbb{R}^d$

Smooth dynamical systems

Definition [Lipschitz]

$f : \mathbb{R}^m \to \mathbb{R}^k$ is Lipschitz over X iff

$$\exists K > 0; \forall x, y \in X, \|f(x) - f(y)\| \leq K\|x - y\|.$$

Definition [Smooth dynamical system]

A continuous-time dynamical system $\mathcal{H} = (X, f)$ is smooth if f is Lipschitz.

Note If f is Lipschitz, $\begin{cases} x' = f(t, x) \\ x(t_0) = x_0 \end{cases}$ has a unique solution.

Dynamical Systems as language recognizers

Let $\Sigma = \{0, 1\}$ and $\Sigma^\star$ the set of words over Σ.

Definition [Encoding words in $\mathbb{R}$]

$$\nu : \Sigma^\star \to \mathbb{R}$$

$$\nu(w) = \sum_{i=0}^{n} 2^i \times w_i$$

where $w = w_0 w_1 w_2 ... w_n$ with $w_1, w_2, ..., w_n \in \Sigma$.

Definition [Encoding words in $-1, 1$]

$$\nu_c : \Sigma^\star \to [-1, 1]$$

$$\nu_c(w) = \frac{2}{\pi} \arctan(\nu(w))$$

Dynamical Systems as language recognizers

Definition

Given a dynamical system $\mathcal{H} = (X, f)$ and a set $V_{accept} \subset X$.

$$L_{\mathcal{H}} = \{w \in \Sigma^*; \mathcal{T}_{\mathcal{H}}(\sigma(w)) \cap V_{accept} \neq \emptyset\}$$

Perturbations and robustness

Definition [ε-perturbation]

Let $\mathcal{H} = (X, f)$ a dynamical system.
Given $\varepsilon > 0$, $\mathcal{H}_\varepsilon$ is the system defined over the same space X, where:

1. $(\mathbf{x}_0, \mathbf{x}_1, ...)$ is a trajectory[1] of $\mathcal{H}_\varepsilon$ if $\|\mathbf{x}_{i+1} - f(\mathbf{x}_i)\| \leq \varepsilon$ for all $i \in \mathbb{N}$;
2. $\phi : \mathbb{R}_0^+ \to X$ is a trajectory[1] of $\mathcal{H}_\varepsilon$ if $\|\phi'(t) - f(\phi(t))\| \leq \varepsilon$ for all $t \in \mathbb{R}_0^+$.

Note $\mathcal{H}_\varepsilon$ is not a dynamical system.

[1] The trajectory may not be unique.

Perturbations and robustness

Definition [Perturbed system as language recognizers]

L_ε, language recognized by $\mathcal{H}_\varepsilon$, is defined by:
$w \in L_\varepsilon$ iff $\exists$ trajectory from $\sigma(w)$ reaching V_{accept}.

Lemma

For $0 < \varepsilon < \varepsilon'$, $\quad L \subset L_\varepsilon \subset L_{\varepsilon'}$.

Definition

$$L_\omega = \bigcap_{\varepsilon > 0} L_\varepsilon.$$

Perturbations and robustness

Intuitively, a system is said robust if infinitesimal perturbations do not change the recognized language.

Definition [Robustness]

A dynamical system $\mathcal{H}$ is said robust iff $L = L_\omega$

Accepted language

For a Dynamical System to be a reliable language accepter, it should have disjoint open computable sets V_{accept} and $V_{compute}$ s.t.

- V_{accept}, $V_{compute}$ are separated.
- Trajectories escaping $V_{compute}$ will reach V_{accept} and stay in V_{accept}
- $\forall w, \sigma(w) \in V_{compute}$

Computable sets

Definition

- $E \subset \mathbb{R}^d$ is r.e. open set iff there exist computable sequences of rationals (a_n) and (r_n) s.t.

$$E = \bigcup_{n=0}^{\infty} B(a_n, r_n)$$

  with $B(a, r)$ open ball of center a and radius r.
- F is r.e. closed if $\exists$ a family of rational tuples $\{c_n\}$ that is dense in F.
- E is an open computable set iff it is r.e. open and its complement is r.e. closed.

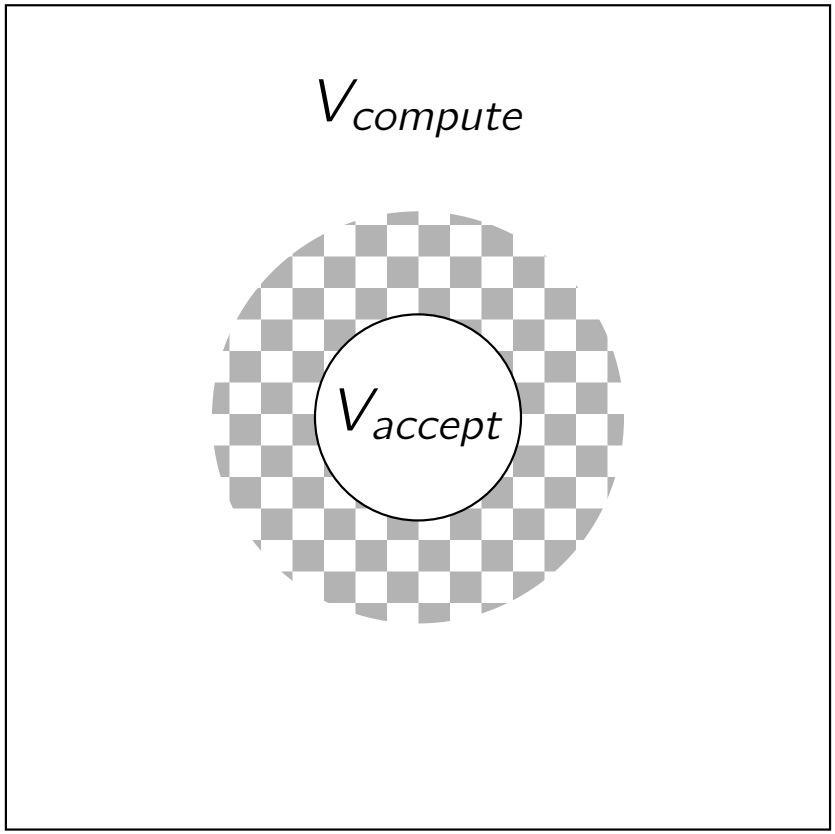
DS as language accepters

Definition

$\mathcal{H}$ over $X = [-1, 1]^d$.
$V_{accept} = \{x \in X; \|x\| < 1/4\}$
$V_{compute} = \{x \in X; \|x\| > 1/2\}$

$\mathcal{H}$ accepts $L \subset \Sigma^\star$ if
$w \in L$ iff the trajectory from $(\nu(w), 1, ..., 1, 1)$ reaches V_{accept}.

If $w \notin L$, the trajectory must never leave $V_{compute}$.



The zones $V_{compute}$ and V_{accept}.

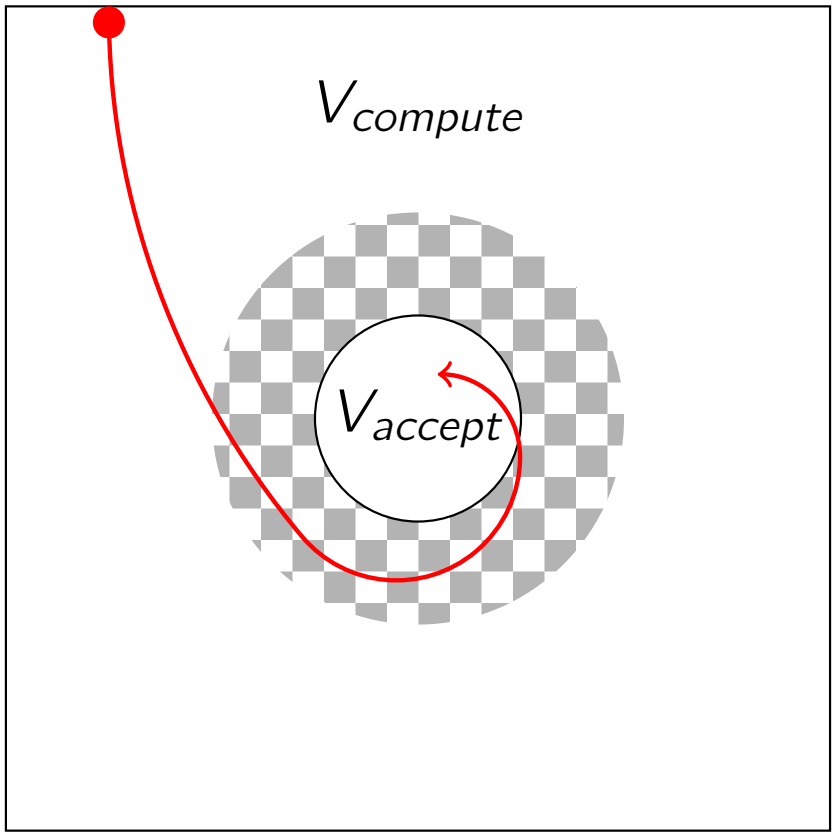
DS as language accepters

Definition

$\mathcal{H}$ over $X = [-1, 1]^d$.
$V_{accept} = \{x \in X; \|x\| < 1/4\}$
$V_{compute} = \{x \in X; \|x\| > 1/2\}$

$\mathcal{H}$ accepts $L \subset \Sigma^\star$ if
$w \in L$ iff the trajectory from $(\nu(w), 1, ..., 1, 1)$ reaches V_{accept}.

If $w \notin L$, the trajectory must never leave $V_{compute}$.



An accepted trajectory goes into V_{accept}.

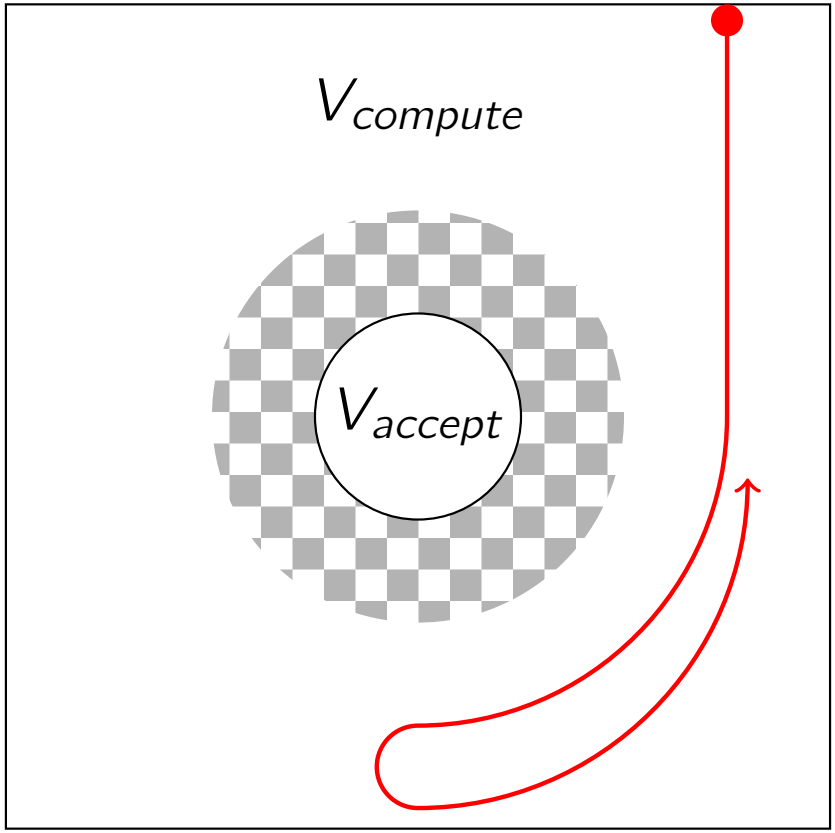
DS as language accepters

Definition

$\mathcal{H}$ over $X = [-1, 1]^d$.
$V_{accept} = \{x \in X; \|x\| < 1/4\}$
$V_{compute} = \{x \in X; \|x\| > 1/2\}$

$\mathcal{H}$ accepts $L \subset \Sigma^\star$ if
$w \in L$ iff the trajectory from $(\nu(w), 1, ..., 1, 1)$ reaches V_{accept}.

If $w \notin L$, the trajectory must never leave $V_{compute}$.



A refused trajectory never enters $\mathcal{B}(1/2)$.

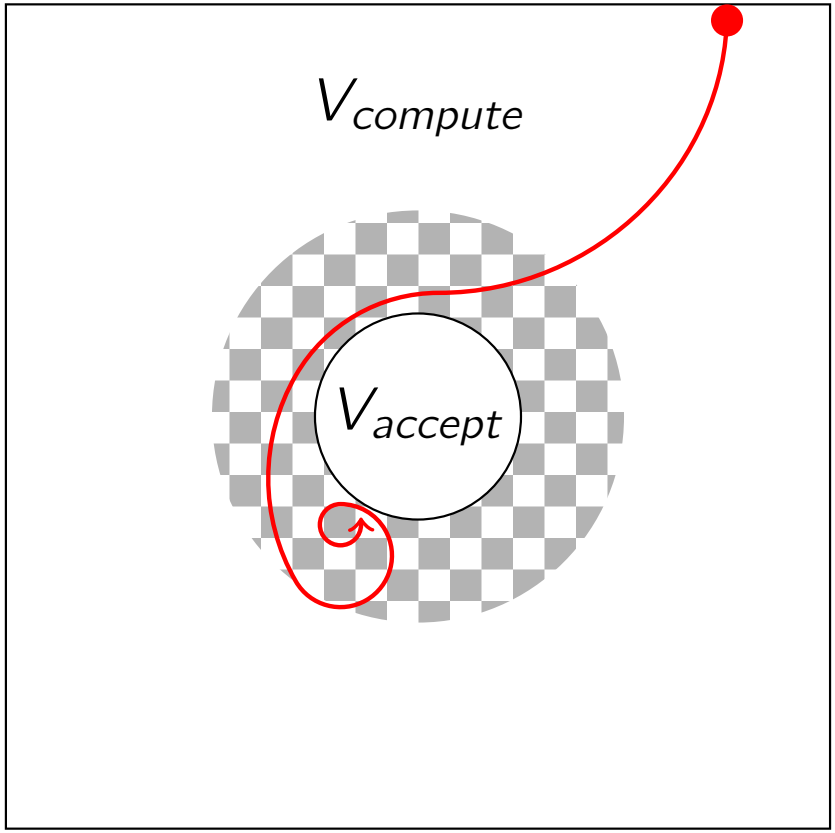
DS as language accepters

Definition

$\mathcal{H}$ over $X = [-1, 1]^d$.
$V_{accept} = \{x \in X; \|x\| < 1/4\}$
$V_{compute} = \{x \in X; \|x\| > 1/2\}$

$\mathcal{H}$ accepts $L \subset \Sigma^\star$ if
$w \in L$ iff the trajectory from
$(\nu(w), 1, ..., 1, 1)$ reaches
V_{accept}.

If $w \notin L$, the trajectory must never leave $V_{compute}$.



A forbidden trajectory.

Discrete compact result

Theorem [L_ω is co-r.e. for Lipschitz computable discrete systems]

Let L language recognized by $\mathcal{H} = ([-1,1]^d, f)$ with f computable and Lipschitz.
L_ω is co-recursively enumerable.

Proof Partition X in d-dimensional hypercubes of size $1/n$.

Design a finite automaton A_n which will roughly recognize $L_{\frac{1}{n}}$ (s.t. rejection implies $w \notin L_{1/n}$).

Simulate A_n with n growing until we know that w is rejected.

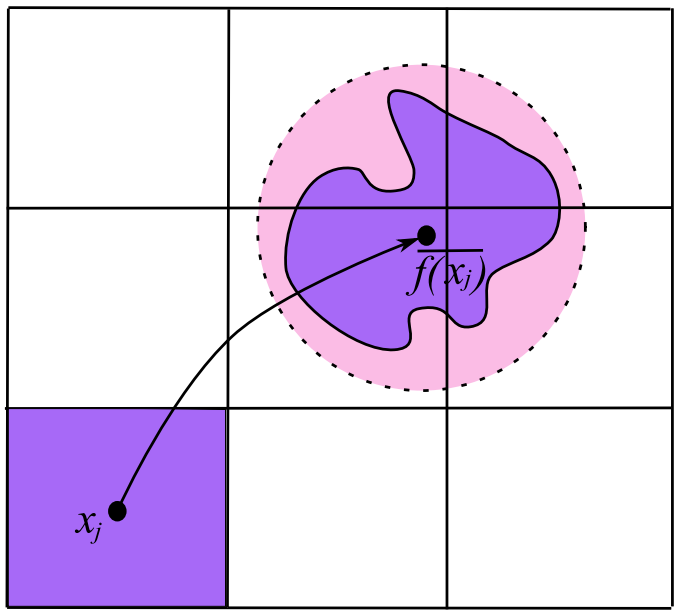
L_ω is co-r.e. for Lipschitz computable discrete systems

- Design of A_n

States V_1, ..., V_s hypercubes of size $1/n$.

Accepting V_is that intersect V_{accept}.

Transition $V_j \to V_k$ if $\{x \in X; \|f(x) - f(x_j)\| \leq \frac{K+2}{n}\} \cap V_k \neq \emptyset$.



- $L_{1/n} \subseteq L_{A_n} \subseteq L_{\frac{K+3}{n}} \quad \Rightarrow \quad \bigcap L_{A_n} = L_\omega$.

Continuous compact result

Definition [Stroboscopic map]

$\mathcal{H} = (X, f)$ a continuous-time dynamical system.
The stroboscopic map $g : X \to X$ is defined by:
$g(x) = \phi(1)$ where $\phi(0) = x$ and $\phi'(t) = f(\phi(t))$.

Lemma

Let $\mathcal{H} = (X, f)$ a continuous-time dynamical system.
$\widehat{\mathcal{H}} = (X, g)$ discrete-time stroboscopic version.

- $L = \widehat{L}$
- $L_\omega = \widehat{L}_\omega$ *and* $\forall \varepsilon > 0, L_\varepsilon = \widehat{L}_\varepsilon$.

Continuous compact result

Theorem [L_ω is co-r.e. for Lipschitz computable continuous systems]

Let L language recognized by $\mathcal{H} = ([-1,1]^d, f)$ with f computable and Lipschitz.
L_ω is co-recursively enumerable.

Proof The stroboscopic map g is Lipschitz and computable.

Robust ⇒ Recursive

Lemma

$\mathcal{H} = ([-1,1]^d, f)$ smooth system (f Lipschitz and computable). L is recursively enumerable.

Theorem [Robust ⇒ Recursive]

$\mathcal{H} = ([-1,1]^d, f)$ smooth system (f Lipschitz and computable). If $\mathcal{H}$ is robust, then L is recursive.

Unbounded case

The automaton construction does not work in the unbounded case.

Unbounded case

The automaton construction does not work in the unbounded case.

Theorem [Collins]

The compactness condition is necessary.

(this is a free translation of Theorem 4.9 from Collins, Pieter. "Continuity and computability of reachable sets." *Theoretical Computer Science* 341, no. 1 (2005): 162-195.)

Unbounded case

The automaton construction does not work in the unbounded case.

Theorem [Collins]

The compactness condition is necessary.

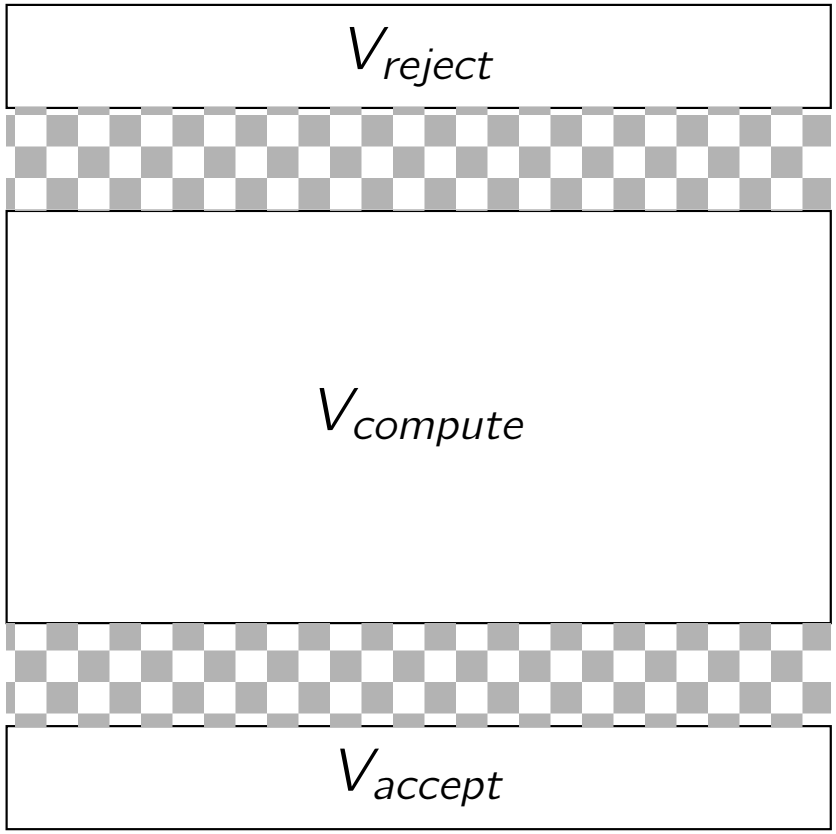
(this is a free translation of Theorem 4.9 from Collins, Pieter. "Continuity and computability of reachable sets." *Theoretical Computer Science* 341, no. 1 (2005): 162-195.)

However, our "reject" condition can be improved.

Intuition: Robust language decision

Sandwich model:

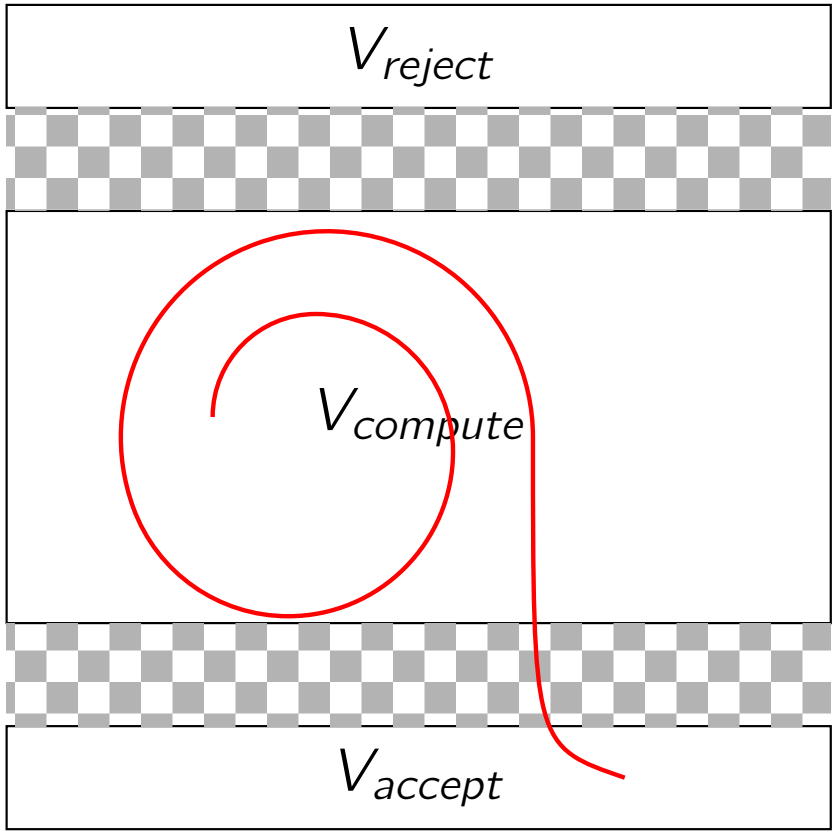
- 3 separated zones: V_c, V_a and V_r.
- V_r and V_a are sinks.



Intuition: Robust language decision

Sandwich model:

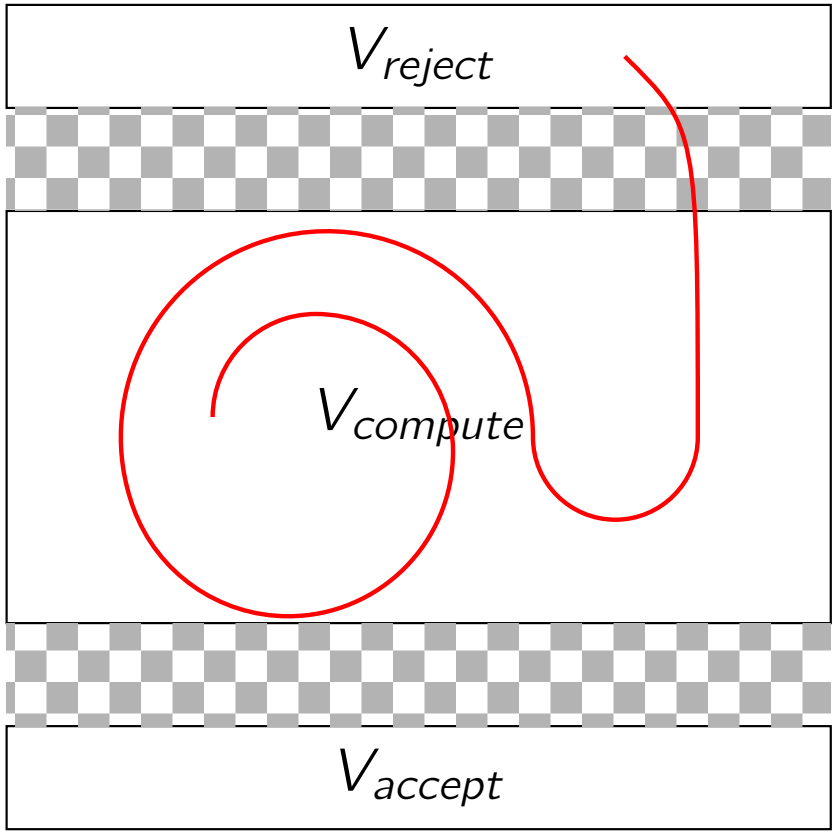
- 3 separated zones: V_c, V_a and V_r.
- V_r and V_a are sinks.



Intuition: Robust language decision

Sandwich model:

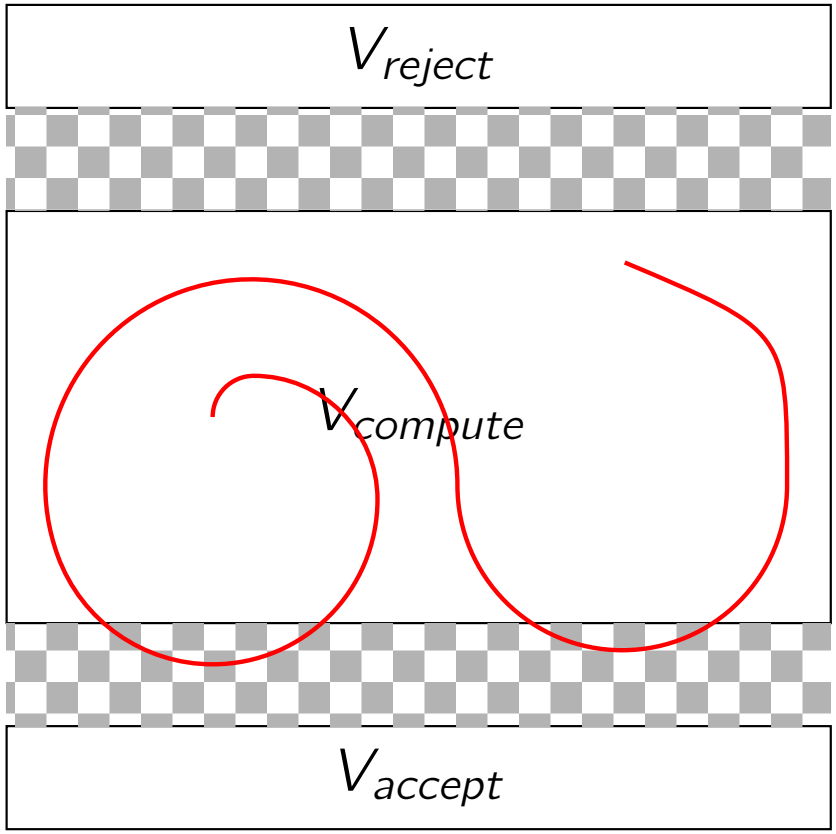
- 3 separated zones: V_c, V_a and V_r.
- V_r and V_a are sinks.



Intuition: Robust language decision

Sandwich model:

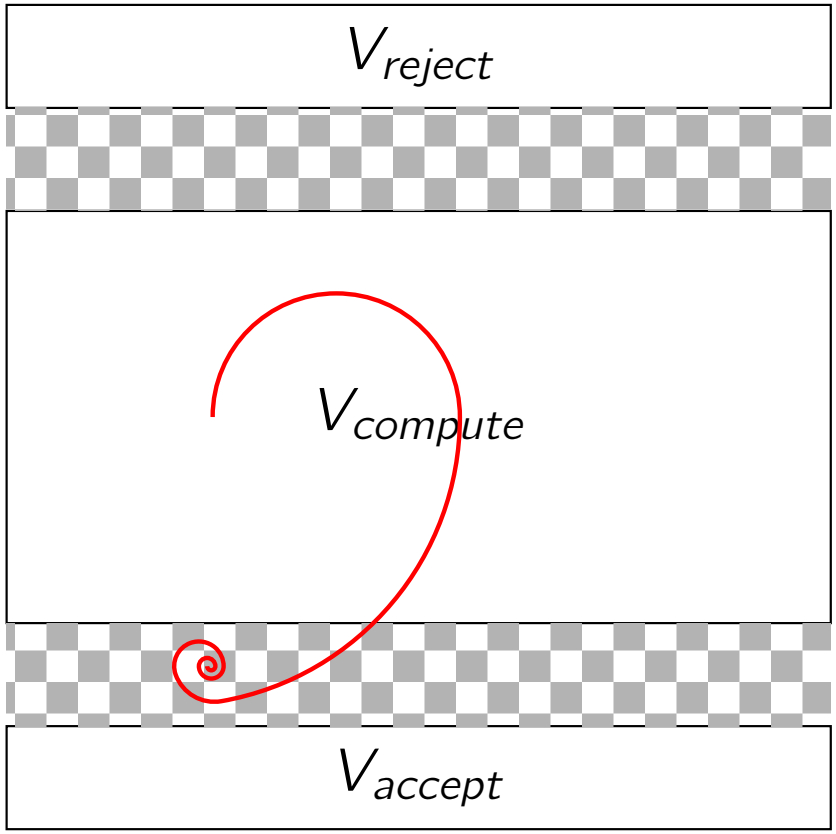
- 3 separated zones: V_c, V_a and V_r.
- V_r and V_a are sinks.



Intuition: Robust language decision

Sandwich model:

- 3 separated zones: V_c, V_a and V_r.
- V_r and V_a are sinks.



Language Recognizer

For a Dynamical System to be a reliable language recognizer, it should have disjoint open computable sets V_{accept}, V_{reject} and $V_{compute}$ s.t.

- V_{accept}, V_{reject} and $V_{compute}$ are separated.
- $\exists$ open set A such that $V_{accept} \cup V_{compute} \subset A$ and $A \cap V_{reject} = \emptyset$
- $\exists$ open set R such that $V_{reject} \cup V_{compute} \subset R$ and $A \cap V_{accept} = \emptyset$
- All correct trajectories will escape $V_{compute}$ and reach V_{accept} or V_{reject} and stay there.
- $\forall w, \sigma(w) \in V_{compute}$

Unbounded Result

Both in the discrete and continous case, we have:

Theorem [L_ω is co-r.e.]

Let L language robustly recognized by $\mathcal{H} = (\mathbb{R}^d, f)$ with f computable and effectively locally Lipschitz.
L is recursive

Recursive ⇒ Robust

If a language is recursive, we can implement a dynamical system recognizing robustly (with a Lipschitz dynamic).
Here, the easy case is the unbounded one:

Theorem

Given a Turing machine M, there exist a *(discrete/continous)* dynamical system over $\mathbb{R}^d$ that recognizes L which is robust to $\frac{1}{4}$ perturbation.

Recursive ⇒ Robust

If a language is recursive, we can implement a dynamical system recognizing robustly (with a Lipschitz dynamic).
Here, the easy case is the unbounded one:

Theorem [Graça Campagnolo Buescu]

Given a Turing machine M, there exist a continuous-time dynamical system over $\mathbb{R}^6$ that recognizes L which is robust to $\frac{1}{4}$ perturbation.
And the dynamics is effectively locally Lipschitz

Recursive ⇒ Robust (bounded case)

The previous simulation uses 6 components:

- 2 for the left part of the Turing tape,
- 2 for the right part of the Turing tape,
- 2 for the state.

States being finite, robustly compacting their space is easy.
For the other components, compose through $\frac{2}{\pi}\arctan$.

Problem Lipschitz?

Problem Robust?

Recursive ⇒ Robust (bounded case)

Theorem

Let A be a recursive language. There is an analytic and computable continuous-time system over $(-1,1)^6$ that robustly recognizes A.

Since all computation halt, there is an ϵ under which the perturbations do not alter the result.
It is Lipschitz on the state space.

Conclusion

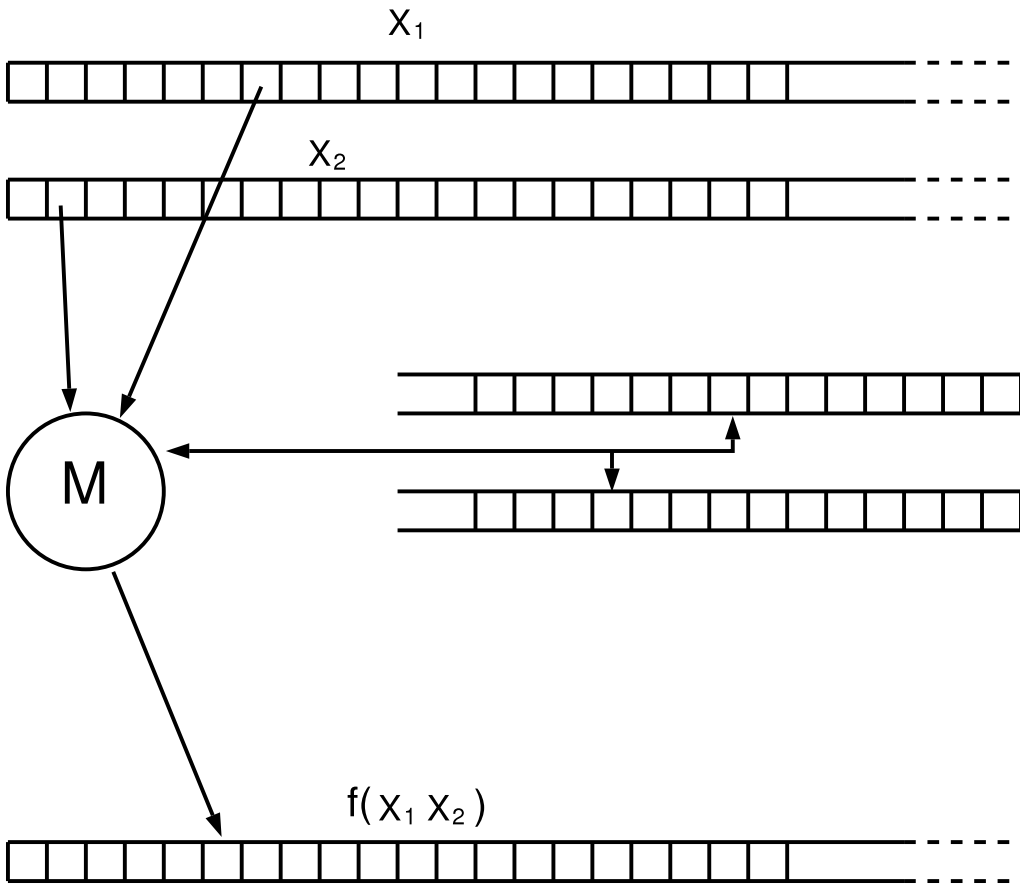
- The intuition "robustness implies computability" is indeed true for this kind of systems.
- Robustness is not too strong an hypothesis ("computability implies robustness").
- A good candidate for modelizing real systems?

Recursive analysis

Idea Reals are represented by Cauchy sequences of rationals.

An infinite tape can contain a real.

A Turing machine manipulates those tapes.



Recursive analysis

Definition [Representation of real numbers]

$x \in \mathbb{R}$ can be represented by a sequence (x_n) s.t.
$\forall n, \|x_n - x\| < 2^{-n}$.
This is denoted $(x_n) \rightsquigarrow x$.

Definition [Computable functions]

A function $f : \mathbb{R} \to \mathbb{R}$ is computable iff there exists an oracle Turing machine M s.t. $\forall (x_n) \rightsquigarrow x$, M taking (x_n) as oracle computes $(y_n) \rightsquigarrow f(x)$.

◂ Return

Example: Linear recurrent sequence

$$X = \mathbb{Z}^2$$

$$f : (x, y) \mapsto (x + y, x)$$

A trajectory: $(1, 1) \to (2, 1) \to (3, 2) \to (5, 3) \to (8, 5) \to \ldots$.

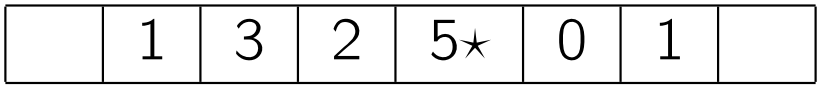
This system can also be written

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Example: Turing machine

Let $\mathcal{M}$ be a one-tape Turing machine
$\delta : \mathbb{N} \times \{0, .., 9\} \to \mathbb{N} \times \{0, .., 9\} \times \{g, r, d\}$

$X = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ (state, left and right part of the tape)



is represented by $a = 1325$ and $b = 10$

$f : (n, a, b) \mapsto$ (n', a', b') with ab representing the tape
$\sigma = a \bmod 10$; $\delta(n, \sigma) = (n', \sigma', \tau)$
$a' = a - \sigma + \sigma'$; $b' = b$ if $\tau = r$
$a' = a/10$; $b' = 10 * b + \sigma'$ if $\tau = d$
$b' = b/10$; $a' = 10 * (a - \sigma + \sigma') + (b \bmod 10)$ if $\tau =$

◂ Return