Comput-, predict- and decidability in symbolic dynamical systems

Benjamin Hellouin de Menibus

Algorithmic questions in dynamical systems

LRI, Université Paris Sud

March 26, 2018

Two types of models of computation:

- **Functional models**, e.g. recursive functions, λ-calculus;
- **Dynamical models**, e.g. Turing machines, counter machines, cellular automata.

Classical dynamical models are **symbolic dynamical systems**: continuous transformations of $\mathcal{A}^{\mathbb{N}}$ or $\mathcal{A}^{\mathbb{Z}}$ (or a subset thereof) with the Cantor topology, where $\mathcal{A}$ is a finite alphabet.

Dynamical models of computations

Two types of models of computation:

- **Functional models**, e.g. recursive functions, λ-calculus;
- **Dynamical models**, e.g. Turing machines, counter machines, cellular automata.

Classical dynamical models are **symbolic dynamical systems**: continuous transformations of $\mathcal{A}^{\mathbb{N}}$ or $\mathcal{A}^{\mathbb{Z}}$ (or a subset thereof) with the Cantor topology, where $\mathcal{A}$ is a finite alphabet.

From here:

Which dynamical systems are models of computation?

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary** / **universal computation**;

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary** / **universal computation**;
- Its halting problem is **undecidable** (or as hard as for Turing machines);

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary / universal computation**;
- Its **unbounded-time prediction** problem is **undecidable** (or as hard as for Turing machines);

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary / universal computation**;
- Its **unbounded-time prediction** problem is **undecidable** (or as hard as for Turing machines);

(and “everything is undecidable”, i.e. Rice-style theorem)

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation...) if:

- It can embed **arbitrary** / **universal computation**;
- Its **unbounded-time prediction** problem is **undecidable** (or as hard as for Turing machines);
  (and “everything is undecidable”, i.e. Rice-style theorem)
- It is universal à la Blondel-Delvenne-Kurka;

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary / universal computation**;
- Its **unbounded-time prediction** problem is **undecidable** (or as hard as for Turing machines);
  (and "everything is undecidable", i.e. Rice-style theorem)
- It is universal à la Blondel-Delvenne-Kurka;
- The **bounded-time prediction** problem is **P**-complete;

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary** / **universal computation**;
- Its **unbounded-time prediction** problem is **undecidable** (or as hard as for Turing machines);
  (and “everything is undecidable”, i.e. Rice-style theorem)
- It is universal à la Blondel-Delvenne-Kurka;
- The **bounded-time prediction** problem is **P**-complete;
- Its single limit points are the same as Turing machines;

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary / universal computation**;
- Its **unbounded-time prediction** problem is **undecidable** (or as hard as for Turing machines);
  (and “everything is undecidable”, i.e. Rice-style theorem)
- It is universal à la Blondel-Delvenne-Kurka;
- The **bounded-time prediction** problem is **P**-complete;
- Its single limit points are the same as Turing machines;
  (and maybe sets of limit points as well)

A symbolic dynamical system is **Turing-universal** (or **Turing-complete**, or is a model of computation. . .) if:

- It can embed **arbitrary** / **universal computation**;
- Its **unbounded-time prediction** problem is **undecidable** (or as hard as for Turing machines);
  (and "everything is undecidable", i.e. Rice-style theorem)
- It is universal à la Blondel-Delvenne-Kurka;
- The **bounded-time prediction** problem is **P**-complete;
- Its single limit points are the same as Turing machines;
  (and maybe sets of limit points as well)
- It is able of "robust" computation, i.e. on random points or subjected to noise.

Definition

The **shift action** is defined as $\sigma(x)_i = x_{i+1}$.

Dynamical models of computation

Definition

The **shift action** is defined as $\sigma(x)_i = x_{i+1}$.

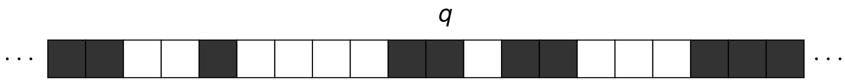
Standard (moving-tape) **Turing machine**

Given two finite set of states Q and symbols $\mathcal{A}$ and a decision function

$$\delta : Q \times \mathcal{A} \mapsto Q \times \mathcal{A} \times \{-1, 0, 1\},$$

For $q, x \in Q \times \mathcal{A}^{\mathbb{Z}}$, $TM(q, x)$ is determined as follows:

- The new state is $\delta_1(q, x_0)$;
- The new configuration is obtained by replacing x_0 by $\delta_2(q, x_0)$ and shifting by $\delta_3(q, x_0)$.



$$\delta(q, \blacksquare) = (q', \square, +1)$$

Dynamical models of computation

Definition

The **shift action** is defined as $\sigma(x)_i = x_{i+1}$.

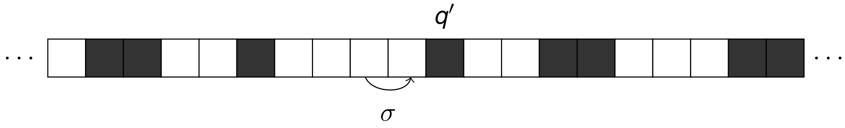
Standard (moving-tape) **Turing machine**

Given two finite set of states Q and symbols $\mathcal{A}$ and a decision function

$$\delta : Q \times \mathcal{A} \mapsto Q \times \mathcal{A} \times \{-1, 0, 1\},$$

For $q, x \in Q \times \mathcal{A}^{\mathbb{Z}}$, $TM(q, x)$ is determined as follows:

- The new state is $\delta_1(q, x_0)$;
- The new configuration is obtained by replacing x_0 by $\delta_2(q, x_0)$ and shifting by $\delta_3(q, x_0)$.



Dynamical models of computation

Definition

The **shift action** is defined as $\sigma(x)_i = x_{i+1}$.

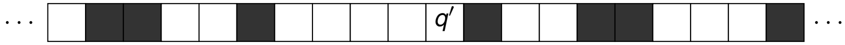
Standard (moving-tape) **Turing machine**

Given two finite set of states Q and symbols $\mathcal{A}$ and a decision function

$$\delta : Q \times \mathcal{A} \mapsto Q \times \mathcal{A} \times \{-1, 0, 1\},$$

For $q, x \in Q \times \mathcal{A}^{\mathbb{Z}}$, $TM(q, x)$ is determined as follows:

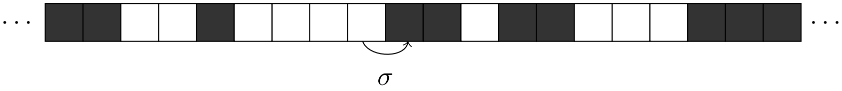
- The new state is $\delta_1(q, x_0)$;
- The new configuration is obtained by replacing x_0 by $\delta_2(q, x_0)$ and shifting by $\delta_3(q, x_0)$.



Cellular automata

A cellular automaton is a **continuous, σ-commuting** transformation of $\mathcal{A}^{\mathbb{Z}}$.

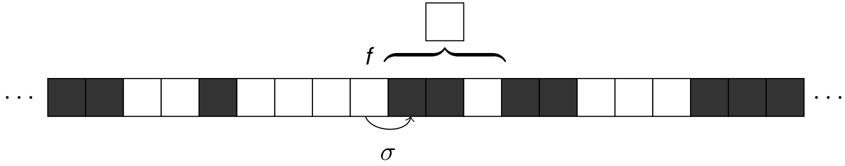
Equivalently, it can be defined by a local rule over a finite window:



Cellular automata

A cellular automaton is a **continuous, σ-commuting** transformation of $\mathcal{A}^{\mathbb{Z}}$.

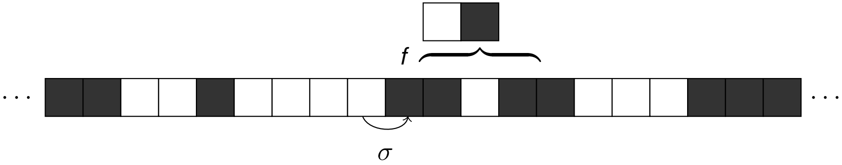
Equivalently, it can be defined by a local rule over a finite window:



Cellular automata

A cellular automaton is a **continuous, σ-commuting** transformation of $\mathcal{A}^{\mathbb{Z}}$.

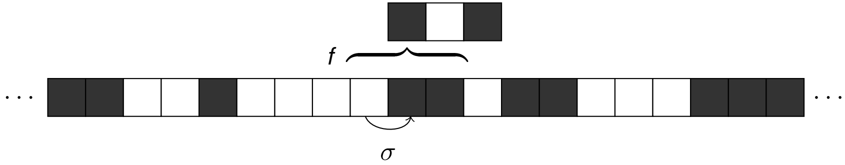
Equivalently, it can be defined by a local rule over a finite window:



Cellular automata

A cellular automaton is a **continuous, σ-commuting** transformation of $\mathcal{A}^{\mathbb{Z}}$.

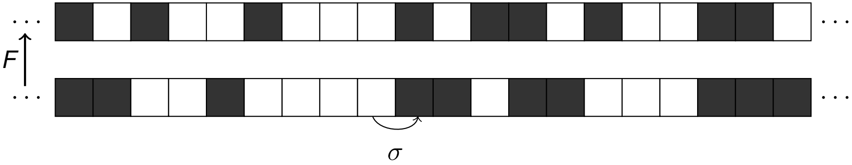
Equivalently, it can be defined by a local rule over a finite window:



Cellular automata

A cellular automaton is a **continuous, σ-commuting** transformation of $\mathcal{A}^{\mathbb{Z}}$.

Equivalently, it can be defined by a local rule over a finite window:



Halting problem

Input An finite word;

Output Does the Turing machine eventually halt on this input?

Theorem (Turing 36)

The halting problem for Turing machines is undecidable.

Halting problem

Input An finite word;

Output Does the Turing machine eventually halt on this input?

Theorem (Turing 36)

The halting problem for Turing machines is undecidable.

Theorem (Rice 51)

Any nontrivial property on the set of input words on which the Turing machine halts is undecidable.

Point-to-set reachability problem for Turing machines

Input An finite point x;

Output Does $TM^t(x)$ eventually reach the set $[h]$?

Theorem (Turing 36)

The point-to-set reachability problem on finite points for Turing machines is undecidable.

Theorem (Rice 51)

Any nontrivial property on the set of input words on which the Turing machine halts is undecidable.

Point-to-set reachability problem for Turing machines

Input An finite point x;

Output Does $TM^t(x)$ eventually reach the set $[h]$?

Theorem (Turing 36)

The point-to-set reachability problem on finite points for Turing machines is Σ_1^0-complete.

Theorem (Rice 51)

Any nontrivial property on the set of input words on which the Turing machine halts is Σ_1^0-hard.

Input/output discussion

Universal system The input is a description of the initial point

Universal class The input is a description of the system + the initial point.

Input/output discussion

Universal system The input is a description of the initial point

Universal class The input is a description of the system + the initial point.

Choice of the input family

- Models of computation work on a **countable** family of states (usual choices: **finite, σ-periodic, almost σ-periodic**);
- Choosing an arbitrary countable family may lead to absurd notions (example later).

Definition (Davis 56)

A system is Turing-universal if its halting problem is Σ^0_1-complete.

Definition (Davis 56)

A system is Turing-universal if its halting problem is Σ^0_1-complete.

Theorem (Banks / Smith III 71)

The point-to-point reachability problem on finite points for cellular automata is Σ^0_1-complete.

Definition (Davis 56)

A system is Turing-universal if its halting problem is Σ_1^0-complete.

Theorem (Banks / Smith III 71)

The point-to-point reachability problem on finite points for cellular automata is Σ_1^0-complete.

Theorem (Cook 00)

The point-to-point reachability problem on almost periodic points for **Rule 110** is Σ_1^0-complete.

(point-to-set and Rice-style theorem follow)

Can we generalize this intuition to an arbitrary countable dense family (Hemmerling 02)?

Can we generalize this intuition to an arbitrary countable dense family (Hemmerling 02)?

Counterexample (Durand, Róka 99)

Take the full shift $(\{0,1\}^{\mathbb{Z}}, \sigma)$ and the set of computable points as input states. Fix:

$$\{x_n = 1^n 0^t 1^\infty : \text{the } n\text{-th TM stops in } t \text{ steps}\} \quad (\text{possibly } t = \infty)$$

Can we generalize this intuition to an arbitrary countable dense family (Hemmerling 02)?

Counterexample (Durand, Róka 99)

Take the full shift $(\{0,1\}^{\mathbb{Z}}, \sigma)$ and the set of computable points as input states. Fix:

$$\{x_n = 1^n 0^t 1^\infty : \text{the } n\text{-th TM stops in } t \text{ steps}\} \quad (\text{possibly } t = \infty)$$

The point-to-set reachability problem:

Input A point x_n;

Output Is $[01]$ reachable under the action of σ?

is Σ^0_1-complete.

Can we generalize this intuition to an arbitrary countable dense family (Hemmerling 02)?

Counterexample (Durand, Róka 99)

Take the full shift $(\{0,1\}^{\mathbb{Z}}, \sigma)$ and the set of computable points as input states. Fix:

$$\{x_n = 1^n 0^t 1^\infty : \text{the } n\text{-th TM stops in } t \text{ steps}\} \quad (\text{possibly } t = \infty)$$

The point-to-set reachability problem:

Input A point x_n;

Output Is $[01]$ reachable under the action of σ?

is Σ^0_1-complete.

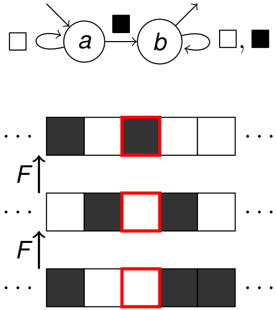
Another problem

The infinite precision on the choice of the initial input might be unphysical and lead to non-robust undecidability results.

A definition in the spirit of set-to-set reachability.

Universality à la Delvenne - Kurka - Blondel

A definition in the spirit of set-to-set reachability.



Model-checking problem (on the trace)

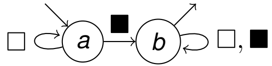
Input A finite automaton labelled by elements of $\mathcal{A}$;

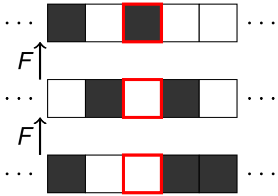
Output Does the automaton reach a halting state for some trajectory?

Equivalently: is the intersection of the trace language w/ some regular language empty?

Universality à la Delvenne - Kurka - Blondel

A definition in the spirit of set-to-set reachability.





Model-checking problem (on the trace)

Input A finite automaton labelled by elements of $\mathcal{A}$;

Output Does the automaton reach a halting state for some trajectory?

Equivalently: is the intersection of the trace language w/ some regular language empty?

Definition (Delvenne, Kurka, Blondel 05)

A system is Turing-universal if its model-checking problem is Σ_1^0-complete.

Definition

A problem is **P**-complete if any problem in **P** can be reduced to it using logarithmic space.

Definition

A problem is **P**-complete if any problem in **P** can be reduced to it using logarithmic space.

Proposition

The bounded-time point-to-set reachability problem for Turing machines:

Input	An input word
Output	Does the machine reach $[h]$ before time t?

is **P**-complete.

Definition

A problem is **P**-complete if any problem in **P** can be reduced to it using logarithmic space.

Proposition

The bounded-time point-to-set reachability problem for Turing machines:

Input	An input word
Output	Does the machine reach $[h]$ before time t?

is **P**-complete.

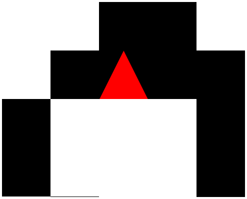
Theorem (Neary, Woods 06)

The bounded-time, point-to-set reachability problem for almost periodic input points in the **rule 110 cellular automaton** is **P**-complete.

A **Turmite** is a two-dimensonal Turing machine on the alphabet $\mathcal{A} = \mathbb{Z}/n\mathbb{Z}$ defined by a rule $\mathcal{A} \to \{R, L\}$.

At each step:

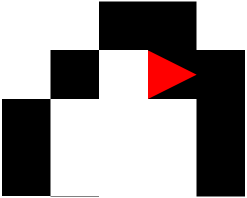
- it increases x_0 by one;
- it shifts the configuration by $(-1, 0)$;
- it rotates the configuration by $\pm 90^o$ (depending on the rule).



A **Turmite** is a two-dimensonal Turing machine on the alphabet $\mathcal{A} = \mathbb{Z}/n\mathbb{Z}$ defined by a rule $\mathcal{A} \to \{R, L\}$.

At each step:

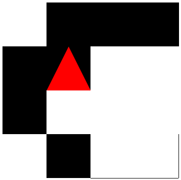
- it increases x_0 by one;
- it shifts the configuration by $(-1, 0)$;
- it rotates the configuration by $\pm 90^o$ (depending on the rule).



A **Turmite** is a two-dimensonal Turing machine on the alphabet $\mathcal{A} = \mathbb{Z}/n\mathbb{Z}$ defined by a rule $\mathcal{A} \to \{R, L\}$.

At each step:

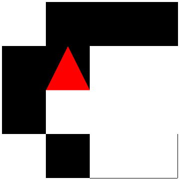
- it increases x_0 by one;
- it shifts the configuration by $(-1, 0)$;
- it rotates the configuration by $\pm 90^o$ (depending on the rule).



A **Turmite** is a two-dimensonal Turing machine on the alphabet $\mathcal{A} = \mathbb{Z}/n\mathbb{Z}$ defined by a rule $\mathcal{A} \to \{R, L\}$.

At each step:

- it increases x_0 by one;
- it shifts the configuration by $(-1, 0)$;
- it rotates the configuration by $\pm 90^o$ (depending on the rule).



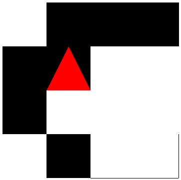
Theorem (Maldonado, Gajardo, H., Moreira 18)

Bounded-time point-to-set reachability on almost periodic or finite points for Turmites is **P**-complete (except for some trivial rules).

A **Turmite** is a two-dimensonal Turing machine on the alphabet $\mathcal{A} = \mathbb{Z}/n\mathbb{Z}$ defined by a rule $\mathcal{A} \to \{R, L\}$.

At each step:

- it increases x_0 by one;
- it shifts the configuration by $(-1, 0)$;
- it rotates the configuration by $\pm 90^o$ (depending on the rule).



Theorem (Maldonado, Gajardo, H., Moreira 18)

Bounded-time point-to-set reachability on almost periodic or finite points for Turmites is **P**-complete (except for some trivial rules).

Conjecture (Langton)

For the rule RL (at least), the unbounded-time reachability problem on finite points is decidable.

Definition?

The points that can be reached as the **single limit** of $TM^t(x)$ for some Turing machine TM and some finite point x are the Δ^0_2-computable points.

Definition?

The points that can be reached as the **single limit** of $TM^t(x)$ for some Turing machine TM and some finite point x are the Δ_2^0-computable points.

Proposition

The points that can be reached as the **single limit** of $F^t(x)$ for some cellular automaton F and some finite point x are the Δ_2^0-computable points.

Definition?

The sets that can be reached as **sets of limit points** of $TM^t(x)$ for some Turing machine TM and some finite point x are the Σ_2^0-computable sets.

Proposition

The sets that can be reached as **sets of limit points** of $F^t(x)$ for some cellular automaton F and some finite point x are the Σ_2^0-computable sets.

Definition?

The sets that can be reached as **sets of limit points** of $TM^t(x)$ for some Turing machine TM and some **computable** point x are the Σ_2^0-computable sets.

Proposition

The sets that can be reached as **sets of limit points** of $F^t(x)$ for some cellular automaton F and some **computable** point x are the Σ_2^0-computable sets.

Definition?

The sets that can be reached as **sets of limit points** of $TM^t(x)$ for some Turing machine *TM* and some **computable** point x are the Σ^0_2-computable sets.

Proposition

The sets that can be reached as **sets of limit points** of $F^t(x)$ for some cellular automaton F and some **computable** point x are the Σ^0_2-computable sets.

Corollary (Rice-style theorem)

Input: A finite point x

Output: Does $F^t(x)$ has a limit point that satisfy property $\mathcal{P}$?

is **undecidable** for any property $\mathcal{P}$.

Definition?

The probability measures that can be reached as the single limit of $\mu \circ TM^{-t}$ for some Turing machine *TM* and initial uniform Bernoulli measure μ are the Δ_2^0-computable measures.

Definition?

The probability measures that can be reached as the single limit of $\mu \circ TM^{-t}$ for some Turing machine *TM* and initial uniform Bernoulli measure μ are the Δ_2^0-computable measures.

Theorem (H., Sablik 14; Delacourt, H. 16)

The probability measures that can be reached as the single limit of $\mu \circ F^{-t}$ for some cellular automaton *F* and initial uniform Bernoulli measure μ are the σ**-invariant** Δ_2^0-computable measures.

(+ set version and Rice-style theorem)

The **topological entropy** of a Turing machine is defined as

$$\lim_{n\to\infty} \frac{\log \#\mathcal{L}_n}{n},$$

where $\mathcal{L}_n$ is the set of n-letter words appearing in its trace.

Cautionary tale III

The **topological entropy** of a Turing machine is defined as

$$\lim_{n\to\infty} \frac{\log \# \mathcal{L}_n}{n},$$

where $\mathcal{L}_n$ is the set of n-letter words appearing in its trace.

Theorem (Delvenne, Blondel 04)

Approximating the entropy of Turing machines:

Input: A Turing machine TM and a precision $n \in \mathbb{N}$

Output: A 2^{-n}-approximation of the entropy of TM.

is an **uncomputable** problem.

The **topological entropy** of a Turing machine is defined as

$$\lim_{n\to\infty} \frac{\log \#\mathcal{L}_n}{n},$$

where $\mathcal{L}_n$ is the set of n-letter words appearing in its trace.

Theorem (Delvenne, Blondel 04)

Approximating the entropy of Turing machines:

Input A Turing machine *TM* and a precision $n \in \mathbb{N}$

Output A 2^{-n}-approximation of the entropy of *TM*.

is an **uncomputable** problem.

Theorem (Jeandel 13)

The entropy of **one-tape** Turing machines is computable.

- $\sim$50% of this talk due to:

J. C. Delvenne, "What is a universal computing machine?"
(Applied Mathematics and Computation, 2009)

- $\sim$50% of this talk due to:

  J. C. Delvenne, “What is a universal computing machine?”
  (Applied Mathematics and Computation, 2009)

- “Turing-universal system” is ambiguous, but it’s getting clearer
  (at least for me, hopefully for you)

Conclusion

- $\sim$50% of this talk due to:

  J. C. Delvenne, "What is a universal computing machine?"
  (Applied Mathematics and Computation, 2009)

- "Turing-universal system" is ambiguous, but it's getting clearer
  (at least for me, hopefully for you)

- A better point of view could be:

  I've embedded (some kind of) universal computation, what can I prove?

Conclusion

- $\sim$50% of this talk due to:

  J. C. Delvenne, “What is a universal computing machine?”
  (Applied Mathematics and Computation, 2009)

- “Turing-universal system” is ambiguous, but it’s getting clearer
  (at least for me, hopefully for you)

- A better point of view could be:

  I’ve embedded (some kind of) universal computation, what can I prove?

- Many topics left uncovered: limit set / attractor point of view, dynamical properties, robustness to noise, other universalities. . .

Questions and remarks are appreciated!