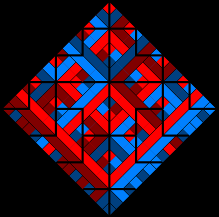
# Autour de deux propriétés dynamiques simples indécidables dans les automates cellulaires

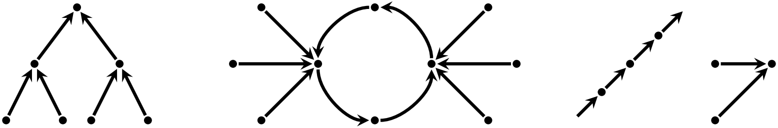
Nicolas Ollinger (LIF, Aix-Marseille Université, CNRS)

GdT automates — **LIAFA**, 6 fév. 2009



# Discrete dynamical systems

**Definition** A **DDS** is a pair $(X, F)$ where $X$ is a **topological space** and $F : X \to X$ is a **continuous** map.



The **orbit** of $x \in X$ is the sequence $(F^n(x))$ obtained by iterating $F$.

In this talk, $X = S^{\mathbb{Z}}$ where $S$ is a finite alphabet and $X$ is endowed with the **Cantor topology** (product of the discrete topology on $S$), and $F$ is a continuous map **commuting with the shift map** $\sigma$: $F \circ \sigma = \sigma \circ F$ where $\sigma(x)(z) = x(z + 1)$.

# Two dynamical properties

We consider two simple dynamical properties (as opposed to more computational properties like reachability questions).

**Definition** A DDS $(X, F)$ is **periodic** if for all $x \in X$ there exists $n \in \mathbb{N}$ such that $F^n(x) = x$.

**Definition** A DDS $(X, F)$ is **nilpotent** if there exists $0 \in X$ such that for all $x \in X$ there exists $n \in \mathbb{N}$ such that $F^n(x) = 0$.

**Question** With a **proper recursive encoding** of the DDS, can we decide given a DDS if it is periodic? if it is nilpotent?

# 1. cellular automata

# Cellular automata

**Definition** A **CA** is a triple $(S, r, f)$ where $S$ is a **finite set of states**, $r \in \mathbb{N}$ is the **radius** and $f : S^{2r+1} \to S$ is the **local rule** of the cellular automaton.

A **configuration** $c \in S^{\mathbb{Z}}$ is a coloring of $\mathbb{Z}$ by $S$.

The **global map** $F : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ applies $f$ uniformly and locally:

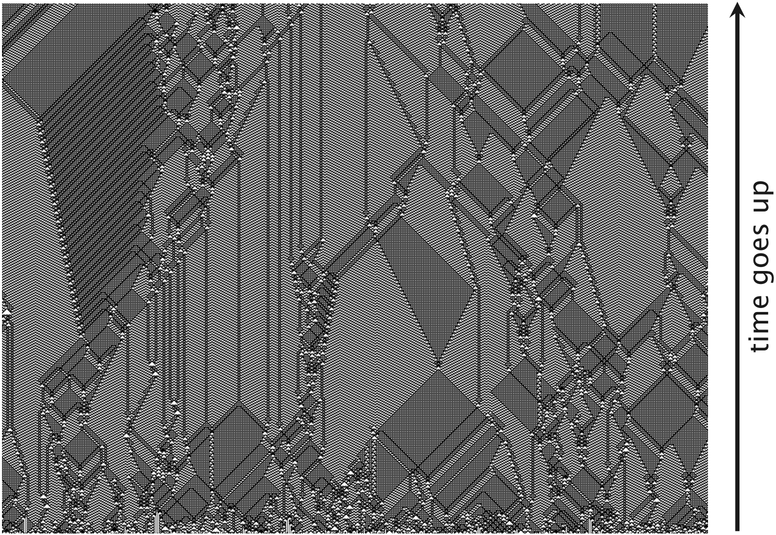
$$\forall c \in S^{\mathbb{Z}}, \forall z \in \mathbb{Z}, \qquad F(c)(z) = f(c(z-r), \ldots, c(z+r)).$$

A **space-time diagram** $\Delta \in S^{\mathbb{N}\times\mathbb{Z}}$ satisfies, for all $t \in \mathbb{Z}^+$,

$$\Delta(t+1) = F(\Delta(t)).$$

The associated DDS is $(S^{\mathbb{Z}}, F)$.

# Space-time diagram




$S = \{0, 1, 2\}$, $r = 1$, $f(x, y, z) = \lfloor 6430564760289/3^{9x+3y+z} \rfloor \pmod 3$

# König's lemma

**König's lemma** Every infinite tree with finite branching admits an infinite path.

For all $n \in \mathbb{N}$ and $u \in S^{2n+1}$, the **cylinder** $[u] \subseteq S^{\mathbb{Z}}$ is

$$[u] = \left\{ c \in S^{\mathbb{Z}} \,\middle|\, \forall i \in [-n, n]\; c(i) = u_{i+n} \right\} \quad .$$

For all $C \subseteq S^{\mathbb{Z}}$, the **König tree** $\mathcal{A}_C$ is the tree of cylinders intersecting $C$ ordered by inclusion.

The **topping** $\overline{\mathcal{A}_C} \subseteq S^{\mathbb{Z}}$ of a König tree is the set of configurations tagging an infinite path from the root (intersection of the cylinders on the path).

**Definition** The **König topology** over $S^{\mathbb{Z}}$ is the topology whose close sets are the toppings of König trees.

# Curtis-Hedlund-Lyndon's theorem

König and Cantor topologies coincide: their open sets are unions of cylinders. Compacity arguments have combinatorial counterparts.

The **clopen sets** are finite unions of cylinders.

Therefore in this topology **continuity** means **locality**.

**Theorem [Hedlund 1969]** The continuous maps commuting with the shift coincide with the global maps of cellular automata.

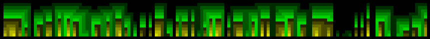
Cellular automata have a dual nature : topological maps with finite automata description.

# Nilpotency

A CA is nilpotent iff there exists a **uniform bound** $n \in \mathbb{Z}^+$ such that $F^n$ is a constant map.

**Hint** Take the bound of a **universal configuration** containing all words on $S$.

**The Nilpotency Probem (NP)** given a CA decide if it is nilpotent.

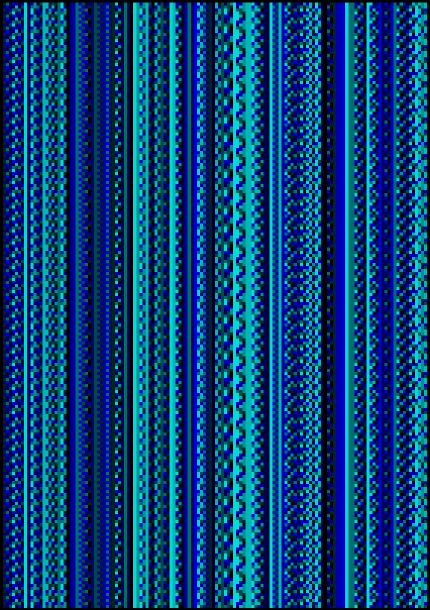


# Periodicity

A CA is periodic iff there exists a **uniform period** $n \in \mathbb{Z}^+$ such that $F^n$ is the identity map.

**Hint** Take the period of a **universal configuration** containing all words on $S$.

**The Periodicity Probem (PP)** given a CA decide if it is periodic.



# Undecidability of dynamical properties

Both **NP** and **PP** are **recursively undecidable**.

Undecidability is not necessarily a negative result:

it is a **hint of complexity**.

There exists non trival nilpotent and periodic CA with a very large bound for quite simple CA (the bound grows faster than any recursive function).

To prove these results we inject computation into dynamics.

A direct reduction of the halting problem of Turing machines does not work.

# Back to the nilpotency problem

The **limit set** $\Lambda_F = \bigcap_{n \in \mathbb{N}} F^n(S^{\mathbb{Z}})$ of a CA $F$ is the non-empty subshift of configurations appearing in biinfinite space-time diagrams $\Delta \in S^{\mathbb{Z} \times \mathbb{Z}}$ such that $\forall t \in \mathbb{Z}, \Delta(t+1) = F(\Delta(t))$.

A CA is nilpotent iff its limit set is a **singleton**.

A state $\perp \in S$ is **spreading** if $f(N) = \perp$ when $\perp \in N$.

A CA with a spreading state $\perp$ is not nilpotent iff it admits a biinfinite space-time diagram without $\perp$.

**A tiling problem** Find a coloring $\Delta \in (S \setminus \{\perp\})^{\mathbb{Z}^2}$ satisfying the tiling constraints given by $f$.

# Undecidability of the nilpotency problem

A classical undecidability result concerning tilings is the undecidability of the **domino problem** (**DP**).

**Theorem [Berger 1964]** **DP** is undecidable.

Here we need a restriction on the set of tilings.

**Theorem [Kari 1992]** **NW-deterministic DP** is undecidable.

NW-deterministic **DP** reduces to **NP** for spreading CA.

**Theorem [Kari 1992]** **NP** is undecidable.

# Back to the periodicity problem

A periodic CA is **reversible**, which for CA is the same as **bijective** and even **injective**.

One can reduce the **periodicity problem** of **complete reversible Turing machines** to **PP**.

**Immortality** is the property of having at least one non-halting orbit.

One can reduce the **immortality problem** of reversible Turing machines without periodic orbit to the periodicity problem of complete reversible Turing machines.

# Undecidability of the periodicity problem

A classical undecidability result concerning Turing machines is the **immortality problem** (**IP**).

**Theorem [Hooper 1966]** **IP** is undecidable.

Here we need a restriction to reversible machines.

**Theorem [Kari O 2008]** **Reversible IP** is undecidable.

Reversible **IP** reduces to **PP**.

**Theorem [Kari O 2008]** **PP** is undecidable.

# Revisiting classical results

For both **NP** and **PP**, we need a **stronger version** of a classical result, essentially a restriction on inputs.

The difficult part of the proofs hides into this task.

The **main difficulty** is to understand the dusty proofs.

Hopefully, we tend to reuse this for other variants.

Now, we will discuss the main ingredients.

# 2. Domino Problem (CiE 2008)

# Entscheidungsproblem: the $\forall\exists\forall$ case

**Hilbert's Entscheidungsproblem (semantic version)** To find a method which for every sentence of elementary quantification theory yields a decision as to whether or not the sentence is satisfiable.

In the 60s, the **classical decision problem** is studied with respect to classes of quantification types.

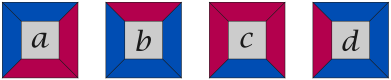
One big open class: the $\forall\exists\forall$ class. Wang and Büchi introduce in 1961 two decision problems in order to solve it.

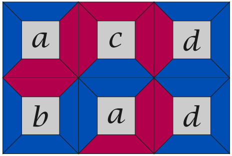
The problem is proved undecidable in 1962 by Kahr, Moore and Wang using a simpler reduction.

# The Domino Problem (DP)

*"Assume we are **given a finite set of square plates** of the same size with **edges colored**, each in a different manner. Suppose further there are infinitely many copies of each plate (plate type). We are **not permitted to rotate or reflect a plate**. The question is to find an effective procedure by which we can **decide**, for each given finite set of plates, **whether we can cover up the whole plane** (or, equivalently, an infinite quadrant thereof) **with copies of the plates subject to the restriction that adjoining edges must have the same color**."*

*(Wang, 1961)*



# Aperiodicity in DP

The set of tilings of a tile set $T$ is a compact subset of $T^{\mathbb{Z}^2}$.

By compacity, if a tile set does not tile the plane, there exists a square of size $n \times n$ that cannot be tiled.

Tile sets without tilings are recursively enumerable.

A set of Wang tiles with a periodic tiling admits a biperiodic tiling.

Tile sets with a biperiodic tiling are recursively enumerable.

Undecidability is to be found in **aperiodic tile sets**, tile sets that only admit aperiodic tilings.

# Undecidability of DP: a short history

**1964** Berger proves the undecidability of **DP**.

Two main type of related activities in the literature:

(1) construct aperiodic tile sets (small ones);
(2) give a full proof of the undecidability of **DP** (implies (1)).

From 104 tiles (Berger, 1964) to 13 tiles (Čulik, 1996) aperiodic sets.

Seminal self-similarity based proofs *(reduction from **HP**)*:

- Berger, 1964 *(20426 tiles, a full PhD thesis)*
- Robinson, 1971 *(56 tiles, 17 pages, long case analysis)*
- Durand et al, 2007 *(Kleene's fixpoint existence argument)*

Tiling rows seen as transducer trace based proof:
Kari, 2007 *(affine maps, short concise proof, reduces **IP**)*

# In this talk

A new self-similarity based construction building on classical proof schemes with concise arguments and few tiles:

1. two-by-two substitution systems and aperiodicity
2. an aperiodic tile set of 104 tiles
3. enforcing any substitution and reduction from **HP** *(sketch)*

This work combines tools and ideas from:

**[Berger 64]** *The Undecidability of the Domino Problem*

**[Robinson 71]** *Undecidability and nonperiodicity for tilings of the plane*

**[Grünbaum Shephard 89]** *Tilings and Patterns, an introduction*

**[Durand Levin Shen 05]** *Local rules and global order, or aperiodic tilings*

# Two-by-two substitution systems

A 2×2 substitution system maps a finite alphabet to 2×2 squares of letters on that alphabet.

$$s : \Sigma \to \Sigma^{\boxplus}$$

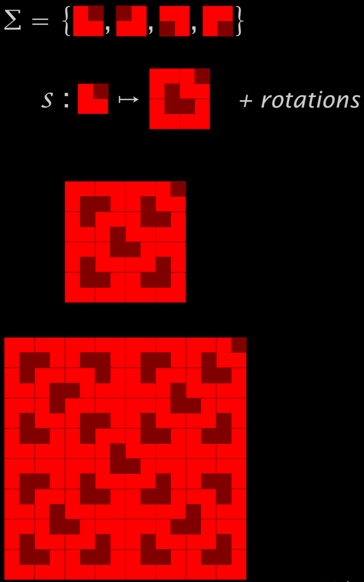
The substitution is iterated to generate bigger squares.

$$S : \Sigma^{\mathcal{P}} \to \Sigma^{\square(\mathcal{P})}$$

$\forall z \in \mathcal{P}, \forall c \in \boxplus,$

$$S(C)(2z + c) = s(C(z))(c)$$

$$S(u \cdot C) = 2u \cdot S(C)$$



# Coloring the whole plane *via* limit sets

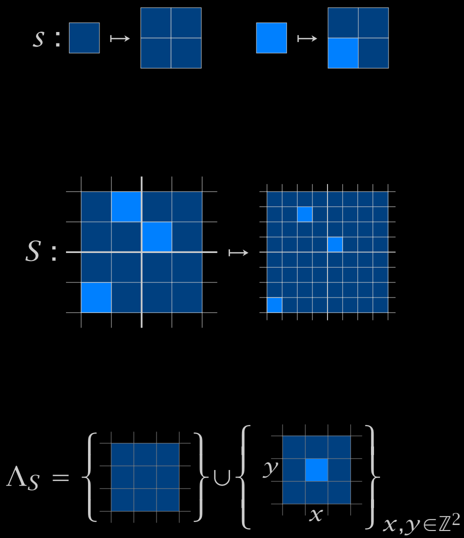
What is a coloring of the plane generated by a substitution?

With tilings in mind the set of colorings should be closed by translation and compact.

We take the limit set of iterations of the (continuous) global map closed up to translations.

$\Lambda_S = \bigcap_n \Lambda_S^n$ where $\Lambda_S^0 = \Sigma^{\mathbb{Z}^2}$

$\Lambda_S^{n+1} = \left\{ u \cdot S(C) \,\middle|\, C \in \Lambda_S^n, u \in \boxplus \right\}$



# Unambiguous substitutions are aperiodic

A substitution is **aperiodic** if its limit set $\Lambda_S$ is aperiodic.

A substitution is **unambiguous** if, for every coloring $C$ from its limit set $\Lambda_S$, there exists a **unique** coloring $C'$ and a **unique** translation $u \in \boxplus$ satisfying $C = u \cdot S(C')$.

**Proposition 3.** Unambiguity implies aperiodicity.

**Sketch of the proof.** Consider a periodic coloring with minimal period $p$, its preimage has period $p/2$. $\Diamond$

**Idea.** Construct a tile set whose tilings are in the limit set of an unambiguous substitution system.

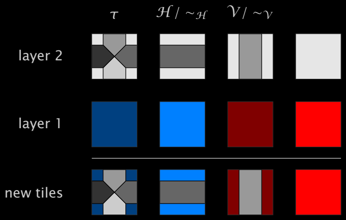
# Coding tile sets into tile sets

A tile set $\tau$ is a triple $(T, \mathcal{H}, \mathcal{V})$ where $\mathcal{H}$ and $\mathcal{V}$ define horizontal and vertical matching constraints.

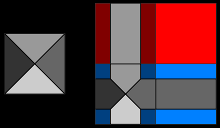
The set of tilings of $\tau$ is $X_\tau$.

A tile set $(T', \mathcal{H}', \mathcal{V}')$ **codes** a tile set $(T, \mathcal{H}, \mathcal{V})$, according to a **coding rule** $t : T \to T'^{\boxplus}$ if $t$ is injective and

$$X_{\tau'} = \{u \cdot t(C) | C \in X_\tau, u \in \boxplus\}.$$




*coding tile set*



*coding rule*

# Aperiodicity *via* unambiguous self-coding

A tile set $(T, \mathcal{H}, \mathcal{V})$ **codes** a substitution $s : T \to T^{\boxplus}$ if it codes **itself** according to the coding rule $s$.

**Proposition 4.** A tile set both admitting a tiling and coding an unambiguous substitution is aperiodic.

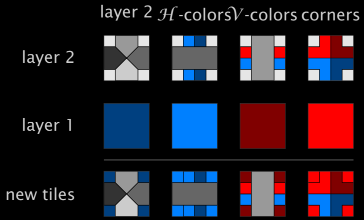
**Sketch of the proof.** $X_\tau \subseteq \Lambda_S$ and $X_\tau \neq \emptyset$. $\Diamond$

# A coding scheme with fixpoint?

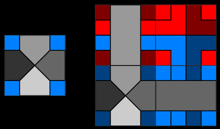
Better scheme: not strictly increasing the number of tiles.

**Problem.** it cannot encode any layered tile set, constraints between layer 1 and layer 2 are checked edge by edge.

**Solution.** add a third layer with one bit of information per edge.




*coding tile set*



*coding rule*

# Canonical substitution

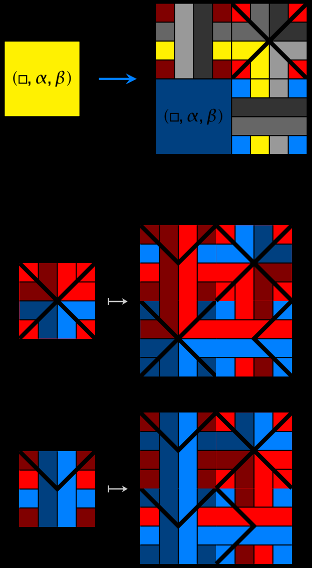
Copy the tile in the SW corner but for layer 1.

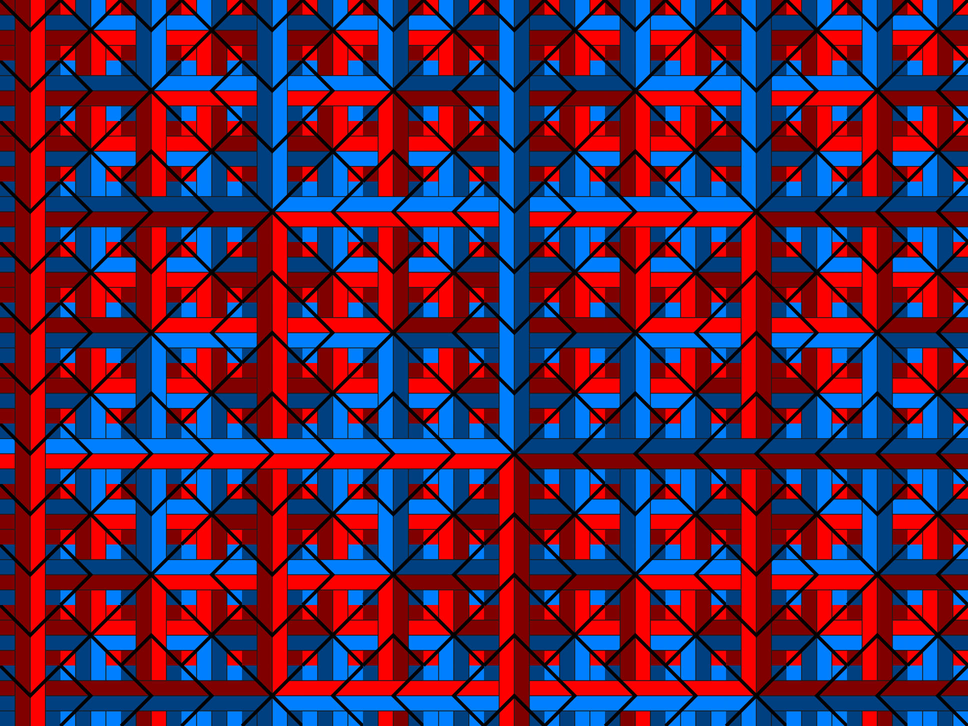
Put the only possible X in NE that carry layer 1 of the original tile on SW wire.

Propagate wires colors.

Let H et V tile propagate layer 3 arrows.

The substitution is injective.





# Aperiodicity: sketch of the proof

**1. The tile set admits a tiling:**
*Generate a valid tiling by iterating the substitution rule:*
$X_\tau \cap \Lambda_S \neq \emptyset$.

**2. The substitution is unambiguous:**
*It is injective and the projectors have disjoined images.*

**3. The tile set codes the substitution:**

(a) each tiling is an image of the canonical substitution
*Consider any tiling, level by level, short case analysis.*

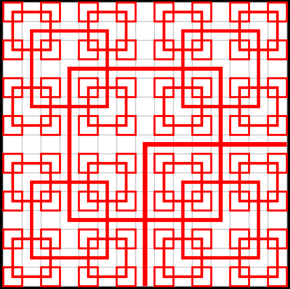
(b) the preimage of a tiling is a tiling
*Straightforward by construction (preimage remove constraints).*

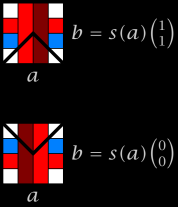
# Enforcing substitutions *via* tilings

Let $\pi$ map every tile of $\tau(s')$ to $s'(a)(u)$ where $a$ and $u$ are the letter and the value of ⊞ on layer 1.

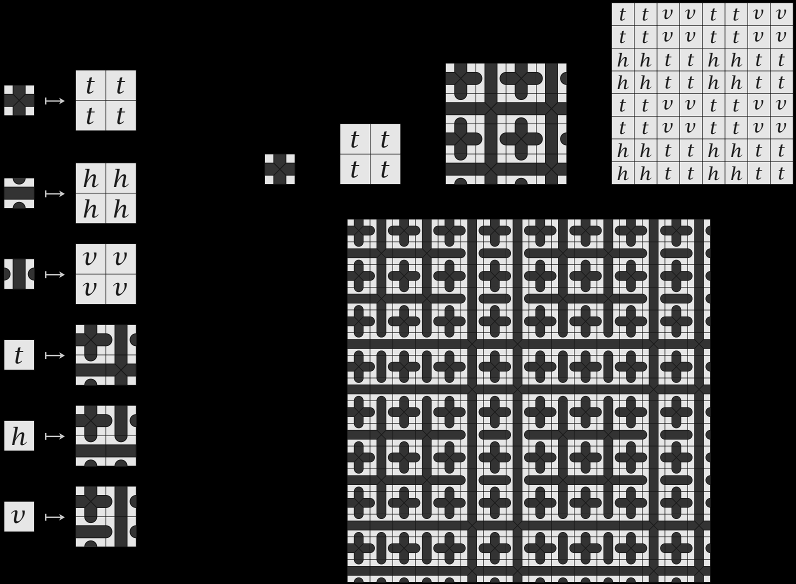
**Theorem 2.** Let $s'$ be any substitution system. The tile set $\tau(s')$ enforces $s'$: $\pi\left(X_{\tau(s')}\right) = \Lambda_{S'}$.

**Idea.** Every tiling of $\tau(s')$ codes an history of $S'$ and every history of $S'$ can be encoded into a tiling of $\tau(s')$.





# Infinitely many squares of unbounded size



# Reducing HP to DP

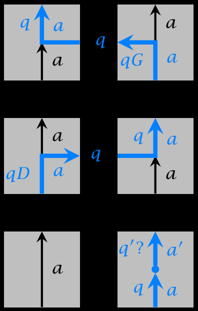
Any tiling by previous tile set contains infinitely many finite squares of unbounded size.

In each square, simulate the computation of the given Turing machine from an empty tape.

Initial computation is enforced in the SW corner.

Remove the halting state.

The tile set tiles the plan iff the Turing machine does not halt.



# 3. Immortality Problem (MFCS 2008)

# The Immortality Problem (IP)

*"($T_2$) To find an effective method, which for every Turing-machine $M$ decides whether or not, for all tapes $I$ (finite and infinite) and all states $B$, $M$ will eventually halt if started in state $B$ on tape $I$"* *(Büchi, 1962)*

A **TM** is a triple $(S, \Sigma, T)$ where $S$ is a finite set of states, $\Sigma$ a finite alphabet and $T \subseteq (S \times \{\leftarrow, \rightarrow\} \times S) \cup (S \times \Sigma \times S \times \Sigma)$ is a set of instructions.

$(s, \delta, t)$ : *"in state $s$ move according to $\delta$ and enter state $t$."*
$(s, a, t, b)$ : *"in state $s$, reading letter $a$, write letter $b$ and enter state $t$."*

Partial DDS $(S \times \Sigma^{\mathbb{Z}}, G)$ where $G$ is a partial continuous map.

A TM is **mortal** if all configurations are ultimately halting.

# Aperiodicity in IP

As $S \times \Sigma^{\mathbb{Z}}$ is compact, $G$ is continuous and the set of halting configurations is open, **mortality** implies **uniform mortality**.

Mortal TM are recursively enumerable.

TM with a periodic orbit are recursively enumerable.

Undecidability is to be found in **aperiodic TM**, TM whose infinite orbits are all aperiodic.

# In this talk

We investigate the **(un)decidability** of **dynamical properties** of three models of **reversible** computation.

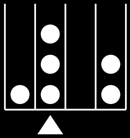
We consider the behavior of the models starting from **arbitrary initial configurations**.

**Immortality** is the property of having at least one non-halting orbit.

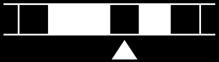
**Periodicity** is the property of always eventually returning back to the starting configuration.

# Models of reversible computation

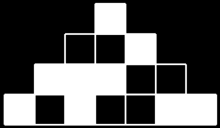
Counter Machines (**CM**)



Turing Machines (**TM**)



Cellular Automata (**CA**)



A machine is **deterministic** if there exists at most one transition from each configuration.

A machine is **reversible** if there exists **another machine** that can inverse **each step** of computation.

# The periodicity problem (PP)

$\mathcal{S}$ is **complete** if $F$ is total.

A configuration $x$ is **$n$-periodic** if $F^n(x) = x$.

$\mathcal{S}$ is **periodic** if all its configurations are periodic.

$\mathcal{S}$ is **uniformly periodic** if a uniform bound $n$ exists such that $F^n$ is the identity map.

**Periodicity Problem** Given $\mathcal{S} \in \mathcal{M}$, is $\mathcal{S}$ periodic?

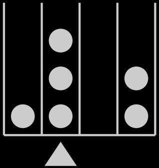
When $X$ is compact and the set of $n$-periodic configurations is open, uniform periodicity is the same as periodicity.

# Results

| | HP | IP | PP |
|---|---|---|---|
| **RCA** | | [KL08] | Thm 12 |
| **RTM** | [Lecerf63] | Thm 7 | Thm 8 |
| **RCM** | [Morita96] | Thm 1 | Thm 3 |

⟶ denotes many-one reductions.

# Reversible Counter Machines



A $k$-**CM** is a triple $(S, k, T)$ where $S$ is a finite set of states and $T \subseteq S \times \{0, +\}^k \times \mathbb{Z}_k \times \{-, 0, +\} \times S$ is a set of instructions.

$(s, u, i, \phi, t) \in T$ : *"in state $s$ with counter values $u$,*
*apply $\phi$ to counter $i$ and enter to state $t$."*

DDS $(S \times \mathbb{N}^k, G)$ where $G(\mathfrak{c})$ is the unique $\mathfrak{c}'$ such that $\mathfrak{c} \vdash \mathfrak{c}'$.

# Immortality

**[Minsky67]** Every recursive function is computed by a 2-DCM and thus **HP** is undecidable for 2-DCM.

# Immortality

**[Minsky67]** Every recursive function is computed by a $2$-DCM and thus **HP** is undecidable for $2$-DCM.

**[Hooper66] IP** is undecidable for $2$-DCM.
***Idea for new proof*** *Enforce infinite orbits to go through unbounded initial segments of an orbit from $x_0$ to reduce **HP**.* $\Diamond$

# Immortality

**[Minsky67]** Every recursive function is computed by a $2$-DCM and thus **HP** is undecidable for $2$-DCM.

**[Hooper66]** **IP** is undecidable for $2$-DCM.
***Idea for new proof*** *Enforce infinite orbits to go through unbounded initial segments of an orbit from $x_0$ to reduce **HP**.* ◊

**[Morita96]** Every $k$-DCM is **simulated** by a $2$-RCM.
***Idea*** *Encode a stack with two counters to keep an history of simulated instructions.* ◊

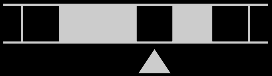
# Immortality

**[Minsky67]** Every recursive function is computed by a $2$-DCM and thus **HP** is undecidable for $2$-DCM.

**[Hooper66] IP** is undecidable for $2$-DCM.
***Idea for new proof*** *Enforce infinite orbits to go through unbounded initial segments of an orbit from $x_0$ to reduce **HP**.* ◊

**[Morita96]** Every $k$-DCM is **simulated** by a $2$-RCM.
***Idea*** *Encode a stack with two counters to keep an history of simulated instructions.* ◊

**Theorem 1** **IP** is undecidable for $2$-RCM.
***Idea*** *Morita's simulation preserves immortality.* ◊



# Reversible Turing Machines

A **TM** is a triple $(S, \Sigma, T)$ where $S$ is a finite set of states, $\Sigma$ a finite alphabet and $T \subseteq (S \times \{\leftarrow, \rightarrow\} \times S) \cup (S \times \Sigma \times S \times \Sigma)$ is a set of instructions.

$(s, \delta, t)$ : *"in state $s$ move according to $\delta$ and enter state $t$."*
$(s, a, t, b)$ : *"in state $s$, reading letter $a$, write letter $b$ and enter state $t$."*

DDS $(S \times \Sigma^{\mathbb{Z}}, G)$ where $G(\mathfrak{c})$ is the unique $\mathfrak{c}'$ such that $\mathfrak{c} \vdash \mathfrak{c}'$.

# Immortality: a first attempt

"($T_2$) *To find an effective method, which for every Turing-machine* $M$ *decides whether or not, for all tapes* $I$ *(finite and infinite) and all states* $B$, $M$ *will eventually halt if started in state* $B$ *on tape* $I$" *(Büchi, 1962)*

# Immortality: a first attempt

*"$(T_2)$ To find an effective method, which for every Turing-machine $M$ decides whether or not, for all tapes $I$ (finite and infinite) and all states $B$, $M$ will eventually halt if started in state $B$ on tape $I$"* *(Büchi, 1962)*

**[Hooper66] IP** is undecidable for DTM.

***Idea*** *TM with recursive calls! (we will discuss this)* ◊

# Immortality: a first attempt

*“($T_2$) To find an effective method, which for every Turing-machine $M$ decides whether or not, for all tapes $I$ (finite and infinite) and all states $B$, $M$ will eventually halt if started in state $B$ on tape $I$”* *(Büchi, 1962)*

**[Hooper66] IP** is undecidable for DTM.

***Idea*** *TM with recursive calls! (we will discuss this)* ◊

**[Lecerf63]** Every DTM is **simulated** by a RTM.

***Idea*** *Keep history on a stack encoded on the tape.* ◊

# Immortality: a first attempt

*"($T_2$) To find an effective method, which for every Turing-machine $M$ decides whether or not, for all tapes $I$ (finite and infinite) and all states $B$, $M$ will eventually halt if started in state $B$ on tape $I$"* *(Büchi, 1962)*

**[Hooper66] IP** is undecidable for DTM.
***Idea*** *TM with recursive calls! (we will discuss this)* ◊

**[Lecerf63]** Every DTM is **simulated** by a RTM.
***Idea*** *Keep history on a stack encoded on the tape.* ◊

**Problem** The simulation **does not** preserve immortality due to **unbounded searches**. We need to rewrite Hooper's proof for reversible machines.

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.

***Idea*** *by compacity, extract infinite failure sequence*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

$\underset{s}{\underline{@}}$1111111111111x2222y      *search x →*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

$@\underline{1}11111111111x2222y$ $\overline{s}'_1$ *bounded search 1*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

@1$\underline{1}$11111111111x2222y    *bounded search 2*
$s_2'$

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

$@11\underline{1}111111111x2222y$ *bounded search 3*
$\overline{s}'_3$

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

@$\underset{s_0}{\underline{@}}_s$**xy**1111111111x2222y *recursive call*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

@$\mathbf{@_s 11111x22222\underline{y}}_{\mathbf{s_c}}$x2222y *ultimately in case of collision...*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

@$\underset{\mathbf{s_b}}{\underline{@}}$$_{\mathbf{s}}$**xy**1111111111x2222y *...revert to clean*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

$@111\underset{s_1'}{\underline{1}}11111111x2222y$ *pop and continue bounded search 1*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

$@1111\underline{1}11111111x2222y$ *bounded search 2*
$\overline{s}'_2$

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

@11111$\underline{1}$1111111x2222y *bounded search 3*
$\overline{s}'_3$

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

$@111\underset{s_0}{\underline{@}}_s xy1111111x2222y$ *recursive call*

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for 2-RCM $(s, \underline{@}1^m x 2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

$@111\underline{@}_{s}\mathbf{xy}1111111x2222y$ *recursive call*
$s_0$

# Immortality: simulating RCM

**Theorem 7** **IP** is undecidable for RTM.

**Reduction** reduce **HP** for $2$-RCM $(s, \underline{@}1^m x2^n y)$

**Problem** unbounded searches produce immortality.
***Idea*** *by compacity, extract infinite failure sequence*

**Hooper's trick** use bounded searches with **recursive calls** to initial segments of the simulation of increasing sizes:

@111$\underline{@}_s$**xy**1111111x2222y *recursive call*
$\mathbf{s_0}$

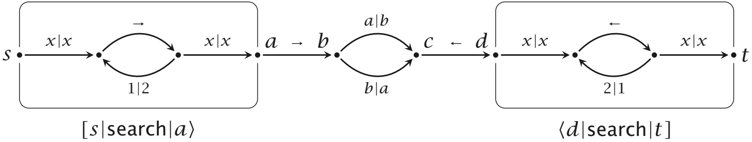
The RTM is immortal iff the $2$-RCM is mortal on $(s_0, (0, 0))$.

# Programming tips and tricks (1/2)

We designed a TM programming language called Gnirut:

`http://www.lif.univ-mrs.fr/~nollinge/rec/gnirut/`

**First ingredient** use **macros** to avoid repetitions:

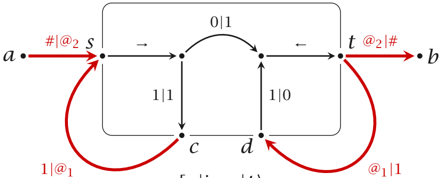



```
def [s|search|t⟩ :
   s. x ⊢ x, u
   u.  →, r
   r. 1 ⊢ 2, u | x ⊢ x, t

[s|search|a⟩
a.  →, b
b. a ⊢ b, c | b ⊢ a, c
c.  ←, d
⟨d|search|t]
```

# Programming tips and tricks (2/2)

**Second ingredient** use **recursive calls**:




$[s|\mathrm{incr}|t\rangle$

```
fun [s|incr|t⟩ :
  s.  →, r
  r. 0 ⊢ 1, b | 1 ⊢ 1, c
  call [c|incr|d⟩ from 1   ⇐ call 1
  d. 1 ⊢ 0, b
  b.  ←, t

call [a|incr|b⟩ from #   ⇐ call 2
```

# Immortality: skeleton

$[s|\mathrm{check}_1|t\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathrm{x} \vdash \underline{@_\alpha}1^m\mathrm{x}, t$ or $s.\,\underline{@_\alpha}1^\omega \uparrow$ or halt.

# Immortality: skeleton

$[s|\mathrm{check}_1|t\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathsf{x} \vdash \underline{@_\alpha}1^m\mathsf{x}, t$ or $s.\,\underline{@_\alpha}1^\omega \uparrow$ or halt.

$[s|\mathrm{search}_1|t_0, t_1, t_2\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathsf{x} \vdash @_\alpha 1^m\underline{\mathsf{x}}, t_{m[3]}$ or $\ldots$

# Immortality: skeleton

$[s|\text{check}_1|t\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathsf{x} \vdash \underline{@_\alpha}1^m\mathsf{x}, t$ or $s.\,\underline{@_\alpha}1^\omega \uparrow$ or halt.

$[s|\text{search}_1|t_0, t_1, t_2\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathsf{x} \vdash @_\alpha 1^m\underline{\mathsf{x}}, t_{m[3]}$ or ...

**RCM** ingredients:

| | |
|---|---|
| testing counters | $[s\,|\,\text{test1}\,|\,z, p\rangle$ and $[s\,|\,\text{test2}\,|\,z, p\rangle$ |
| increment counter | $[s|\text{inc1}|t, co\rangle$ and $[s|\text{inc2}|t, co\rangle$ |
| decrement counter | $[s|\text{dec1}|t, co\rangle$ and $[s|\text{dec2}|t, co\rangle$ |

# Immortality: skeleton

$[s|\text{check}_1|t\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathsf{x} \vdash \underline{@_\alpha}1^m\mathsf{x}, t$ or $s.\,\underline{@_\alpha}1^\omega \uparrow$ or halt.

$[s|\text{search}_1|t_0, t_1, t_2\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathsf{x} \vdash @_\alpha 1^m\underline{\mathsf{x}}, t_{m[3]}$ or …

**RCM** ingredients:

| | |
|---|---|
| testing counters | $[s|\text{test1}|z, p\rangle$ and $[s|\text{test2}|z, p\rangle$ |
| increment counter | $[s|\text{inc1}|t, co\rangle$ and $[s|\text{inc2}|t, co\rangle$ |
| decrement counter | $[s|\text{dec1}|t, co\rangle$ and $[s|\text{dec2}|t, co\rangle$ |

Simulator $[s|\text{RCM}_\alpha|co_1, co_2, \ldots\rangle$ initialize then compute

# Immortality: skeleton

$[s|\mathrm{check}_1|t\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathtt{x} \vdash \underline{@_\alpha}1^m\mathtt{x}, t$ or $s.\,\underline{@_\alpha}1^\omega \uparrow$ or halt.

$[s|\mathrm{search}_1|t_0, t_1, t_2\rangle$ satisfies $s.\,\underline{@_\alpha}1^m\mathtt{x} \vdash @_\alpha 1^m\underline{\mathtt{x}}, t_{m[3]}$ or ...

**RCM** ingredients:

| | |
|---|---|
| testing counters | $[s|\mathrm{test1}|z, p\rangle$ and $[s|\mathrm{test2}|z, p\rangle$ |
| increment counter | $[s|\mathrm{inc1}|t, co\rangle$ and $[s|\mathrm{inc2}|t, co\rangle$ |
| decrement counter | $[s|\mathrm{dec1}|t, co\rangle$ and $[s|\mathrm{dec2}|t, co\rangle$ |

Simulator $[s|\mathrm{RCM}_\alpha|co_1, co_2, \ldots\rangle$ initialize then compute

$$[s|\mathrm{check}_\alpha|t\rangle = [s|\mathrm{RCM}_\alpha|co_1, co_2, \ldots\rangle + \langle co_1, co_2, \ldots|\mathrm{RCM}_\alpha|s]$$

# Program it!

```
def [s|search_1|t_0, t_1, t_2⟩ :
  s. @_α ⊢ @_α, l
  l. →, u
  u. x ⊢ x, t_0
   | 1x ⊢ 1x, t_1
   | 11x ⊢ 11x, t_2
   | 111 ⊢ 111, c
  call [c|check_1|p⟩ from 1
  p. 111 ⊢ 111, l

def [s|search_2|t_0, t_1, t_2⟩ :
  s. x ⊢ x, l
  l. →, u
  u. y ⊢ y, t_0
   | 2y ⊢ 2y, t_1
   | 22y ⊢ 22y, t_2
   | 222 ⊢ 222, c
  call [c|check_2|p⟩ from 2
  p. 222 ⊢ 222, l

def [s|test1|z, p⟩ :
  s. @_α x ⊢ @_α x, z
   | @_α 1 ⊢ @_α 1, p

def [s|endtest2|z, p⟩ :
  s. xy ⊢ xy, z
   | x2 ⊢ x2, p

def [s|test2|z, p⟩ :
  [s|search_1|t_0, t_1, t_2⟩
  [t_0|endtest2|z_0, p_0⟩
  [t_1|endtest2|z_1, p_1⟩
  [t_2|endtest2|z_2, p_2⟩
  ⟨z_0, z_1, z_2|search_1|z]
  ⟨p_0, p_1, p_2|search_1|p]

def [s|mark_1|t, co⟩ :
  s. y1 ⊢ 2y, t
   | yx ⊢ yx, co

def [s|endinc_1|t, co⟩ :
  [s|search_2|r_0, r_1, r_2⟩
  [r_0|mark_1|t_0, co_0⟩
  [r_1|mark_1|t_1, co_1⟩
  [r_2|mark_1|t_2, co_2⟩
  ⟨t_2, t_0, t_1|search_2|t]
  ⟨co_0, co_1, co_2|search_2|co]

def [s|inc2_1|t, co⟩ :
  [s|search_1|r_0, r_1, r_2⟩
  [r_0|endinc_1|t_0, co_0⟩
  [r_1|endinc_1|t_1, co_1⟩
  [r_2|endinc_1|t_2, co_2⟩
  ⟨t_0, t_1, t_2|search_1|t]
  ⟨co_0, co_1, co_2|search_1|co]

def [s|dec2_1|t⟩ :
  ⟨s, co|inc2_1|t]

def [s|mark_2|t, co⟩ :
  s. y2 ⊢ 2y, t
   | yx ⊢ yx, co

def [s|endinc_2|t, co⟩ :
  [s|search_2|r_0, r_1, r_2⟩
  [r_0|mark_2|t_0, co_0⟩
  [r_1|mark_2|t_1, co_1⟩
  [r_2|mark_2|t_2, co_2⟩
  ⟨t_2, t_0, t_1|search_2|t]
  ⟨co_0, co_1, co_2|search_2|co]

def [s|inc2_2|t, co⟩ :
  [s|search_1|r_0, r_1, r_2⟩
  [r_0|endinc_2|t_0, co_0⟩
  [r_1|endinc_2|t_1, co_1⟩
  [r_2|endinc_2|t_2, co_2⟩
  ⟨t_0, t_1, t_2|search_1|t]
  ⟨co_0, co_1, co_2|search_1|co]

def [s|dec2_2|t⟩ :
  ⟨s, co|inc2_2|t]

def [s|pushinc_1|t, co⟩ :
  s. x2 ⊢ 1x, c
   | xy1 ⊢ 1xy, pt
   | xyx ⊢ 1yx, pco
  [c|endinc_1|pt0, pco0⟩
  pt0. →, t0
  t0. 2 ⊢ 2, pt
  pt. ←, t
  pco0. x ⊢ 2, pco
  pco. ←, zco
  zco. 1 ⊢ x, co

def [s|inc1_1|t, co⟩ :
  [s|search_1|r_0, r_1, r_2⟩
  [r_0|pushinc_1|t_0, co_0⟩
  [r_1|pushinc_1|t_1, co_1⟩
  [r_2|pushinc_1|t_2, co_2⟩
  ⟨t_2, t_0, t_1|search_1|t]
  ⟨co_0, co_1, co_2|search_1|co]

def [s|dec1_1|t⟩ :
  ⟨s, co|inc1_1|t]

def [s|pushinc_2|t, co⟩ :
  s. x2 ⊢ 1x, c
   | xy2 ⊢ 1xy, pt
   | xyy ⊢ 1yy, pco
  [c|endinc_2|pt0, pco0⟩
  pt0. →, t0
  t0. 2 ⊢ 2, pt
  pt. ←, t
  pco0. x ⊢ 2, pco
  pco. ←, zco
  zco. 1 ⊢ x, co

def [s|inc1_2|t, co⟩ :
  [s|search_1|r_0, r_1, r_2⟩
  [r_0|pushinc_2|t_0, co_0⟩
  [r_1|pushinc_2|t_1, co_1⟩
  [r_2|pushinc_2|t_2, co_2⟩
  ⟨t_2, t_0, t_1|search_1|t]
  ⟨co_0, co_1, co_2|search_1|co]

def [s|dec1_2|t⟩ :
  ⟨s, co|inc1_2|t]

def [s|init_1|r⟩ :
  s. →, u
  u. 11 ⊢ xy, e
  e. ←, r

def [s|RCM_1|co_1, co_2⟩ :
  [s|init_1|s_0⟩
  [s_0|test1|s_1z, n⟩
  [s_1|inc1_1|s_2, co_1⟩
  [s_2|inc2_1|s_3, co_2⟩
  [s_3|test1|n', s_1p⟩
  ⟨s_1z, s_1p|test1|s_1]

def [s|init_2|r⟩ :
  s. →, u
  u. 22 ⊢ xy, e
  e. ←, r

def [s|RCM_2|co_1, co_2⟩ :
  [s|init_2|s_0⟩
  [s_0|test1|s_1z, n⟩
  [s_1|inc1_2|s_2, co_1⟩
  [s_2|inc2_2|s_3, co_2⟩
  [s_3|test1|n', s_1p⟩
  ⟨s_1z, s_1p|test1|s_1]

fun [s|check_1|t⟩ :
  [s|RCM_1|co_1, co_2, ...⟩
  ⟨co_1, co_2, ...|RCM_1|t]

fun [s|check_2|t⟩ :
  [s|RCM_2|co_1, co_2, ...⟩
  ⟨co_1, co_2, ...|RCM_2|t]
```

# Periodicity

**Theorem 8** **PP** is undecidable for RTM.

**Idea** Reduce **IP** to **PP**:

(1) **IP** is still undecidable for RTM **without periodic orbit**.

# Periodicity

**Theorem 8** **PP** is undecidable for RTM.

**Idea** Reduce **IP** to **PP**:

(1) **IP** is still undecidable for RTM **without periodic orbit**.

(2) Let $\mathcal{M} = (S, \Sigma, T)$ be a RTM without periodic orbit
Let $\mathcal{M}'$ be the complete RTM with set of states $S \times \{+, -\}$ simulating $\mathcal{M}$ on $+$ and $\mathcal{M}^{-1}$ on $-$ and inversing polarity on halting states.

# Periodicity

**Theorem 8** **PP** is undecidable for RTM.

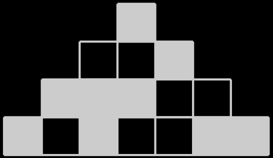
**Idea** Reduce **IP** to **PP**:

(1) **IP** is still undecidable for RTM **without periodic orbit**.

(2) Let $\mathcal{M} = (S, \Sigma, T)$ be a RTM without periodic orbit
Let $\mathcal{M}'$ be the complete RTM with set of states $S \times \{+, -\}$ simulating $\mathcal{M}$ on $+$ and $\mathcal{M}^{-1}$ on $-$ and inversing polarity on halting states.

(3) $\mathcal{M}'$ is periodic iff $\mathcal{M}$ is mortal. $\Diamond$

# Reversible Cellular Automata



A **CA** is a triple $(S, r, f)$ where $S$ is a finite set of states, $r$ the radius and $f : S^{2r+1} \to S$ the local rule.

DDS $(S^{\mathbb{Z}}, G)$ where $\forall z \in \mathbb{Z},\ G(c)(z) = f(c(z-r), \ldots, c(z+r))$

# Periodicity

**Theorem 12** **PP** is undecidable for RCA.

**Idea** Reduce **PP** for RTM to **PP** for RCA:

(1) **PP** is still undecidable for **complete** RTM.

# Periodicity

**Theorem 12** **PP** is undecidable for RCA.

**Idea** Reduce **PP** for RTM to **PP** for RCA:

(1) **PP** is still undecidable for **complete** RTM.

(2) Let $\mathcal{M} = (S, \Sigma, T)$ be a complete RTM
Let $(S', 2, f)$ be the RCA with set of states
$\Sigma \times (S \times \{+, -\} \cup \{\leftarrow, \rightarrow\})$ simulating $\mathcal{M}$ on $+$ and $\mathcal{M}^{-1}$
on $-$ on two levels.

# Periodicity

**Theorem 12** **PP** is undecidable for RCA.

**Idea** Reduce **PP** for RTM to **PP** for RCA:

(1) **PP** is still undecidable for **complete** RTM.

(2) Let $\mathcal{M} = (S, \Sigma, T)$ be a complete RTM
Let $(S', 2, f)$ be the RCA with set of states
$\Sigma \times (S \times \{+, -\} \cup \{\leftarrow, \rightarrow\})$ simulating $\mathcal{M}$ on $+$ and $\mathcal{M}^{-1}$ on $-$ on two levels.

(3) In case of local inconsistency, invert polarity.

# Periodicity

**Theorem 12** **PP** is undecidable for RCA.

**Idea** Reduce **PP** for RTM to **PP** for RCA:

(1) **PP** is still undecidable for **complete** RTM.

(2) Let $\mathcal{M} = (S, \Sigma, T)$ be a complete RTM
Let $(S', 2, f)$ be the RCA with set of states
$\Sigma \times (S \times \{+, -\} \cup \{\leftarrow, \rightarrow\})$ simulating $\mathcal{M}$ on $+$ and $\mathcal{M}^{-1}$
on $-$ on two levels.

(3) In case of local inconsistency, invert polarity.

(4) The RCA is periodic iff $\mathcal{M}$ is periodic. $\Diamond$

# Open Problems and conjectures

**CA** Consider properties from topological classifications (*e.g.* Kůrka). Is positive expansivity decidable?

**RTM** We conjecture undecidable whether a given complete **RTM** admits a periodic configuration. Prove it!

**Tilings** Provide tools to prove that a set of colorings **cannot** be recognized by tilings (up to projection, *aka* sofic subshifts).

**DP and IP** Is it possible to consider the two problems somehow *dual*? Kari reduced **IP** to **DP**. Do the inverse naturally.