Embedding computations in tilings

Andrei Romashchenko

26 September 2017, ENS de Lyon

Two central notions in this talk:

Two central notions in this talk:

- **SFT** (subshift of finite type)

Two central notions in this talk:

- **SFT** (subshift of finite type) on $\mathbb{Z}^2$ over an alphabet Σ:

Two central notions in this talk:

- **SFT** (subshift of finite type) on $\mathbb{Z}^2$ over an alphabet Σ: the set $S_{\mathcal{F}}$ of all configurations $f : \mathbb{Z}^2 \to \Sigma$ that avoid forbidden patterns from some finite family $\mathcal{F}$

Two central notions in this talk:

- **SFT** (subshift of finite type) on $\mathbb{Z}^2$ over an alphabet Σ: the set $S_{\mathcal{F}}$ of all configurations $f : \mathbb{Z}^2 \to \Sigma$ that avoid forbidden patterns from some finite family $\mathcal{F}$
- **COMPUTATION** (algorithm):

Two central notions in this talk:

- **SFT** (subshift of finite type) on $\mathbb{Z}^2$ over an alphabet Σ:
  the set $S_{\mathcal{F}}$ of all configurations $f : \mathbb{Z}^2 \to \Sigma$ that avoid forbidden patterns from some finite family $\mathcal{F}$
- **COMPUTATION** (algorithm):
  Turing machine with one or many bi-infinite tapes

Embedding computation in a subshift...

Embedding computation in a subshift... Why?

Embedding computation in a subshift... Why?

A. undecidability results:

Embedding computation in a subshift... Why?

A. undecidability results:

(i) properties of a Turing machine $\Longleftrightarrow$ properties of an SFT with an embedded computation

Embedding computation in a subshift... Why?

A. undecidability results:

(i) properties of a Turing machine $\iff$ properties of an SFT with an embedded computation

(ii) many properties of a TM are algorithmically undecidable

Embedding computation in a subshift... Why?

A. undecidability results:

(i) properties of a Turing machine $\Longleftrightarrow$ properties of an SFT with an embedded computation

(ii) many properties of a TM are algorithmically undecidable

(i) + (ii) $\Rightarrow$ properties of an SFT are algorithmically undecidable

Embedding computation in a subshift... Why?

A. undecidability results:

(i) properties of a Turing machine $\Longleftrightarrow$ properties of an SFT with an embedded computation

(ii) many properties of a TM are algorithmically undecidable

(i) + (ii) $\Rightarrow$ properties of an SFT are algorithmically undecidable

B. combinatorial/topological/dynamical properties of SFT:

Embedding computation in a subshift... Why?

A. undecidability results:

(i) properties of a Turing machine $\Longleftrightarrow$ properties of an SFT with an embedded computation

(ii) many properties of a TM are algorithmically undecidable

(i) + (ii) $\Rightarrow$ properties of an SFT are algorithmically undecidable

B. combinatorial/topological/dynamical properties of SFT:

- proofs of *positive* results require *constructions*

Embedding computation in a subshift... Why?

A. undecidability results:

(i) properties of a Turing machine $\Longleftrightarrow$ properties of an SFT with an embedded computation

(ii) many properties of a TM are algorithmically undecidable

(i) + (ii) $\Rightarrow$ properties of an SFT are algorithmically undecidable

B. combinatorial/topological/dynamical properties of SFT:

- proofs of *positive* results require *constructions*
- algorithm is a construction *par excellence*

Embedding computation in a subshift... Why?

A. undecidability results:

(i) properties of a Turing machine $\Longleftrightarrow$ properties of an SFT with an embedded computation

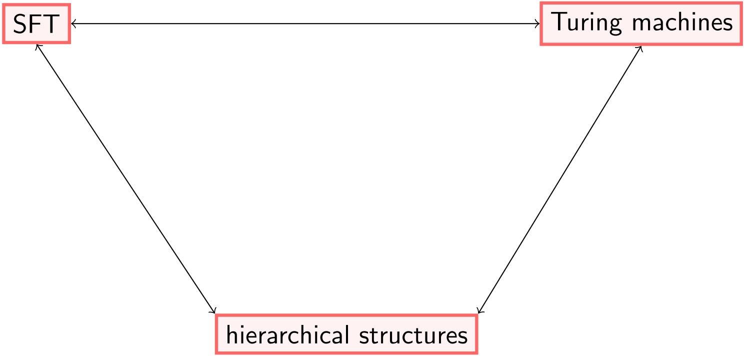
(ii) many properties of a TM are algorithmically undecidable

(i) + (ii) $\Rightarrow$ properties of an SFT are algorithmically undecidable

B. combinatorial/topological/dynamical properties of SFT:

- proofs of *positive* results require *constructions*
- algorithm is a construction *par excellence*

$$\boxed{\text{SFT}} \longleftrightarrow \boxed{\text{Turing machines}}$$


SFT
Turing machines
hierarchical structures

SFT with a hierarchical structure with an embedded computation:

SFT with a hierarchical structure with an embedded computation:

- Berger/Robinson's approach

SFT with a hierarchical structure with an embedded computation:

- Berger/Robinson's approach
- Self-simulating tilings

SFT with a hierarchical structure with an embedded computation:

- Berger/Robinson's approach
- Self-simulating tilings (goes back to self-simulating automata, P. Gács and even earlier J. von Neumann)

SFT with a hierarchical structure with an embedded computation:

- Berger/Robinson's approach
- Self-simulating tilings (goes back to self-simulating automata, P. Gács and even earlier J. von Neumann)

Special type of SFT: Wang tilings

Special type of SFT: Wang tilings

- Color: an element of a finite set $C = \{\cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\}$

Special type of SFT: Wang tilings

- Color: an element of a finite set $C = \{\cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\}$

- Wang Tile: a unit square with colored sides.

Special type of SFT: Wang tilings

- Color: an element of a finite set $C = \{\cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\}$

- Wang Tile: a unit square with colored sides.
  i.e, an element of C^4, e.g., 

Special type of SFT: Wang tilings

- Color: an element of a finite set $C = \{\cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\}$
- Wang Tile: a unit square with colored sides.
  i.e, an element of C^4, e.g., 
- Tile set: a set $\tau \subset C^4$

Special type of SFT: Wang tilings

- **Color:** an element of a finite set $C = \{\cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\}$
- **Wang Tile:** a unit square with colored sides.
  i.e, an element of C^4, e.g., 
- **Tile set:** a set $\tau \subset C^4$
- **Tiling:** a configuration $f : \mathbb{Z}^2 \to \tau$, where
  every two adjacent tiles share the same color on the common side

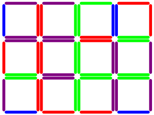
Special type of SFT: Wang tilings

- Color: an element of a finite set $C = \{\cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\}$
- Wang Tile: a unit square with colored sides.

  i.e, an element of C^4, e.g., 
- Tile set: a set $\tau \subset C^4$
- Tiling: a configuration $f : \mathbb{Z}^2 \to \tau$, where

  every two adjacent tiles share the same color on the common side,

  e.g.,



Aperiodic tilings

Aperiodic tilings

$T \in \mathbb{Z}^2$ is a **period** if $f(x + T) = f(x)$ for all x.

Aperiodic tilings

$T \in \mathbb{Z}^2$ is a **period** if $f(x + T) = f(x)$ for all x.

Theorem. [Berger] There exists a non-empty SFT (even a tileset) on $\mathbb{Z}^2$ where all configurations are aperiodic.

Aperiodic tilings

$T \in \mathbb{Z}^2$ is a **period** if $f(x+T) = f(x)$ for all x.

Theorem. [Berger] There exists a non-empty SFT (even a tileset) on $\mathbb{Z}^2$ where all configurations are aperiodic.

A construction of an aperiodic tile set:

Aperiodic tilings

$T \in \mathbb{Z}^2$ is a **period** if $f(x + T) = f(x)$ for all x.

Theorem. [Berger] There exists a non-empty SFT (even a tileset) on $\mathbb{Z}^2$ where all configurations are aperiodic.

A construction of an aperiodic tile set:

- define **self-similar** tile sets

Aperiodic tilings

$T \in \mathbb{Z}^2$ is a **period** if $f(x+T) = f(x)$ for all x.

Theorem. [Berger] There exists a non-empty SFT (even a tileset) on $\mathbb{Z}^2$ where all configurations are aperiodic.

A construction of an aperiodic tile set:

- define **self-similar** tile sets
- observe that *every* **self-similar** tile set is aperiodic

Aperiodic tilings

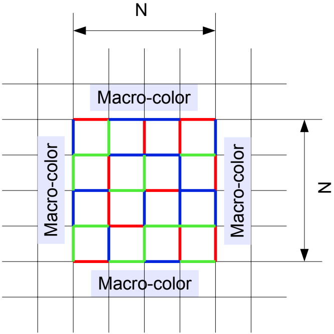
$T \in \mathbb{Z}^2$ is a **period** if $f(x+T) = f(x)$ for all x.

Theorem. [Berger] There exists a non-empty SFT (even a tileset) on $\mathbb{Z}^2$ where all configurations are aperiodic.

A construction of an aperiodic tile set:

- define **self-similar** tile sets
- observe that *every* **self-similar** tile set is aperiodic
- construct *some* **self-similar** tile set

Macro-tile:



an $N \times N$ square made of matching τ-tiles

Fix a tile set τ and an integer $N > 1$.

Fix a tile set τ and an integer $N > 1$.

Definition 1. A τ-**macro-tile**: an $N \times N$ square made of τ-tiles.

Fix a tile set τ and an integer $N > 1$.

Definition 1. A τ-**macro-tile**: an $N \times N$ square made of τ-tiles.

Definition 2. A tile set ρ is **simulated** by τ: there exists a family of τ-macro-tiles R such that

- R is *isomorphic* to ρ, and
- every τ-tiling can be *uniquely* split by an $N \times N$ grid into macro-tiles from R.

Example.

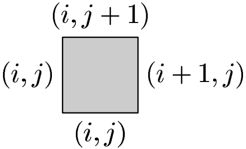
A tile set ρ: Trivial tile set (only one color)

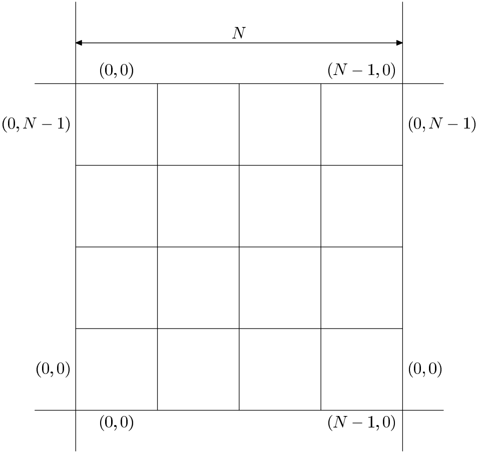
Example.

A tile set ρ: Trivial tile set (only one color)
A tile set τ: A tile set that simulates a trivial tile set ρ

Example.

A tile set ρ: Trivial tile set (only one color)
A tile set τ: A tile set that simulates a trivial tile set ρ




N
$(0,0)$
$(N-1,0)$
$(0,N-1)$
$(0,N-1)$
$(0,0)$
$(0,0)$
$(0,0)$
$(N-1,0)$

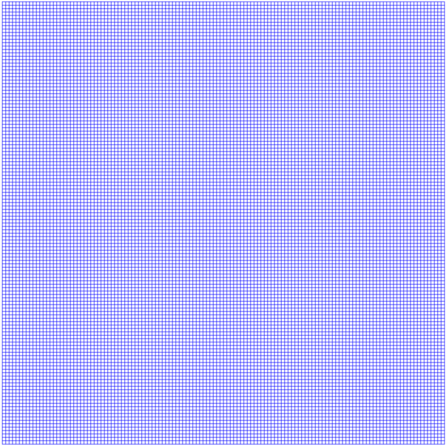
Self-similar tile set: a tile set that simulates a set of macrotiles *isomorphic* to itself.

Self-similar tile set: a tile set that simulates a set of macrotiles *isomorphic* to itself.

Proposition. Self-similar tile set is aperiodic.

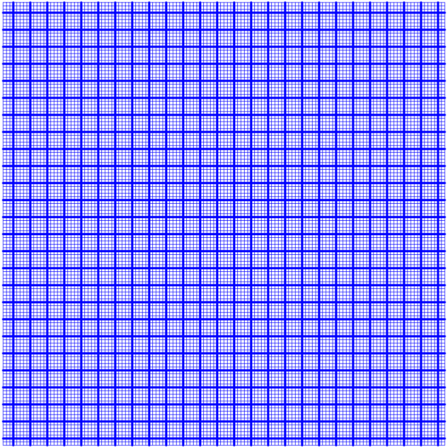
Self-similar tile set: a tile set that simulates a set of macrotiles *isomorphic* to itself.

Proposition. Self-similar tile set is aperiodic.
Sketch of the proof:



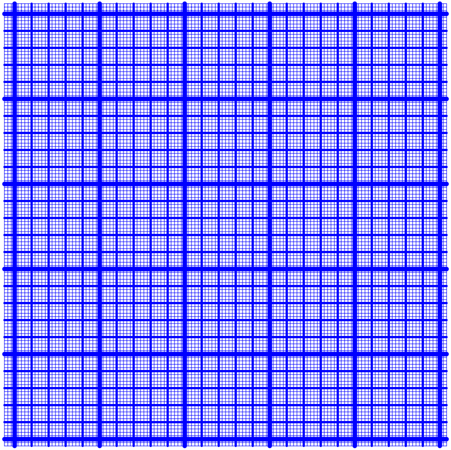
Self-similar tile set: a tile set that simulates a set of macrotiles *isomorphic* to itself.

Proposition. Self-similar tile set is aperiodic.
Sketch of the proof:



Self-similar tile set: a tile set that simulates a set of macrotiles *isomorphic* to itself.

Proposition. Self-similar tile set is aperiodic.
Sketch of the proof:



Simulating a given tile set ρ by macro-tiles.

Simulating a given tile set ρ by macro-tiles.
Representation of the tile set ρ:

Simulating a given tile set ρ by macro-tiles.
Representation of the tile set ρ:

- colors of a tile set ρ $\Longrightarrow$ k-bits strings

Simulating a given tile set ρ by macro-tiles.
Representation of the tile set ρ:

- colors of a tile set ρ $\Longrightarrow$ k-bits strings
- a tile set ρ $\Longrightarrow$ a predicate $\mathcal{P}(x_1, x_2, x_3, x_4)$ on 4-tuples of colors

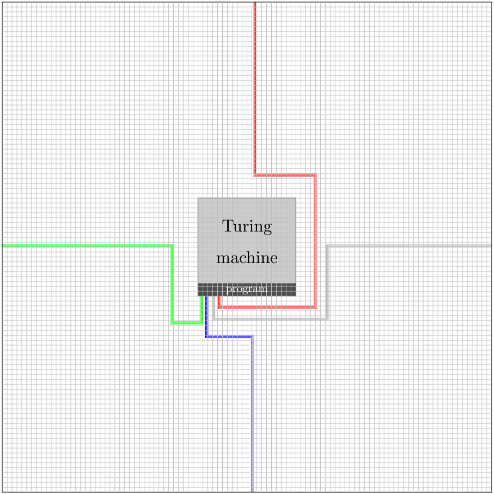
Simulating a given tile set ρ by macro-tiles.
Representation of the tile set ρ:

- colors of a tile set ρ $\Longrightarrow$ k-bits strings
- a tile set ρ $\Longrightarrow$ a predicate $\mathcal{P}(x_1, x_2, x_3, x_4)$ on 4-tuples of colors

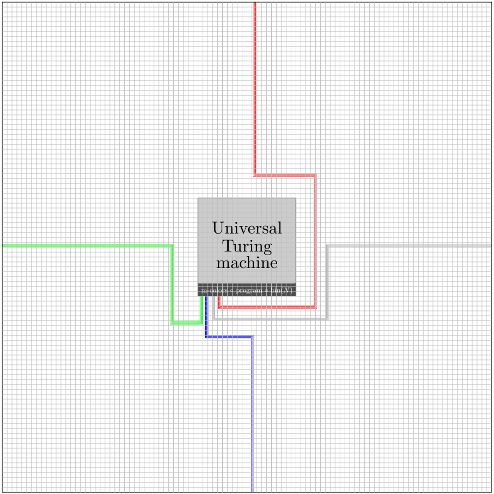
$\Downarrow$

a TM that accepts only 4-tuples of colors for the ρ-tiles

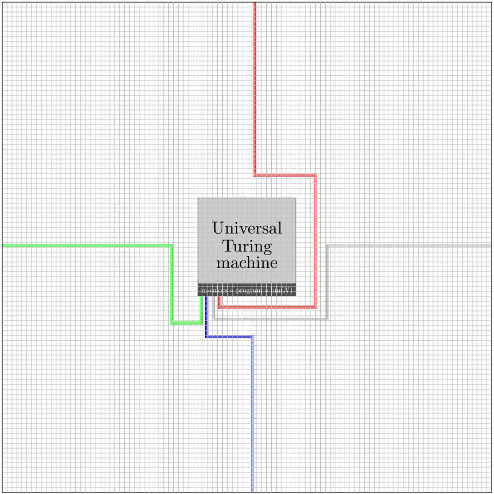
The scheme of implementation:



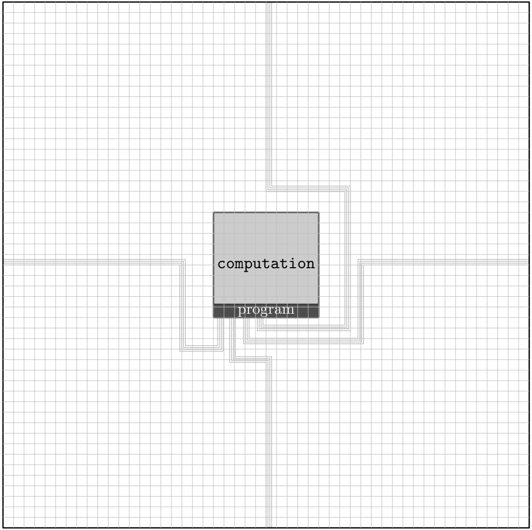
A more generic construction: universal TM + program

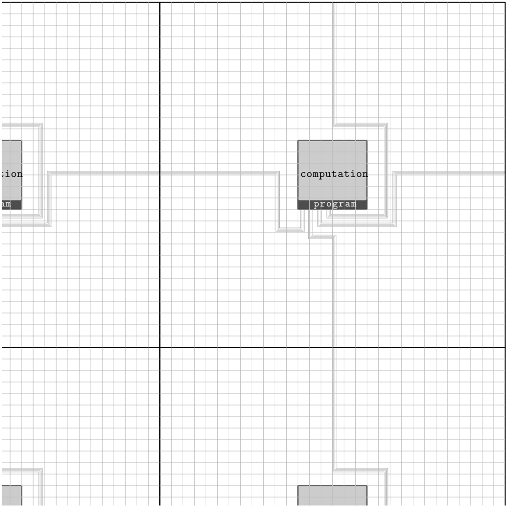


A more generic construction: universal TM + program

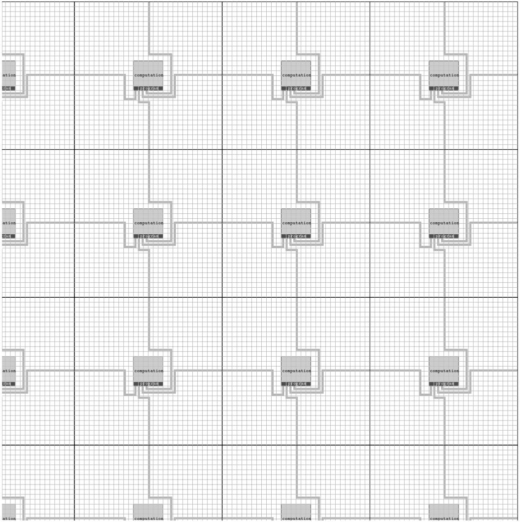


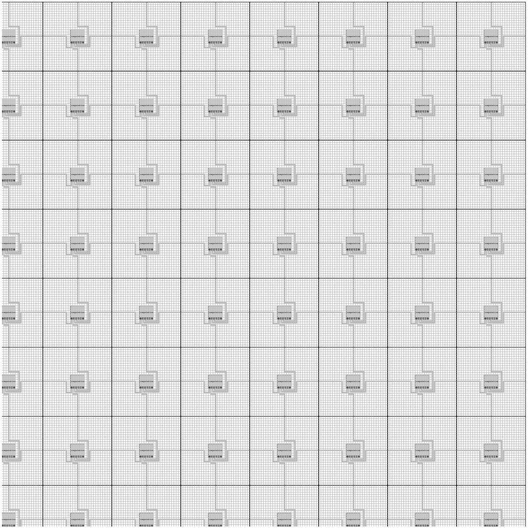
A fixed point: simulating tile set = simulated tile set

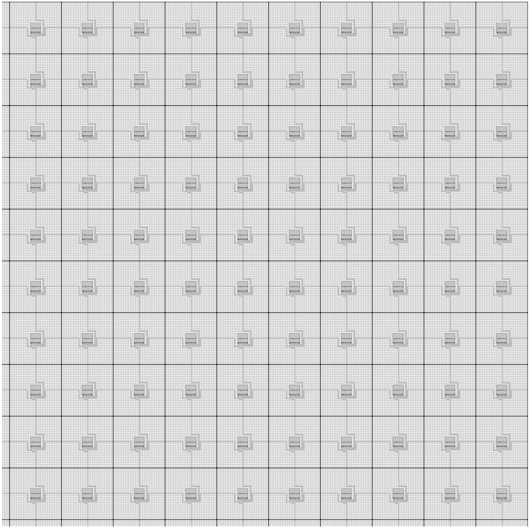

computation
program

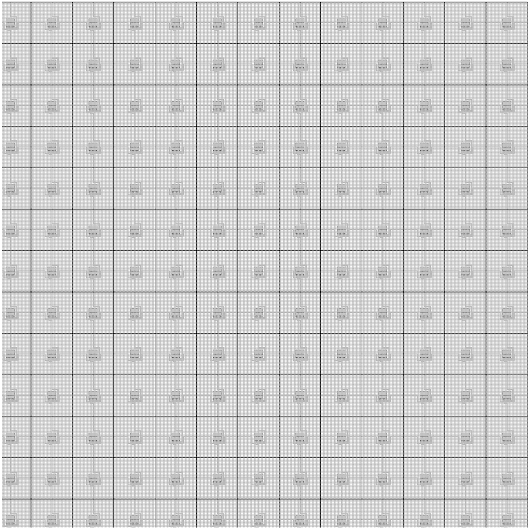


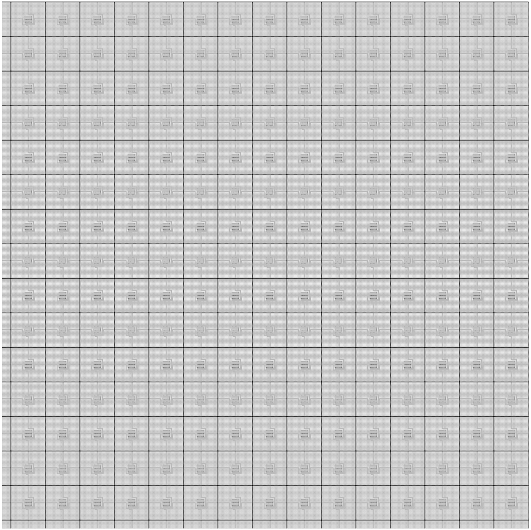
tion
am
computation
program

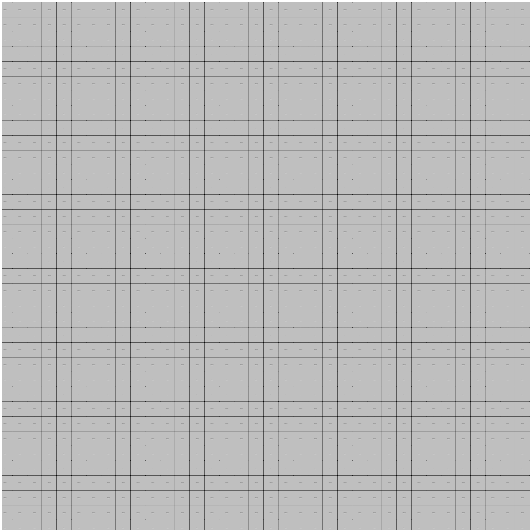

computation
program

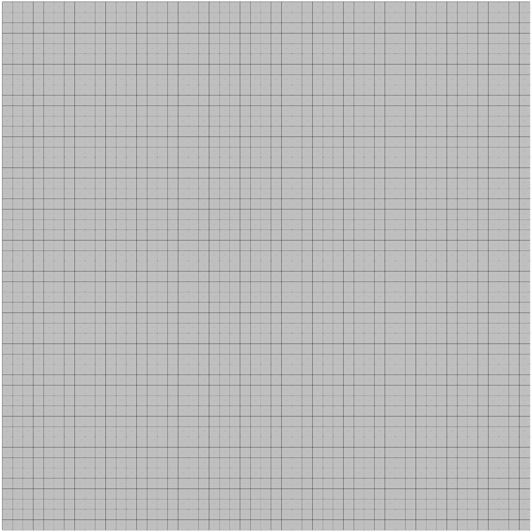


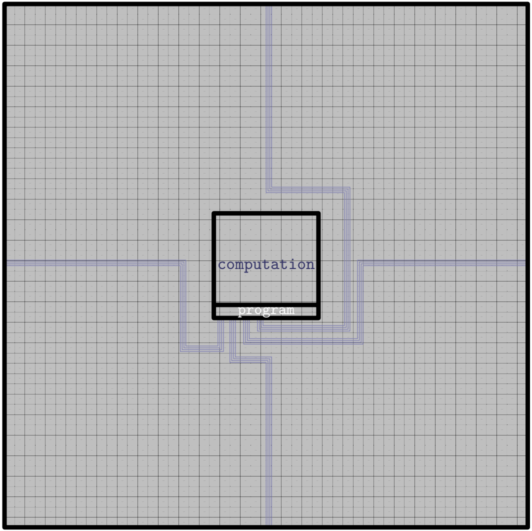


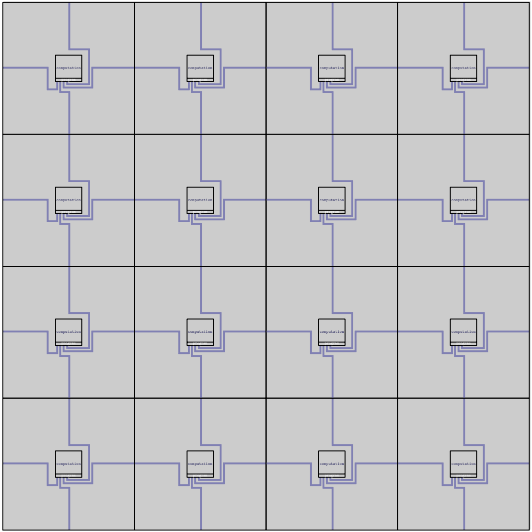


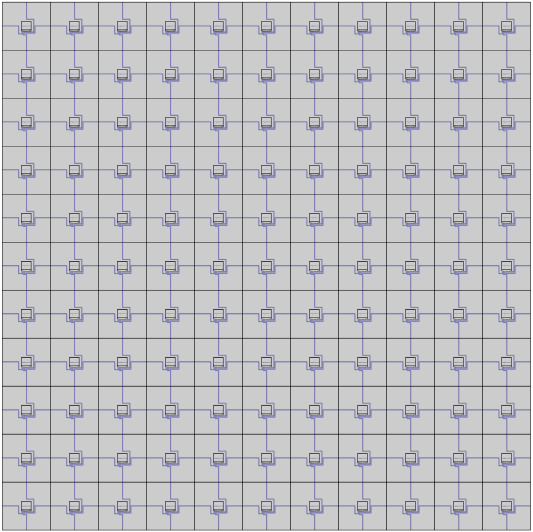


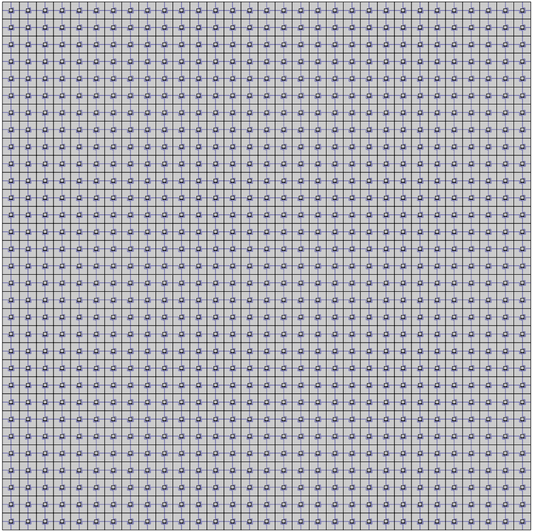


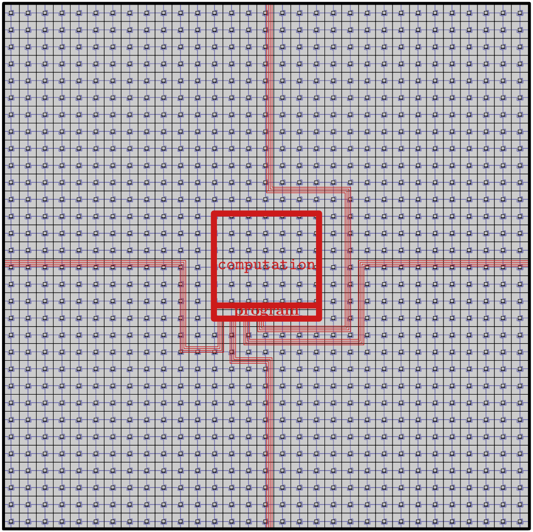

.

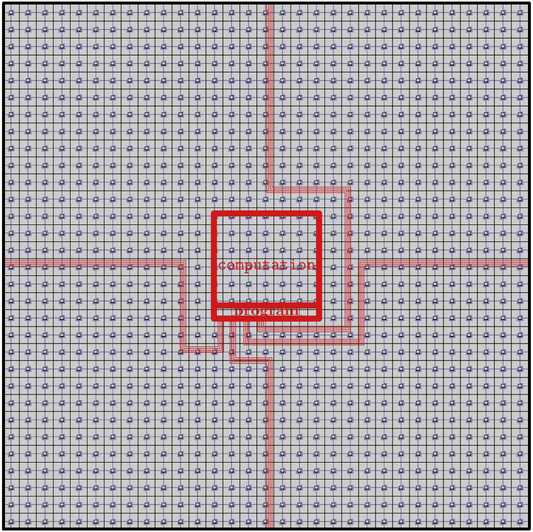

computation
program


computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program
computation
program






computation
program



N.B. We can variate the zoom factor!

Theorem. [Ballier–Ollinger]
There exists an *aperiodic* and *minimal* SFT.

Theorem. [Ballier–Ollinger]
There exists an *aperiodic* and *minimal* SFT.

How to make an aperiodic SFT *minimal* ?

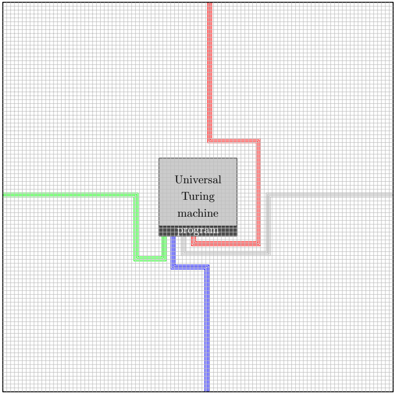
Theorem. [Ballier–Ollinger]
There exists an *aperiodic* and *minimal* SFT.

How to make an aperiodic SFT *minimal* ?

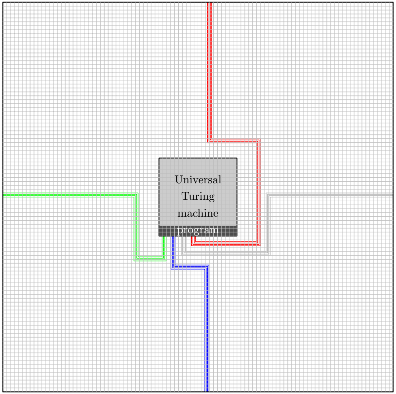
Ollinger: take Robinson's construction and remove everything that may appear *not* infinitely often

our plan: take the fixed-point construction and enforce everything that may appear at least *once*

How to get **aperiodicity** + **minimality** ?

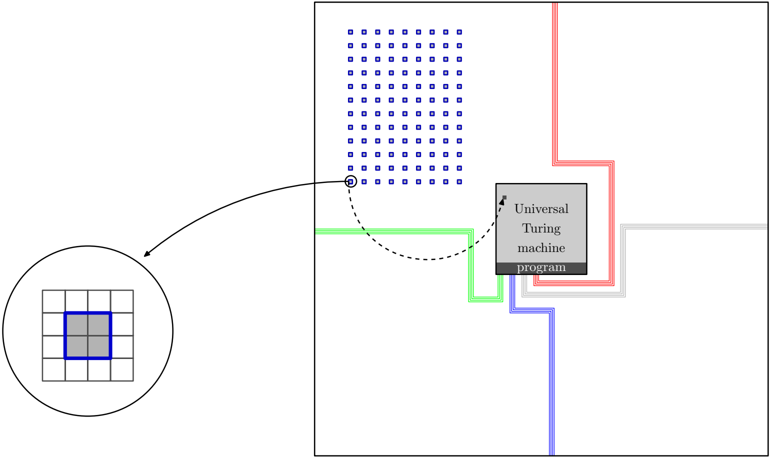


How to get **aperiodicity** + **minimality** ?

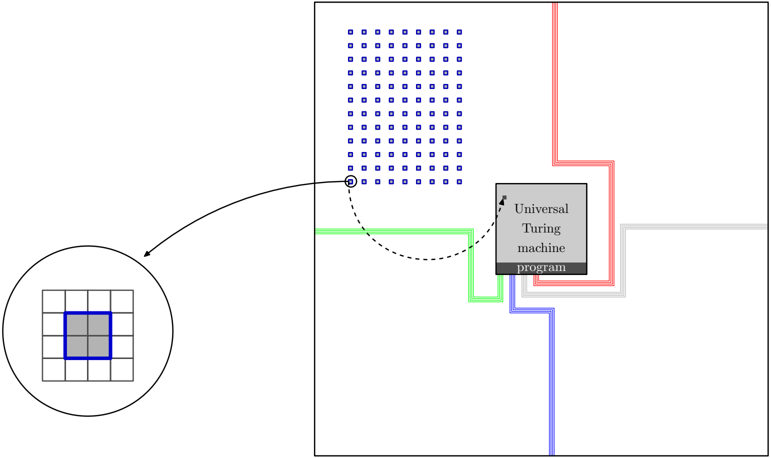


The problematic part is the computation zone...

Duplicate each 2×2 pattern that *may* appear in the computation zone!

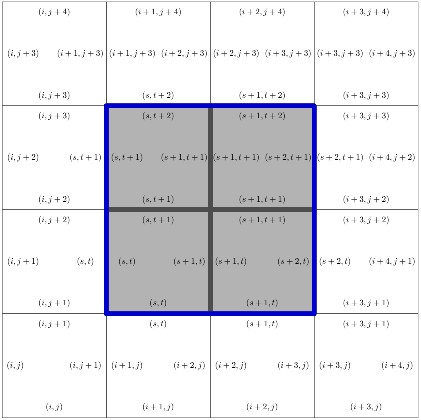


Duplicate each 2×2 pattern that *may* appear in the computation zone!

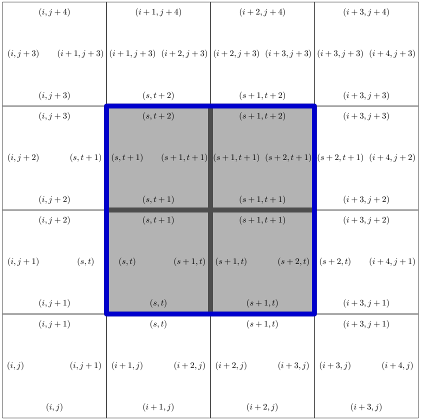


Imprisonment for diversity!

A slot for a 2×2 patterns from the computation zone:

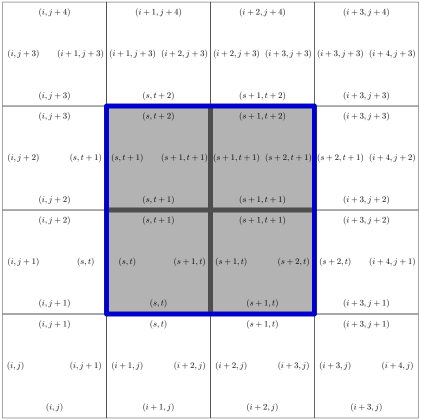


A slot for a 2×2 patterns from the computation zone:



Grey cells: It seems we sit inside of the computation zone,
we make part of a huge computation!

A slot for a 2×2 patterns from the computation zone:



Grey cells: It seems we sit inside of the computation zone,
we make part of a huge computation!

White cells: Guys, you four are living in a small prison
in the middle of nowhere...

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies

$h = \frac{\sqrt{5}-1}{2}$

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies
$h = \frac{\sqrt{5}-1}{2}$, or $6/\pi^2$

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies
$h = \frac{\sqrt{5}-1}{2}$, or $6/\pi^2$, or $\sum_{n=1}^{\infty} \frac{1}{2^{n!}}$

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies
$h = \frac{\sqrt{5}-1}{2}$, or $6/\pi^2$, or $\sum_{n=1}^{\infty} \frac{1}{2^{n!}}$, or $\sqrt[3]{\pi + e}$

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies

$h = \frac{\sqrt{5}-1}{2}$, or $6/\pi^2$, or $\sum_{n=1}^{\infty} \frac{1}{2^{n!}}$, or $\sqrt[3]{\pi + e}$, etc.

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies
$h = \frac{\sqrt{5}-1}{2}$, or $6/\pi^2$, or $\sum_{n=1}^{\infty} \frac{1}{2^{n!}}$, or $\sqrt[3]{\pi + e}$, etc.

Any number that can appear in real maths.

Theorem [Hochman–Meyerovitch] Every explicitly defined number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

e.g., there exist SFTs with entropies
$h = \frac{\sqrt{5}-1}{2}$, or $6/\pi^2$, or $\sum_{n=1}^{\infty} \frac{1}{2^{n!}}$, or $\sqrt[3]{\pi + e}$, etc.

Any number that can appear in real maths. And even slightly more.

Theorem [Hochman–Meyerovitch] Every right recursively enumerable number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

Theorem [Hochman–Meyerovitch] Every right recursively enumerable number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

Questions:

Theorem [Hochman–Meyerovitch] Every right recursively enumerable number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

Questions:

- **Hochman–Meyerovitch:** What about transitivity?

Theorem [Hochman–Meyerovitch] Every right recursively enumerable number $h \geq 0$ is the entropy of some SFT on $\mathbb{Z}^2$.

Questions:

- **Hochman–Meyerovitch:** What about transitivity?
- **Schraudner:** What about some kind of (uniform) mixing?

Theorem. Every right recursively enumerable number $h \geq 0$ is the entropy of some weakly irreducible and transitive SFT on $\mathbb{Z}^2$.

Theorem. Every right recursively enumerable number $h \geq 0$ is the entropy of some weakly irreducible and transitive SFT on $\mathbb{Z}^2$.

Weakly irreducible: every two globally admissible finite patterns can be combined in one infinite configuration

Theorem. Every right recursively enumerable number $h \geq 0$ is the entropy of some weakly irreducible and transitive SFT on $\mathbb{Z}^2$.

Weakly irreducible: every two globally admissible finite patterns can be combined in one infinite configuration (with a bounded distance).

Theorem. Every right recursively enumerable number $h \geq 0$ is the entropy of some weakly irreducible and transitive SFT on $\mathbb{Z}^2$.

Weakly irreducible: every two globally admissible finite patterns can be combined in one infinite configuration (with a bounded distance).

Transitive: There exists a configuration that contains all globally admissible finite patterns.

Sketch of the proof:

Sketch of the proof:

(step 1) construct a tileset τ such that

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility
- $\text{entropy}(\tau) = 0$.

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility
- entropy$(\tau) = 0$.

(step 2) an SFT τ' :

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility
- entropy$(\tau) = 0$.

(step 2) an SFT τ' :

- make two copies of each red tile

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility
- entropy$(\tau) = 0$.

(step 2) an SFT τ' :

- make two copies of each red tile

Result:

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility
- $\text{entropy}(\tau) = 0$.

(step 2) an SFT τ' :

- make two copies of each red tile

Result:

- $\text{entropy}(\tau') = h$

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility
- $\text{entropy}(\tau) = 0$.

(step 2) an SFT τ' :

- make two copies of each red tile

Result:

- $\text{entropy}(\tau') = h$
- weak irreducibility: got for free

Sketch of the proof:

(step 1) construct a tileset τ such that

- two types of tiles, blue and red
- guarantee that $\limsup[\text{density of red tiles}] = h$
- guarantee transitivity and weak irreducibility
- $\text{entropy}(\tau) = 0$.

(step 2) an SFT τ' :

- make two copies of each red tile

Result:

- $\text{entropy}(\tau') = h$
- weak irreducibility: got for free
- transitivity: random instantiation of red tiles

technical trick: a self-simulating SFT with a growing zoom factor, s.t.

- $\limsup[\text{density of red tiles}] = h$
- transitive and irreducibile

technical trick: a self-simulating SFT with a growing zoom factor, s.t.

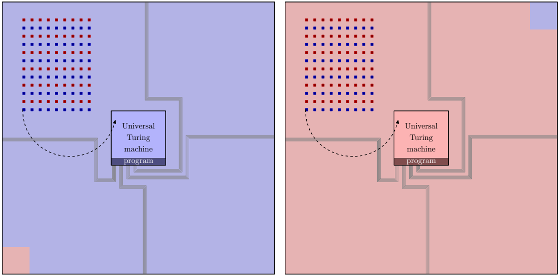
- $\limsup$[density of red tiles] $= h$
- transitive and irreducibile

blue macro-tiles and red macro-tiles:

technical trick: a self-simulating SFT with a growing zoom factor, s.t.

- lim sup[density of red tiles] $= h$
- transitive and irreducibile

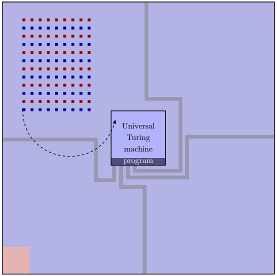
blue macro-tiles and red macro-tiles:

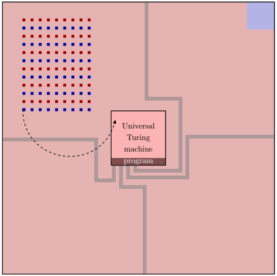


technical trick: a self-simulating SFT with a growing zoom factor, s.t.

- lim sup[density of red tiles] $= h$
- transitive and irreducibile

blue macro-tiles and red macro-tiles:



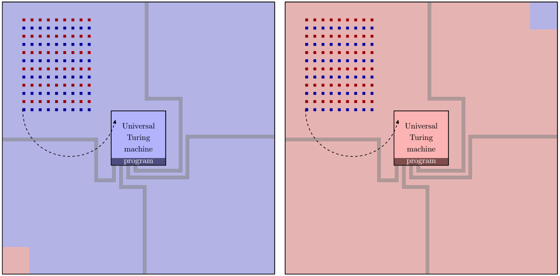


- most ground-level tiles in a blue macro-tile are blue

technical trick: a self-simulating SFT with a growing zoom factor, s.t.

- $\limsup[\text{density of red tiles}] = h$
- transitive and irreducibile

blue macro-tiles and red macro-tiles:

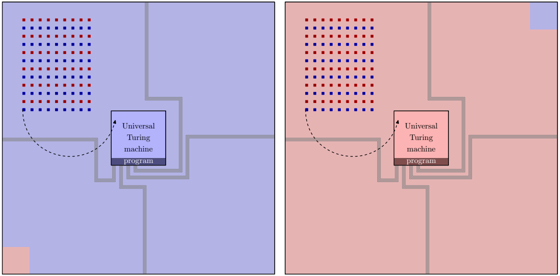


- most ground-level tiles in a blue macro-tile are blue
- most ground-level tiles in a red macro-tile are red

technical trick: a self-simulating SFT with a growing zoom factor, s.t.

- lim sup[density of red tiles] $= h$
- transitive and irreducibile

blue macro-tiles and red macro-tiles:



- most ground-level tiles in a blue macro-tile are blue
- most ground-level tiles in a red macro-tile are red
- the fraction red tiles in a red macro-tile approaches the limit

Hierarchical constructions: self-simulating vs Berger/Robinson's

Hierarchical constructions: self-simulating vs Berger/Robinson's

	self-simulating tilings	Robinson's construction
undecidability of the domino problem	+	Berger'66
tilings with only noncomputable points	+	Hanf–Myers'74
any *effectively closed* 1D subshift is isomorphic to the subdynamics of a 2D SFT	Durand–R–Shen	Aubrun–Sablik
similar result for *minimal* subshifts	Durand–R	?
strongly deterministic tilings	–	Kari, Papasoglu, Lukkarila, Le Gloannec, Ollinger
pairwise different black squares in the white ocean	Westrick	?

Hierarchical structures but no universal computation:

Hierarchical structures but no universal computation:

Mozes'89: tilings simulate 'rectangular' substitution system

Goodman-Strauss'98: tilings simulate any geometric substitution system

Hierarchical structures but no universal computation:

Mozes'89: tilings simulate 'rectangular' substitution system

Goodman-Strauss'98: tilings simulate any geometric substitution system

Non-hierarchical structure and non-universal computations:

Hierarchical structures but no universal computation:

Mozes'89: tilings simulate 'rectangular' substitution system

Goodman-Strauss'98: tilings simulate any geometric substitution system

Non-hierarchical structure and non-universal computations:

Culik–Kari'96

Hierarchical structures but no universal computation:

Mozes'89: tilings simulate 'rectangular' substitution system

Goodman-Strauss'98: tilings simulate any geometric substitution system

Non-hierarchical structure and non-universal computations:

Culik–Kari'96 + Jeandel–Rao'15:

Hierarchical structures but no universal computation:

Mozes'89: tilings simulate 'rectangular' substitution system

Goodman-Strauss'98: tilings simulate any geometric substitution system

Non-hierarchical structure and non-universal computations:

Culik–Kari'96 + Jeandel–Rao'15:

- embedded simple finite automata (transducers)

Hierarchical structures but no universal computation:

Mozes'89: tilings simulate 'rectangular' substitution system

Goodman-Strauss'98: tilings simulate any geometric substitution system

Non-hierarchical structure and non-universal computations:

Culik–Kari'96 + **Jeandel–Rao'15:**

- embedded simple finite automata (transducers),
- aperiodic tilings with *very small* number of tiles

Hierarchical structures but no universal computation:

Mozes'89: tilings simulate 'rectangular' substitution system
Goodman-Strauss'98: tilings simulate any geometric substitution system

Non-hierarchical structure and non-universal computations:

Culik–Kari'96 + Jeandel–Rao'15:

- embedded simple finite automata (transducers),
- aperiodic tilings with *very small* number of tiles
- tilings with really interesting properties

Take home messages:

Take home messages:

- embedding a TM in a tiling can be useful even if you do not care about computability

Take home messages:

- embedding a TM in a tiling can be useful even if you do not care about computability
- self-simulating tilings is a flexible tool

Take home messages:

- embedding a TM in a tiling can be useful even if you do not care about computability
- self-simulating tilings is a flexible tool, arguably the first (lazy) option to try

Take home messages:

- embedding a TM in a tiling can be useful even if you do not care about computability
- self-simulating tilings is a flexible tool, arguably the first (lazy) option to try
- for more subtle problems there exist trickier techniques

Take home messages:

- embedding a TM in a tiling can be useful even if you do not care about computability
- self-simulating tilings is a flexible tool, arguably the first (lazy) option to try
- for more subtle problems there exist trickier techniques

Thank you!