Minicourse on *information, complexity and organisation in multidimensional symbolic dynamics*

Effect of dynamical constraints on the computational power of multidimensional SFT

Silvere Gangloff

March 19, 2021

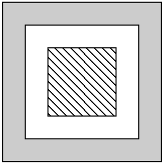
sgangloff@agh.edu.pl ; silvere.gangloff@gmx.com

Reminder: the characterization of entropies of multidimensional SFT

Theorem[M.Hochman,T.Meyerovitch]: for all $d \geq 2$, the possible values of entropy for d-dimensional SFT are the non-negative Π_1-computable numbers.

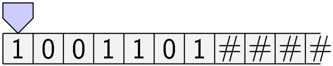
Π_1-computable: exists an algorithm which on input n outputs $r_n \in \mathbb{Q}$ s.t. $r_n \downarrow x$.

Question: what are the possible values for entropy on **irreducible** multidimensional SFT ?



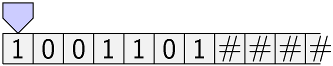
Dynamical constraints and computational power: analogy with the human brain

Classes of dynamical systems *containing* universal computation:



Dynamical constraints and computational power: analogy with the human brain

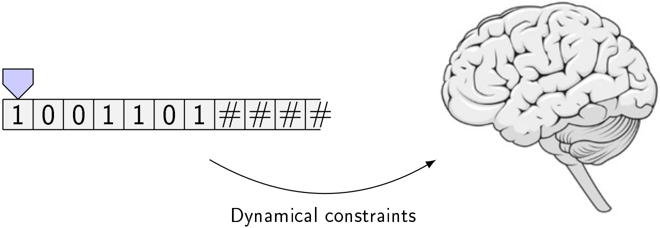
Classes of dynamical systems *containing* universal computation:



Why is the human brain not a Turing machine ?

Dynamical constraints and computational power: analogy with the human brain

Classes of dynamical systems *containing* universal computation:



Why is the human brain not a Turing machine ?

Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

Example: block gluing

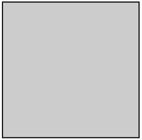
Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

X multidimensional subshift with block gluing:

Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

X multidimensional subshift with block gluing:

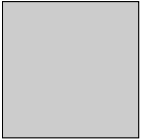


$\in \mathcal{L}(X)$

Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

X multidimensional subshift with block gluing:

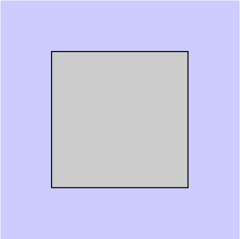


$\in \mathcal{L}(X)$

Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

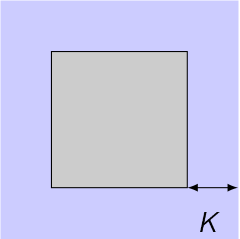
X multidimensional subshift with block gluing:



Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

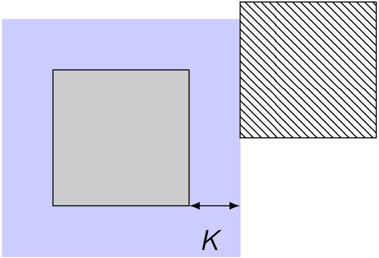
X multidimensional subshift with block gluing:



Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

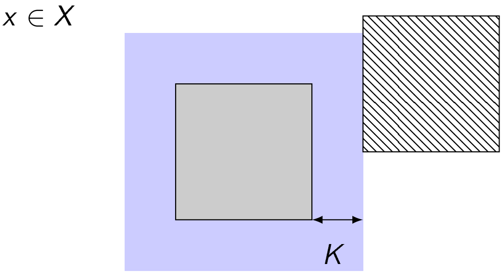
X multidimensional subshift with block gluing:



Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

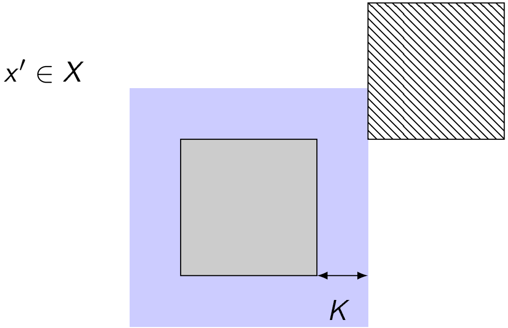
X multidimensional subshift with block gluing:



Example: block gluing

Question: how do dynamical constraints affect the implementation of universal computation in multidimensional SFT ?

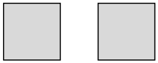
X multidimensional subshift with block gluing:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

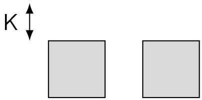
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

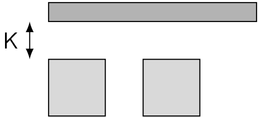
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

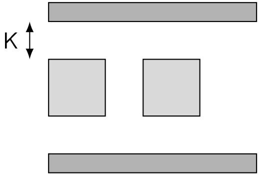
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

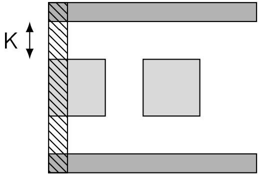
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

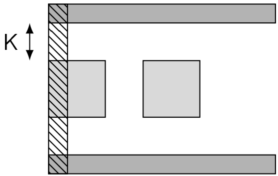
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

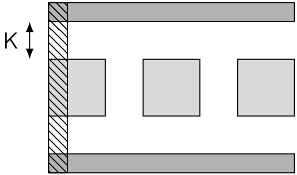
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

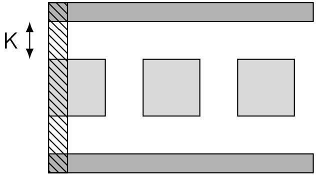
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

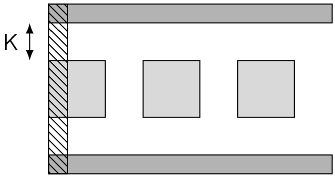
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

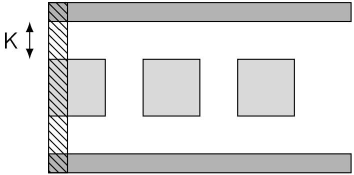
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

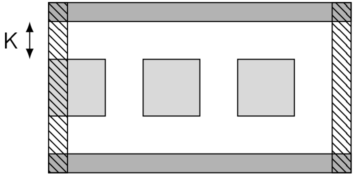
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

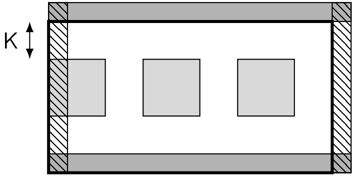
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

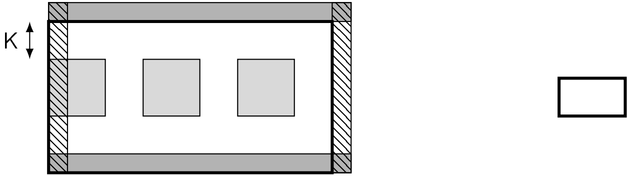
Proof: For nearest neighbour subshifts:



Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

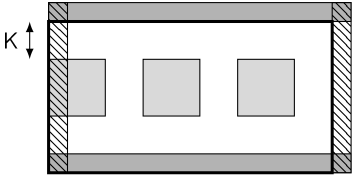
Proof: For nearest neighbour subshifts:

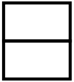


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:

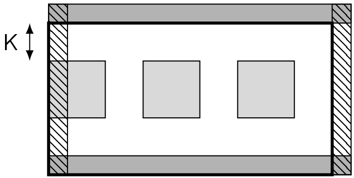


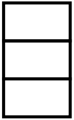


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:

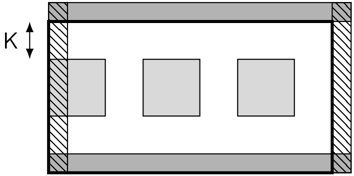


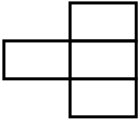


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:

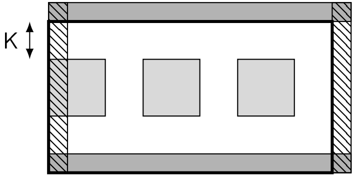


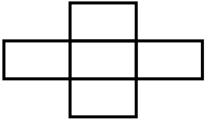


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:

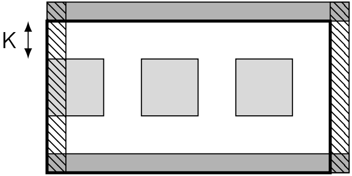


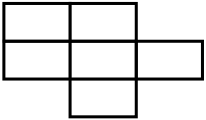


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:

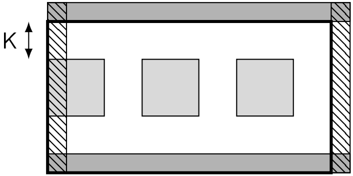


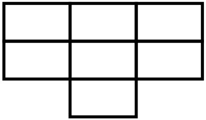


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:

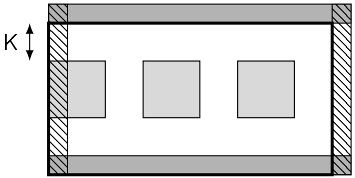


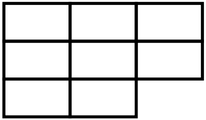


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:

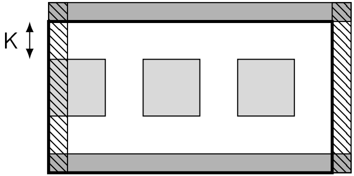


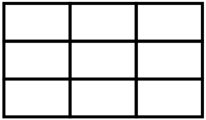


Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:



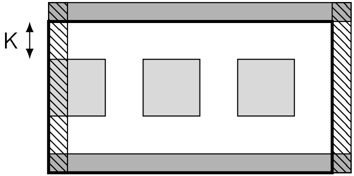


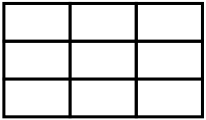
Implications: X not aperiodic (in particular no hierarchical structures);

Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:





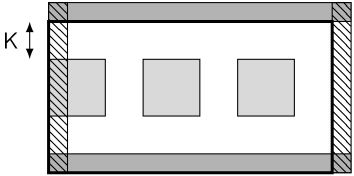
Implications: X not aperiodic (in particular no hierarchical structures);

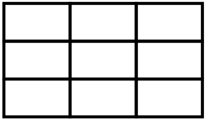
Decidable language: algorithm which decides if a pattern on alphabet $\mathcal{A}$ is in $\mathcal{L}(X)$ or not.

Density of periodic points

Theorem: If X is a 2-dimensional block gluing SFT, its periodic configurations are dense.

Proof: For nearest neighbour subshifts:





Implications: X not aperiodic (in particular no hierarchical structures);

Decidable language: algorithm which decides if a pattern on alphabet $\mathcal{A}$ is in $\mathcal{L}(X)$ or not.

Question: what happens in dimension 3 ?

Computability of entropy for block gluing SFT

Reminders: x is computable when there is an algorithm which on input n outputs r_n s.t. $|x - r_n| \leq 2^{-n}$.

For X d-dim. subshift:

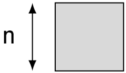
$$h(X) = \lim_n \frac{\log(N_n(X))}{n^d},$$

where $N_n(X)$ is the number of n-blocks appearing in configurations of X.

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:

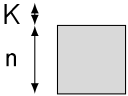


.

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

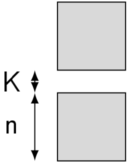
Proof:



Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:

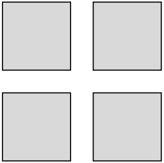


.

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

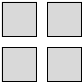
Proof:



Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:

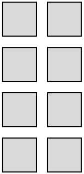


.

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

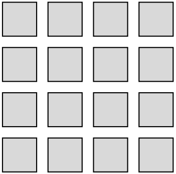
Proof:



Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

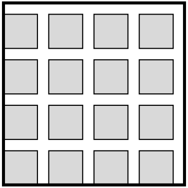
Proof:



Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:

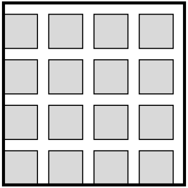


.

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:

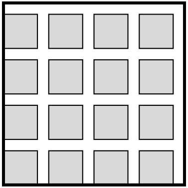


For all $m \geq 1$: $N_{m(n+K)}(X) \geq N_n(X)^{m^2}$.

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:



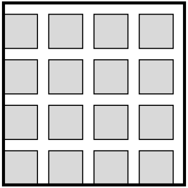
For all $m \geq 1$: $N_{m(n+K)}(X) \geq N_n(X)^{m^2}$.

$$\frac{(mn)^2}{(m(n+K))^2}\frac{\log(N_n(X))}{n^2} \leq \frac{N_{m(n+K)}(X)}{(m(n+K))^2}.$$

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:



For all $m \geq 1$: $N_{m(n+K)}(X) \geq N_n(X)^{m^2}$.

$$\frac{(mn)^2}{(m(n+K))^2} \frac{\log(N_n(X))}{n^2} \leq \frac{N_{m(n+K)}(X)}{(m(n+K))^2}.$$

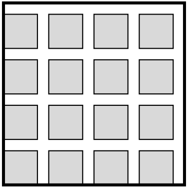
When $m \to +\infty$:

$$\frac{n^2}{(n+K)^2} \frac{\log(N_n(X))}{n^2} \leq h(X) \qquad .$$

Computability of entropy for block gluing SFT

Theorem: For X a 2-dimensional block gluing SFT: $h(X)$ is a computable number.

Proof:



For all $m \geq 1$: $N_{m(n+K)}(X) \geq N_n(X)^{m^2}$.

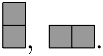
$$\frac{(mn)^2}{(m(n+K))^2} \frac{\log(N_n(X))}{n^2} \leq \frac{N_{m(n+K)}(X)}{(m(n+K))^2}.$$

When $m \to +\infty$:

$$\frac{n^2}{(n+K)^2} \frac{\log(N_n(X))}{n^2} \leq h(X) \leq \frac{\log(N_n(X))}{n^2}.$$

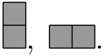
An example: the hard core model

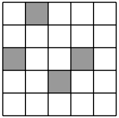
Consider X_0 on $\mathcal{A} = \{\square, \blacksquare\}$, and $\mathcal{F}$ the following patterns:



An example: the hard core model

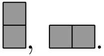
Consider X_0 on $\mathcal{A} = \{\square, \blacksquare\}$, and $\mathcal{F}$ the following patterns:

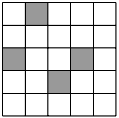




An example: the hard core model

Consider X_0 on $\mathcal{A} = \{\square, \blacksquare\}$, and $\mathcal{F}$ the following patterns:

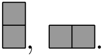


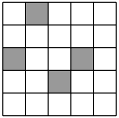


The entropy of this subshift is computable. More efficient than block gluing: *Approximating the hard square entropy constant with probabilistic methods*, **R.Pavlov**.

An example: the hard core model

Consider X_0 on $\mathcal{A} = \{\square, \blacksquare\}$, and $\mathcal{F}$ the following patterns:





The entropy of this subshift is computable. More efficient than block gluing: *Approximating the hard square entropy constant with probabilistic methods*, **R.Pavlov**.

Question: closed formula for the $h(X_0)$?

The result of R.Pavlov and M.Schraudner

Computability condition $(*)$: x rational or has an infinite continuous fraction expansion $[a_0; a_1; ...]$ such that for (t_n) s.t. $t_0 = 1$, $t_1 = a_1$ and

$$t_n = a_n t_{n-1} + t_{n-2},$$

there exists an algorithm which produces $a_1, ..., a_N$ in $a_N \cdot t_{N-1}$ steps.

The result of R.Pavlov and M.Schraudner

Computability condition $(*)$: x rational or has an infinite continuous fraction expansion $[a_0; a_1; ...]$ such that for (t_n) s.t. $t_0 = 1$, $t_1 = a_1$ and

$$t_n = a_n t_{n-1} + t_{n-2},$$

there exists an algorithm which produces $a_1, ..., a_N$ in $a_N \cdot t_{N-1}$ steps.

Theorem: for $d \geq 3$ and $x = z \log(M)$ with M integer and z satisfies $(*)$, x is entropy of a d-dimensional block gluing SFT.

The result of R.Pavlov and M.Schraudner

Computability condition $(*)$: x rational or has an infinite continuous fraction expansion $[a_0; a_1; \ldots]$ such that for (t_n) s.t. $t_0 = 1,\ t_1 = a_1$ and

$$t_n = a_n t_{n-1} + t_{n-2},$$

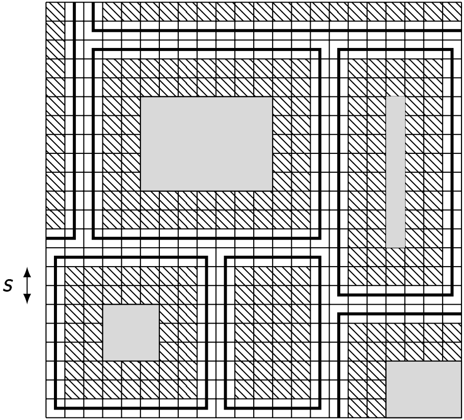
there exists an algorithm which produces $a_1, \ldots, a_N$ in $a_N \cdot t_{N-1}$ steps.

Theorem: for $d \geq 3$ and $x = z \log(M)$ with M integer and z satisfies $(*)$, x is entropy of a d-dimensional block gluing SFT.

Entropies realizable by block gluing $\mathbb{Z}^d$ shifts of finite type, **R.Pavlov, M.Schraudner**.

Schema of proof: an approach by operators

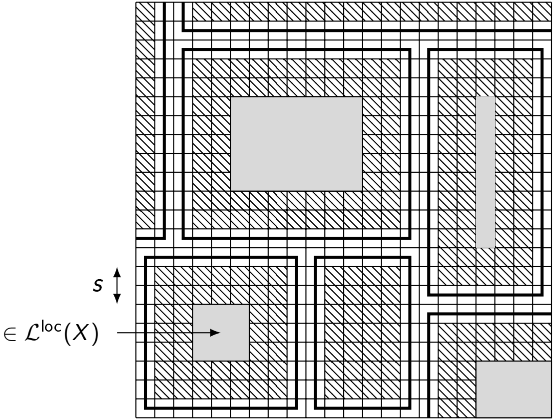
Let us fix X subshift of finite type.
Denote X_s the following subshift:



Schema of proof: an approach by operators

Let us fix X subshift of finite type.

Denote $\nu_s(X)$ the following subshift:



Schema of proof: a condition on the complexity function:

Definition: X **upgradable**: for all k, l,
$|\mathcal{L}^{\text{loc}}_{k,l}(X)| \leq e^{h(X) \cdot kl + \gamma \cdot (k+l)}$.

Schema of proof: a condition on the complexity function:

Definition: X **upgradable**: for all k, l,
$|\mathcal{L}^{\text{loc}}_{k,l}(X)| \leq e^{h(X)\cdot kl+\gamma\cdot(k+l)}$.

Lemma: if X is upgradable, for s large enough, $h(\nu_s(X)) = h(X)$.

Schema of proof: a condition on the complexity function:

Definition: X **upgradable**: for all k, l,
$|\mathcal{L}^{\mathrm{loc}}_{k,l}(X)| \leq e^{h(X)\cdot kl+\gamma\cdot(k+l)}$.

Lemma: if X is upgradable, for s large enough, $h(\nu_s(X)) = h(X)$.

Proof: (schema)

Schema of proof: a condition on the complexity function:

Definition: X **upgradable**: for all k, l,
$|\mathcal{L}_{k,l}^{\text{loc}}(X)| \leq e^{h(X)\cdot kl+\gamma\cdot(k+l)}$.

Lemma: if X is upgradable, for s large enough, $h(\nu_s(X)) = h(X)$.

Proof: (schema)

1. Straightforwardly, $h(X) \leq h(\nu_s(X))$.

Schema of proof: a condition on the complexity function:

Definition: X **upgradable**: for all k, l,
$|\mathcal{L}_{k,l}^{\text{loc}}(X)| \leq e^{h(X)\cdot kl + \gamma\cdot(k+l)}$.

Lemma: if X is upgradable, for s large enough, $h(\nu_s(X)) = h(X)$.

Proof: (schema)

1. Straightforwardly, $h(X) \leq h(\nu_s(X))$.

2. Upgradability allows to bound $N_n(\nu_s(X))$ with

$$(n^3+1)28^{6n^2} e^{h(X)(n+2s)^3},$$

which implies $h(X) \geq h(\nu_s(X))$.

Schema of proof: using the simulation of effective subshifts:

Reminder: every effective one-dimensional subshift can be simulated by a bidimensional SFT.

Schema of proof: using the simulation of effective subshifts:

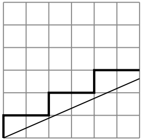
Reminder: every effective one-dimensional subshift can be simulated by a bidimensional SFT.

Take $\alpha \in [0,1[$ with condition $(*)$.

Schema of proof: using the simulation of effective subshifts:

Reminder: every effective one-dimensional subshift can be simulated by a bidimensional SFT.

Take $\alpha \in [0,1[$ with condition $(*)$.

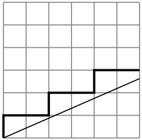


w_α

Schema of proof: using the simulation of effective subshifts:

Reminder: every effective one-dimensional subshift can be simulated by a bidimensional SFT.

Take $\alpha \in [0,1[$ with condition $(*)$.



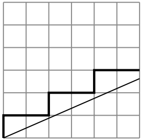
w_α

$X_\alpha = \overline{\{\sigma^n(w_\alpha) \ : \ n \geq 0\}}$:

Schema of proof: using the simulation of effective subshifts:

Reminder: every effective one-dimensional subshift can be simulated by a bidimensional SFT.

Take $\alpha \in [0,1[$ with condition $(*)$.



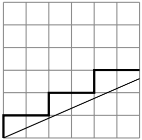
w_α

$X_\alpha = \overline{\{\sigma^n(w_\alpha) \ : \ n \geq 0\}}$: $N_n(X_\alpha) = (n+1)$.

Schema of proof: using the simulation of effective subshifts:

Reminder: every effective one-dimensional subshift can be simulated by a bidimensional SFT.

Take $\alpha \in [0,1[$ with condition $(*)$.



w_α

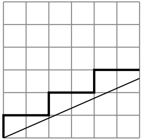
$X_\alpha = \overline{\{\sigma^n(w_\alpha) \ : \ n \geq 0\}}$: $N_n(X_\alpha) = (n+1)$.

Adding random bits on the 1 symbols: upper bound $(n+1)2^{\lceil \alpha \cdot n \rceil} \leq (n+1)2^{\alpha n+1}$.

Schema of proof: using the simulation of effective subshifts:

Reminder: every effective one-dimensional subshift can be simulated by a bidimensional SFT.

Take $\alpha \in [0,1[$ with condition $(*)$.



w_α

$X_\alpha = \overline{\{\sigma^n(w_\alpha) \ : \ n \geq 0\}}$: $N_n(X_\alpha) = (n+1)$.

Adding random bits on the 1 symbols: upper bound $(n+1)2^{\lceil \alpha \cdot n \rceil} \leq (n+1)2^{\alpha n + 1}$.

Stacking copies of this subshift: $\leq (n+1)2^{\alpha n^2 + n}$ (upgradable).

The question of the limit between computability and uncomputability.

1. Two "computability regimes": w/wo block gluing. How to characterize the limit ?

2. What are the exact conditions under which uncomputability phenomena can appear ?

Quantification of block gluing

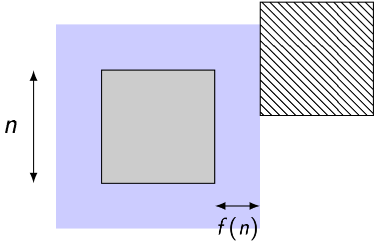
Consider a function $f : \mathbb{N} \to \mathbb{N}$, non-decreasing and computable.

Quantification of block gluing

Consider a function $f : \mathbb{N} \to \mathbb{N}$, non-decreasing and computable.
A subshift is f-block gluing when:

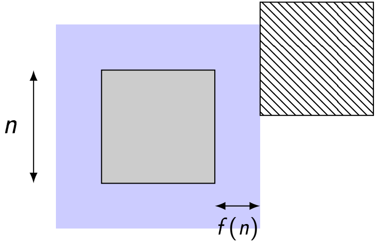
Quantification of block gluing

Consider a function $f : \mathbb{N} \to \mathbb{N}$, non-decreasing and computable.
A subshift is f-block gluing when:



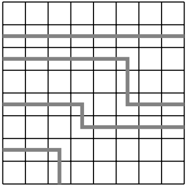
Quantification of block gluing

Consider a function $f : \mathbb{N} \to \mathbb{N}$, non-decreasing and computable. A subshift is f-block gluing when:

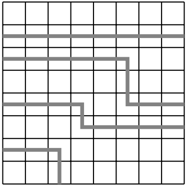


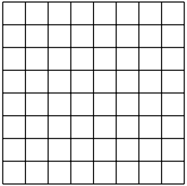
Question: for which functions f the entropy (resp. language) is forced to be computable (resp. decidable) ?

An example of linear block gluing SFT

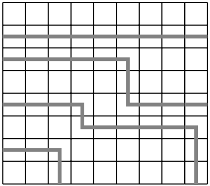


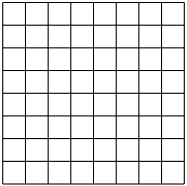
An example of linear block gluing SFT



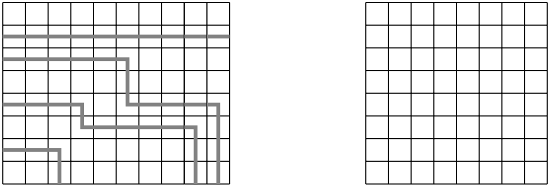


An example of linear block gluing SFT

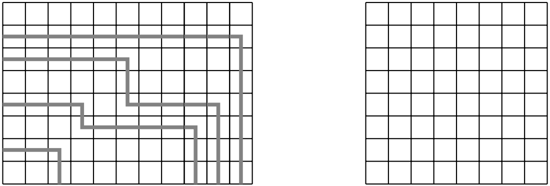




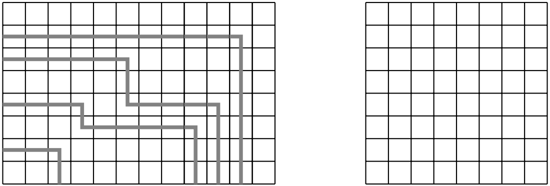
An example of linear block gluing SFT



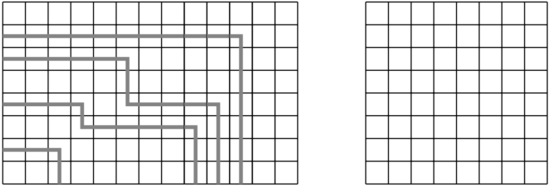
An example of linear block gluing SFT



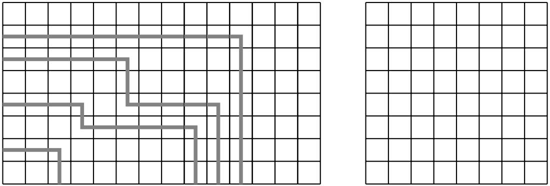
An example of linear block gluing SFT



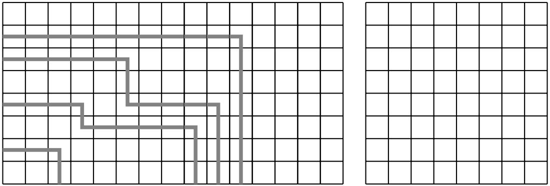
An example of linear block gluing SFT



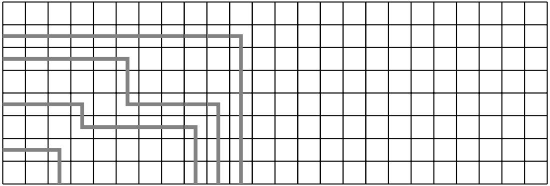
An example of linear block gluing SFT



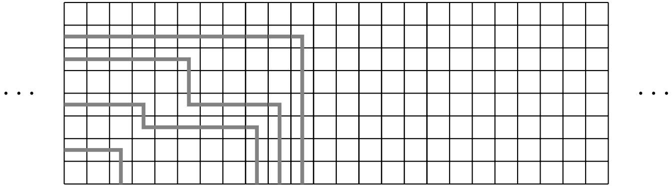
An example of linear block gluing SFT



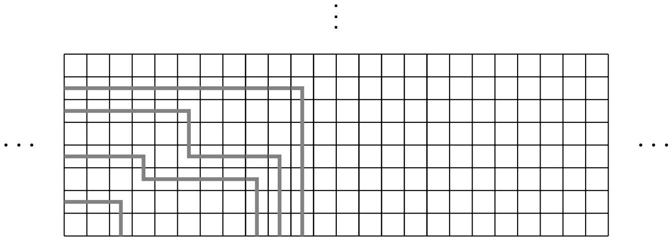
An example of linear block gluing SFT



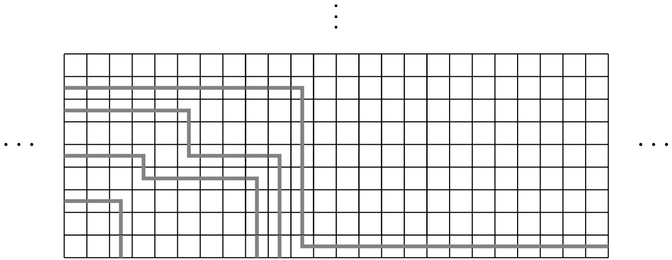
An example of linear block gluing SFT



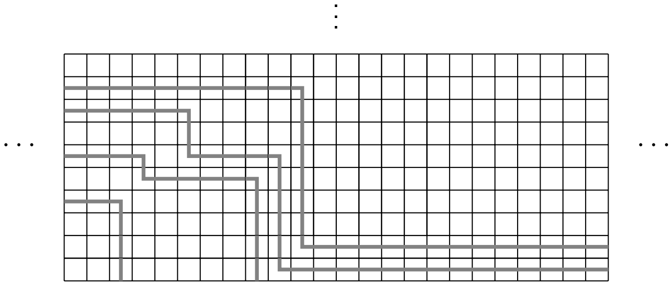
An example of linear block gluing SFT



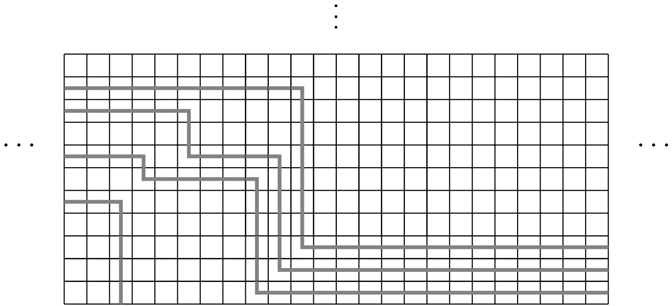
An example of linear block gluing SFT



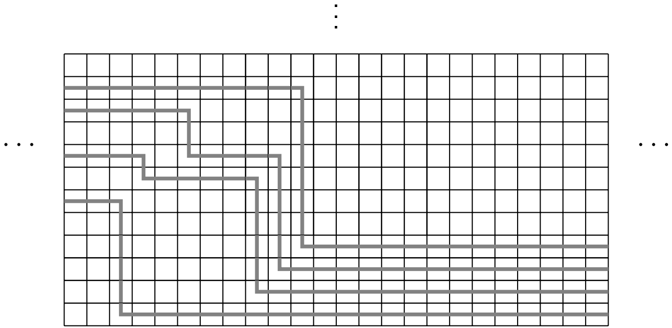
An example of linear block gluing SFT



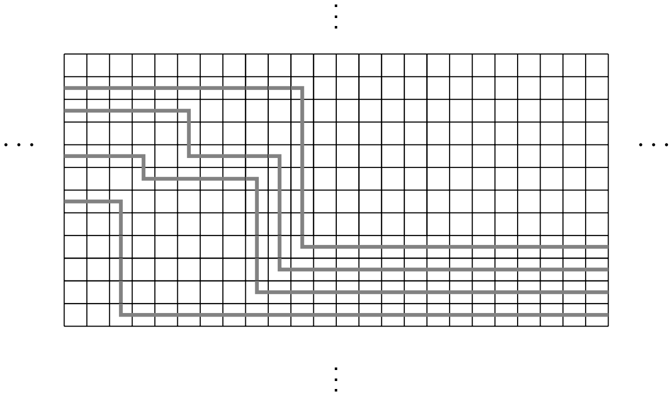
An example of linear block gluing SFT



An example of linear block gluing SFT

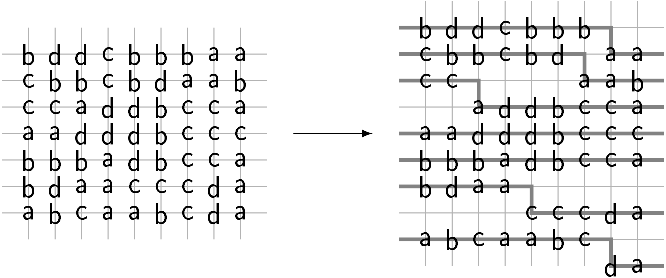


An example of linear block gluing SFT



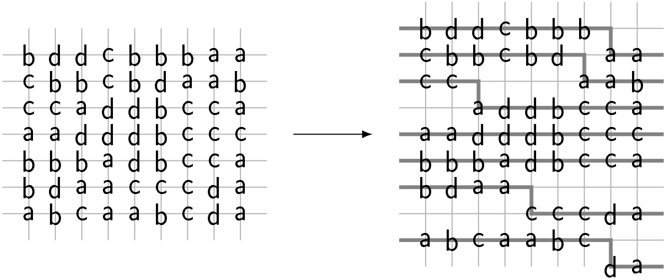
Distortion operator

Transformation on configurations of X subshift on alphabet $\{a, b, c, d\}$:



Distortion operator

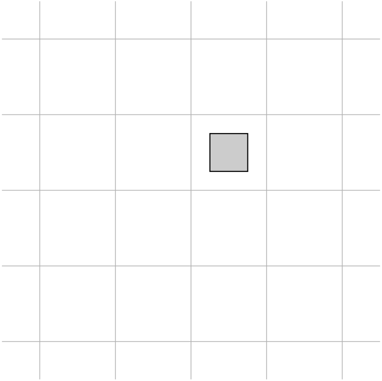
Transformation on configurations of X subshift on alphabet $\{a, b, c, d\}$:



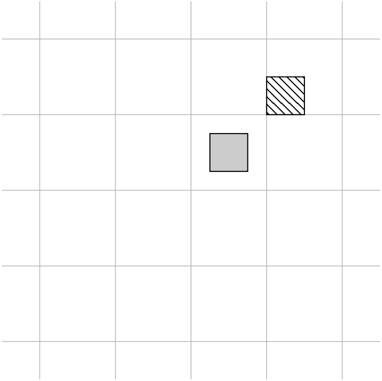
Induces an operator $\mathcal{T}$ on subshifts.

Gluing on sublattices

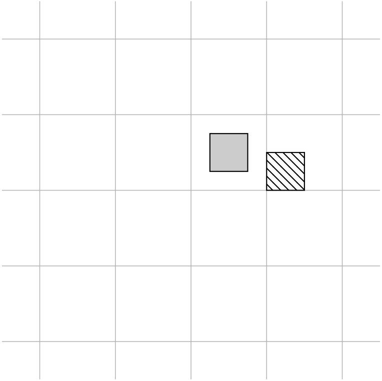
Gluing on sublattices



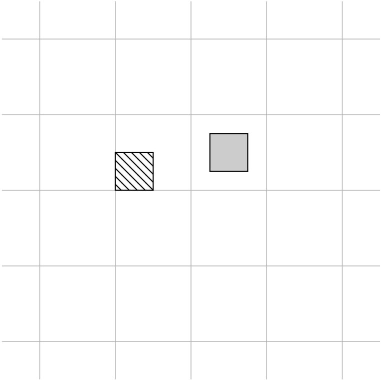
Gluing on sublattices



Gluing on sublattices



Gluing on sublattices



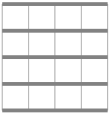
Distortion and block gluing

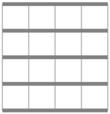
Theorem: if X has linear block gluing on sublattices, $\mathcal{T}(X)$ has linear block gluing on infinite vertical stripes separated by $O(n)$ columns.

Distortion and block gluing

Theorem: if X has linear block gluing on sublattices, $\mathcal{T}(X)$ has linear block gluing on infinite vertical stripes separated by $O(n)$ columns.

Proof:

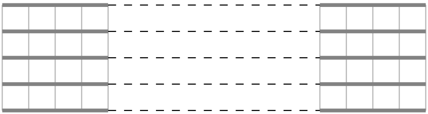




Distortion and block gluing

Theorem: if X has linear block gluing on sublattices, $\mathcal{T}(X)$ has linear block gluing on infinite vertical stripes separated by $O(n)$ columns.

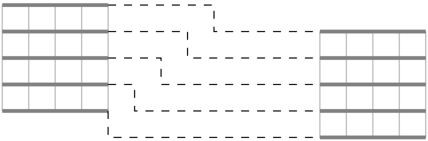
Proof:



Distortion and block gluing

Theorem: if X has linear block gluing on sublattices, $\mathcal{T}(X)$ has linear block gluing on infinite vertical stripes separated by $O(n)$ columns.

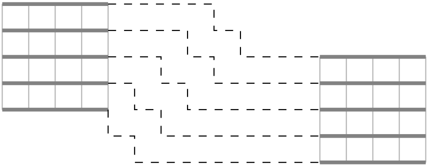
Proof:



Distortion and block gluing

Theorem: if X has linear block gluing on sublattices, $\mathcal{T}(X)$ has linear block gluing on infinite vertical stripes separated by $O(n)$ columns.

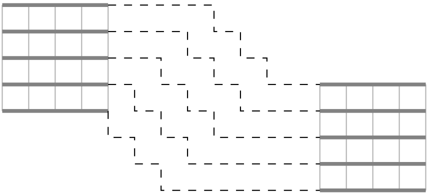
Proof:



Distortion and block gluing

Theorem: if X has linear block gluing on sublattices, $\mathcal{T}(X)$ has linear block gluing on infinite vertical stripes separated by $O(n)$ columns.

Proof:



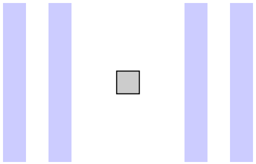
Distortion and aperiodicity

Set of possible gluing positions:

Distortion and aperiodicity

Set of possible gluing positions:

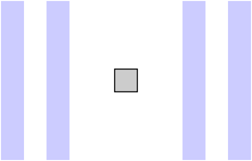
After one distortion:



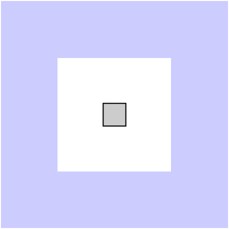
Distortion and aperiodicity

Set of possible gluing positions:

After one distortion:



Then a horizontal version of the operator:



Distortion and aperiodicity

Theorem: if X is aperiodic, then $\mathcal{T}(X)$ is aperiodic.

Distortion and aperiodicity

Theorem: if X is aperiodic, then $\mathcal{T}(X)$ is aperiodic.

Idea of the proof: if $\mathcal{T}(X)$ has a periodic configuration, the "distortion configuration" is also periodic.

Distortion and aperiodicity

Theorem: if X is aperiodic, then $\mathcal{T}(X)$ is aperiodic.

Idea of the proof: if $\mathcal{T}(X)$ has a periodic configuration, the "distortion configuration" is also periodic.

By straightening the curves, one construct a periodic configuration of X.

Distortion and aperiodicity

Theorem: if X is aperiodic, then $\mathcal{T}(X)$ is aperiodic.

Idea of the proof: if $\mathcal{T}(X)$ has a periodic configuration, the "distortion configuration" is also periodic.

By straightening the curves, one construct a periodic configuration of X.

Consequence: there exists an aperiodic linear block gluing bidimensional SFT.

Distortion and aperiodicity

Theorem: if X is aperiodic, then $\mathcal{T}(X)$ is aperiodic.

Idea of the proof: if $\mathcal{T}(X)$ has a periodic configuration, the "distortion configuration" is also periodic.

By straightening the curves, one construct a periodic configuration of X.

Consequence: there exists an aperiodic linear block gluing bidimensional SFT.

Proof: apply the distortion operator on Robinson tilings.

Distortion and aperiodicity

Theorem: if X is aperiodic, then $\mathcal{T}(X)$ is aperiodic.

Idea of the proof: if $\mathcal{T}(X)$ has a periodic configuration, the "distortion configuration" is also periodic.

By straightening the curves, one construct a periodic configuration of X.

Consequence: there exists an aperiodic linear block gluing bidimensional SFT.

Proof: apply the distortion operator on Robinson tilings.

Also: there exists a bidimensional linear block gluing SFT with undecidable language.

Regime transition for bidimensional SFT: entropy

Theorem[G.,Sablik]: for all $d \geq 2$, the possible values of entropy for linear block gluing d-dimensional SFT are the Π_1-computable non-negative real numbers.

Regime transition for bidimensional SFT: entropy

Theorem[G.,Sablik]: for all $d \geq 2$, the possible values of entropy for linear block gluing d-dimensional SFT are the Π_1-computable non-negative real numbers.

Theorem[G.,Sablik]: For all f such that $f(n) = o(\log(n))$, the entropy of any f-block gluing bidimensional SFT is computable.

Regime transition for bidimensional SFT: entropy

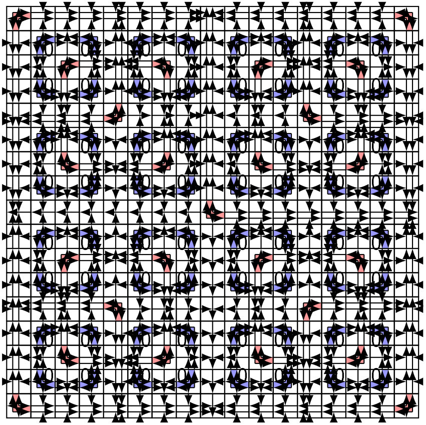
Theorem[G.,Sablik]: for all $d \geq 2$, the possible values of entropy for linear block gluing d-dimensional SFT are the Π_1-computable non-negative real numbers.

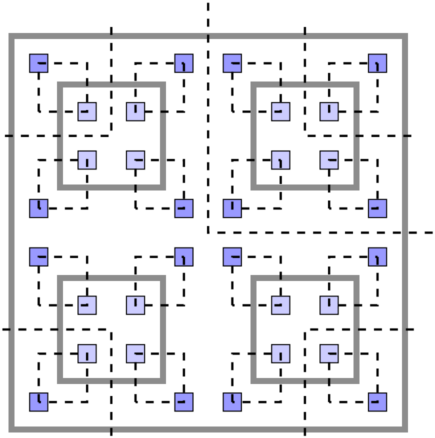
Theorem[G.,Sablik]: For all f such that $f(n) = o(\log(n))$, the entropy of any f-block gluing bidimensional SFT is computable.

See *Quantified block gluing, aperiodicity and entropy for multidimensional SFT*, **S.Gangloff, M.Sablik**.

Proof of the characterization: first layers reminder

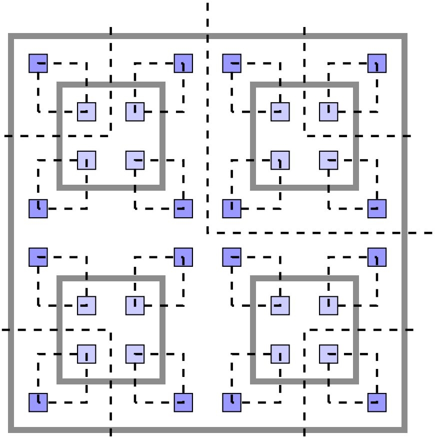
Hierarchical structures and computation areas:





Proof of the characterization: first layers reminder

Implementation of computing machines in hierarchical structures:

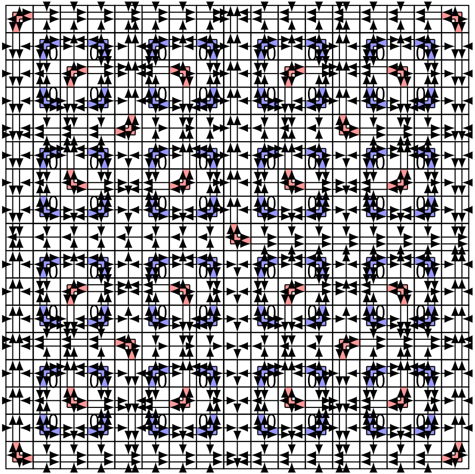


a'_0	a'_1	a'_2	$q^h_{a_3}$	a_4	a_5	a_6	a_7
a'_0	a'_1	a'_2	$q^h_{a_3}$	a_4	a_5	a_6	a_7
a'_0	a'_1	$q^2_{a_2}$	a_3	a_4	a_5	a_6	a_7
a'_0	$q^1_{a_1}$	a_2	a_3	a_4	a_5	a_6	a_7
$q^0_{a_0}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7

a'_0	a'_1	a'_2	$q^h_{a_3}$	a_4	a_5	a_6	a_7
a'_0	a'_1	a'_2	$q^h_{a_3}$	a_4	a_5	a_6	a_7
a'_0	a'_1	$q^2_{a_2}$	a_3	a_4	a_5	a_6	a_7
a'_0	$q^1_{a_1}$	a_2	a_3	a_4	a_5	a_6	a_7
$q^0_{a_0}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7

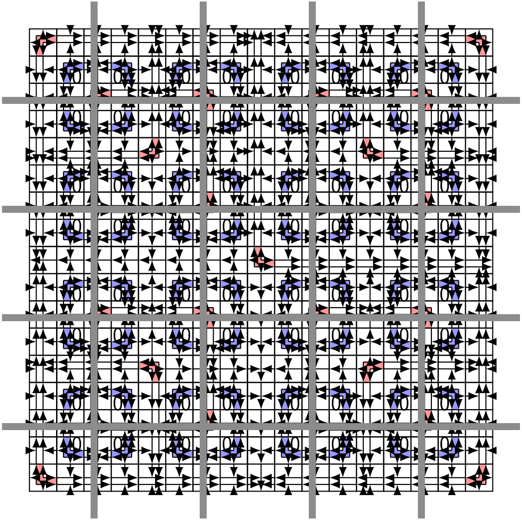
Proof of the characterization: first layers reminder

Control sets:



Proof of the characterization: first layers reminder

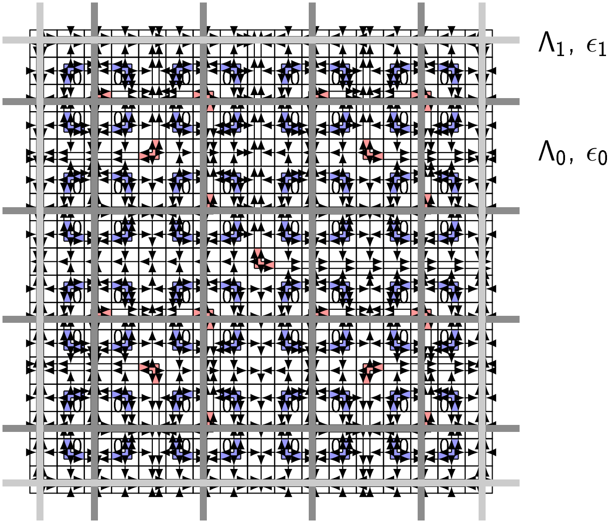
Control sets:



$\Lambda_0,\ \epsilon_0$

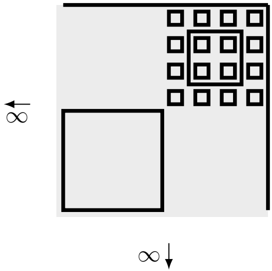
Proof of the characterization: first layers reminder

Control sets:



Obstacles to linear block gluing

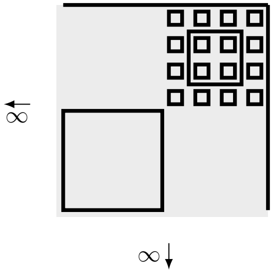
1. **Degenerated behavior** of the computing machines (infinite computation areas, without initialisation).
2. **Rigidity** of hierarchical structures.



Solutions:

Obstacles to linear block gluing

1. **Degenerated behavior** of the computing machines (infinite computation areas, without initialisation).
2. **Rigidity** of hierarchical structures.

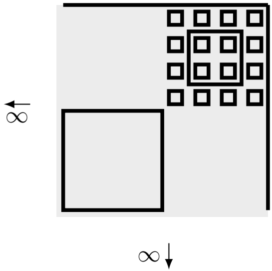


Solutions:

1. **Simulate** degenerated behaviors of the machines everywhere.

Obstacles to linear block gluing

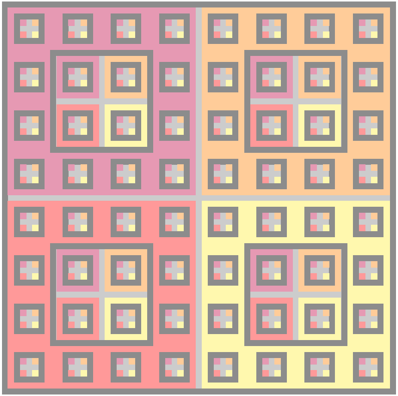
1. **Degenerated behavior** of the computing machines (infinite computation areas, without initialisation).
2. **Rigidity** of hierarchical structures.



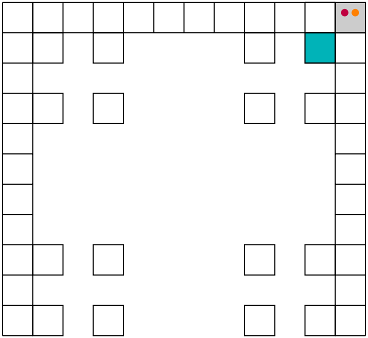
Solutions:

1. **Simulate** degenerated behaviors of the machines everywhere.
2. Use a **distortion operator** to render the structure more *flexible*.

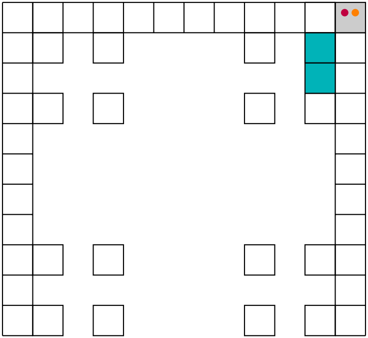
Simulating degenerated behavior: functional subdivision



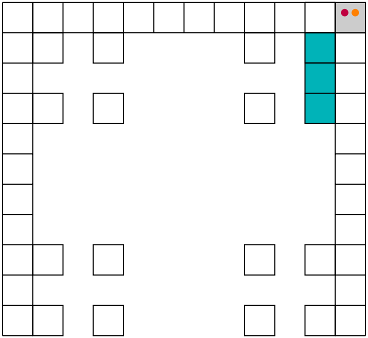
Simulating degenerated behavior: error signals



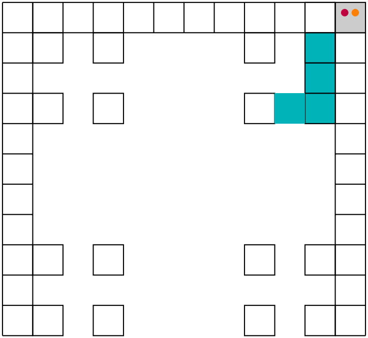
Simulating degenerated behavior: error signals



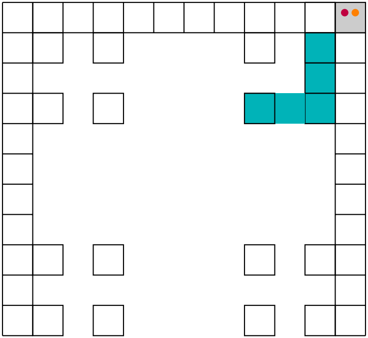
Simulating degenerated behavior: error signals



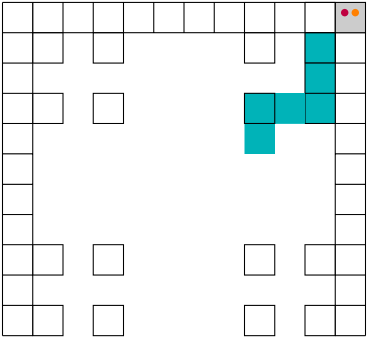
Simulating degenerated behavior: error signals



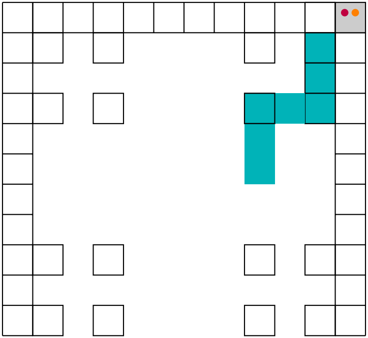
Simulating degenerated behavior: error signals



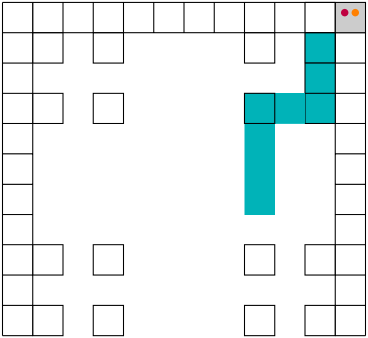
Simulating degenerated behavior: error signals



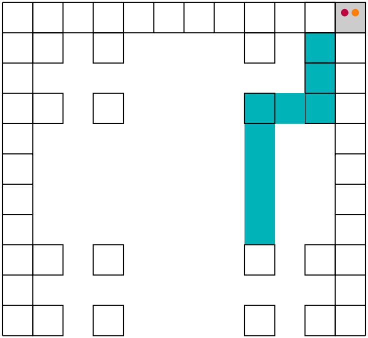
Simulating degenerated behavior: error signals



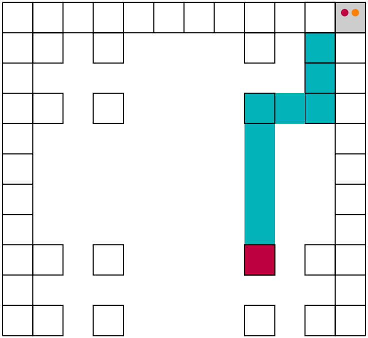
Simulating degenerated behavior: error signals



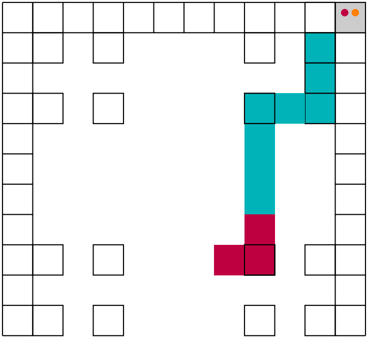
Simulating degenerated behavior: error signals



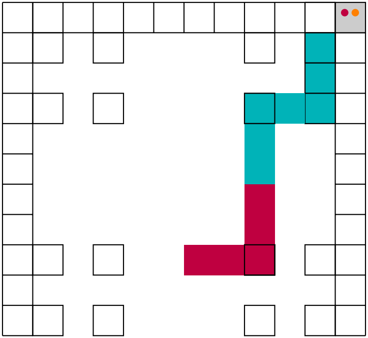
Simulating degenerated behavior: error signals



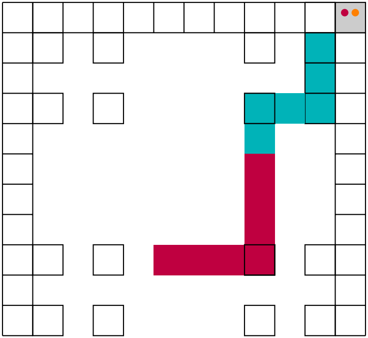
Simulating degenerated behavior: error signals



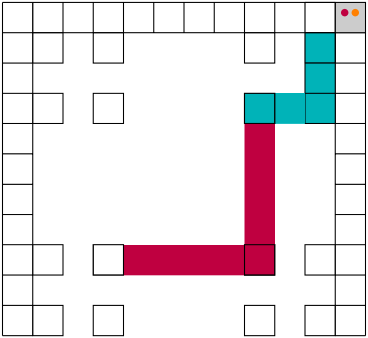
Simulating degenerated behavior: error signals



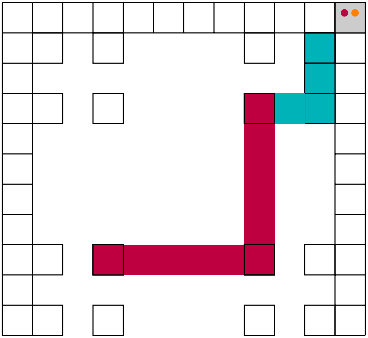
Simulating degenerated behavior: error signals



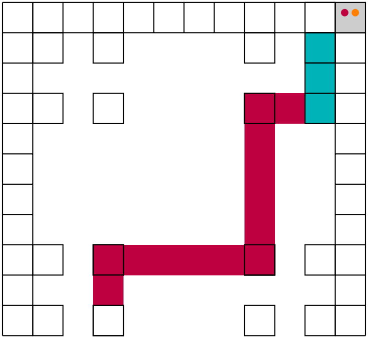
Simulating degenerated behavior: error signals



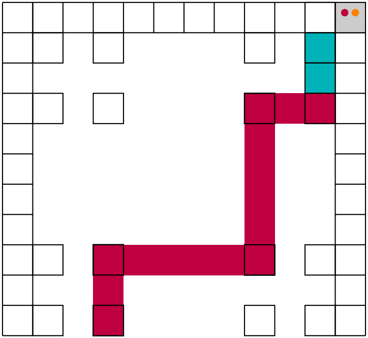
Simulating degenerated behavior: error signals



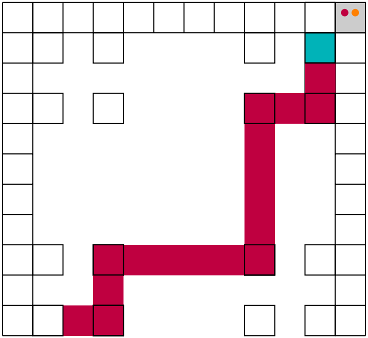
Simulating degenerated behavior: error signals



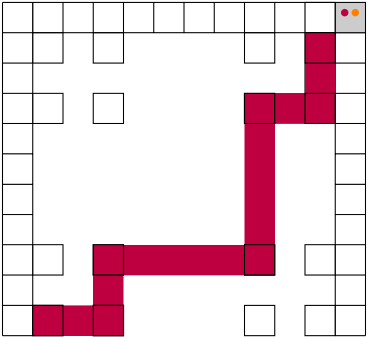
Simulating degenerated behavior: error signals



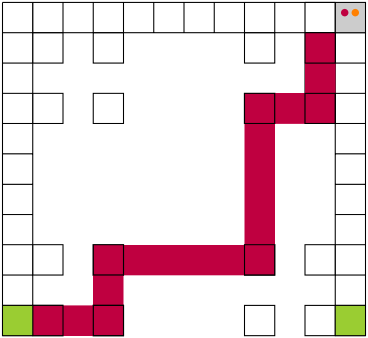
Simulating degenerated behavior: error signals



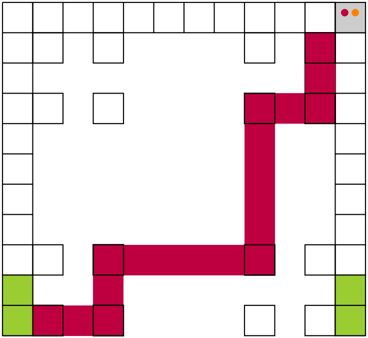
Simulating degenerated behavior: error signals



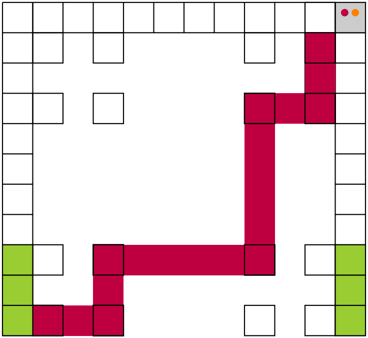
Simulating degenerated behavior: error signals



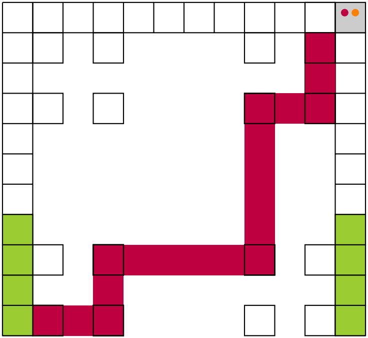
Simulating degenerated behavior: error signals



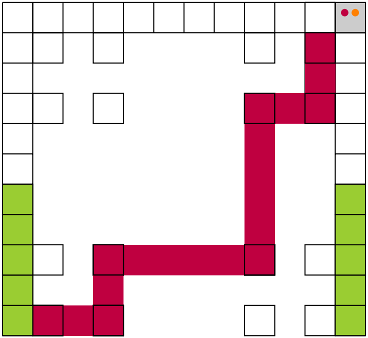
Simulating degenerated behavior: error signals



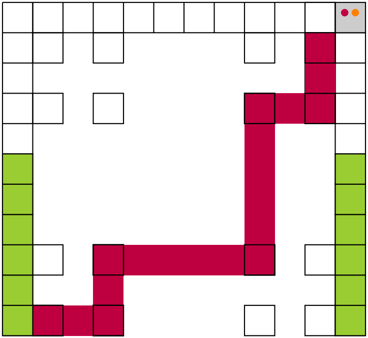
Simulating degenerated behavior: error signals



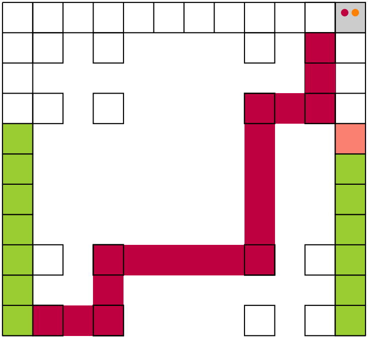
Simulating degenerated behavior: error signals



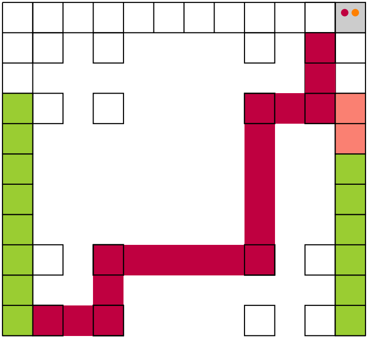
Simulating degenerated behavior: error signals



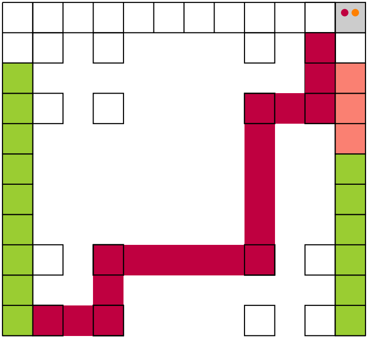
Simulating degenerated behavior: error signals



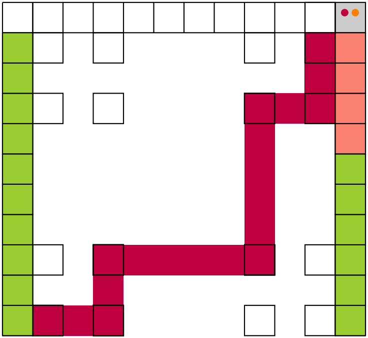
Simulating degenerated behavior: error signals



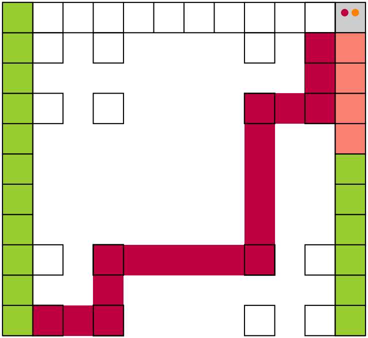
Simulating degenerated behavior: error signals



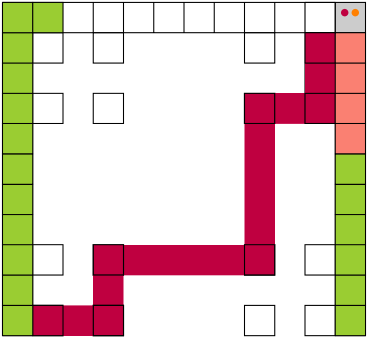
Simulating degenerated behavior: error signals



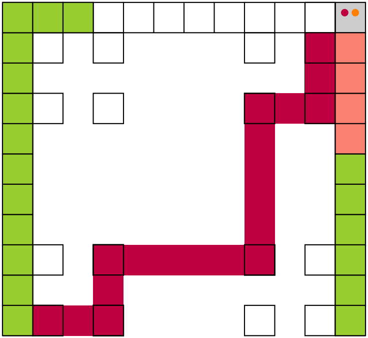
Simulating degenerated behavior: error signals



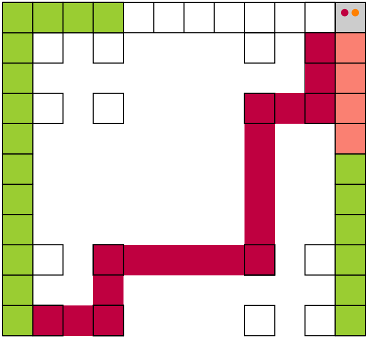
Simulating degenerated behavior: error signals



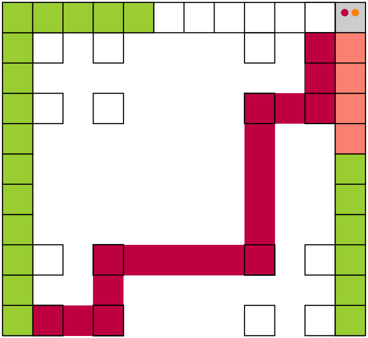
Simulating degenerated behavior: error signals



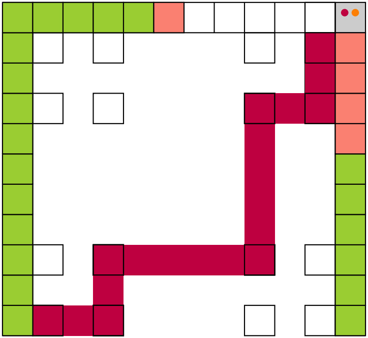
Simulating degenerated behavior: error signals



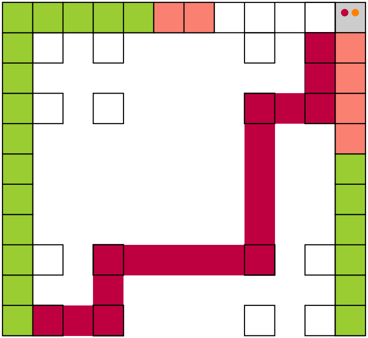
Simulating degenerated behavior: error signals



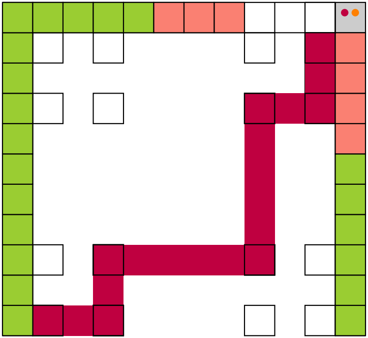
Simulating degenerated behavior: error signals



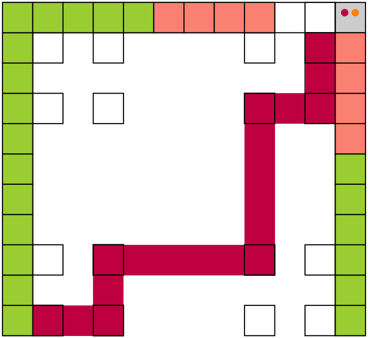
Simulating degenerated behavior: error signals



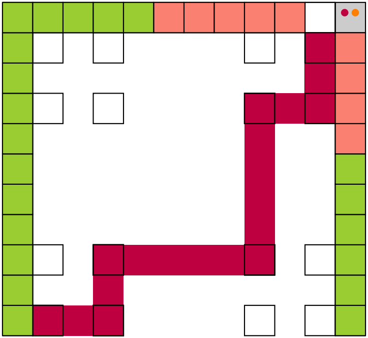
Simulating degenerated behavior: error signals



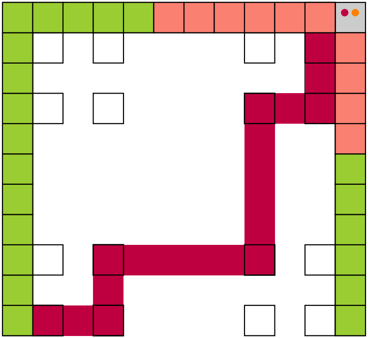
Simulating degenerated behavior: error signals



Simulating degenerated behavior: error signals

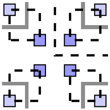


Simulating degenerated behavior: error signals



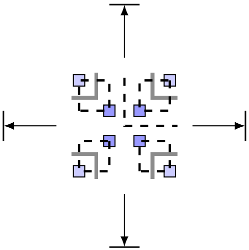
Linear block gluing: completing patterns

Completing a pattern p:



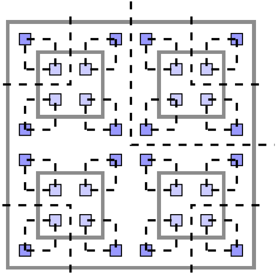
Linear block gluing: completing patterns

Completing a pattern p:



Linear block gluing: completing patterns

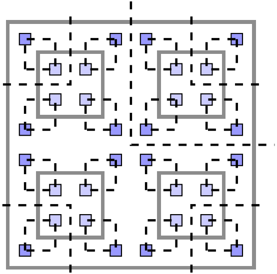
Completing a pattern p:



Two cases:

Linear block gluing: completing patterns

Completing a pattern p:

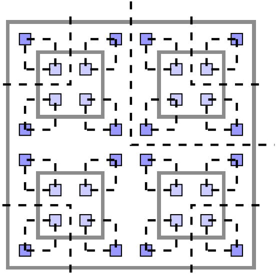


Two cases:

1. Encoding part $\in p$: complete machine layer according to it.

Linear block gluing: completing patterns

Completing a pattern p:

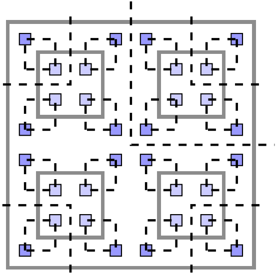


Two cases:

1. Encoding part $\in p$: complete machine layer according to it.
2. Encoding part $\notin p$: choose encoding parts on the opposite side.

Linear block gluing: completing patterns

Completing a pattern p:



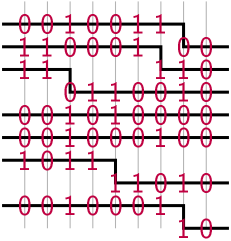
Two cases:

1. Encoding part $\in p$: complete machine layer according to it.
2. Encoding part $\notin p$: choose encoding parts on the opposite side.

As a consequence, the construction has gluing property on sublattices.

Distortion operator and random bits

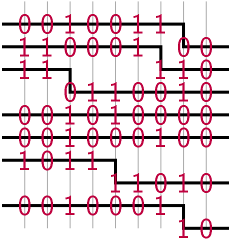
Adding random bits $\rightarrow$ operator $\mathcal{T}'$:



With this we have $h(\mathcal{T}'(X)) = h(X) + 1$.

Distortion operator and random bits

Adding random bits $\rightarrow$ operator $\mathcal{T}'$:

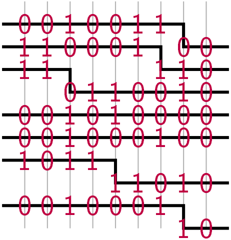


With this we have $h(\mathcal{T}'(X)) = h(X) + 1$. Rigid segments of fixed

length r:

Distortion operator and random bits

Adding random bits $\to$ operator $\mathcal{T}'$:

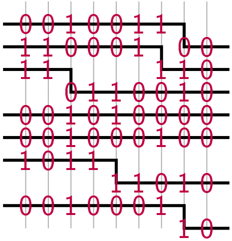


With this we have $h(\mathcal{T}'(X)) = h(X) + 1$. Rigid segments of fixed length r:

,

Distortion operator and random bits

Adding random bits $\to$ operator $\mathcal{T}'$:



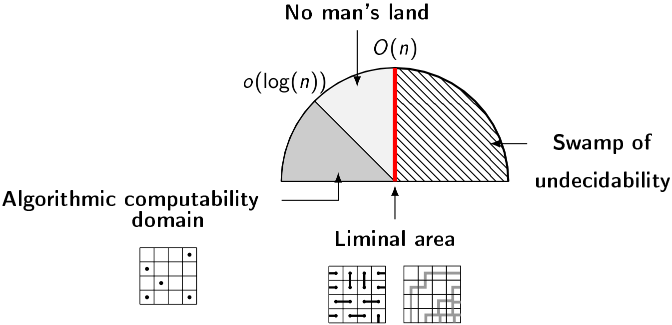
With this we have $h(\mathcal{T}'(X)) = h(X) + 1$. Rigid segments of fixed length r:


,

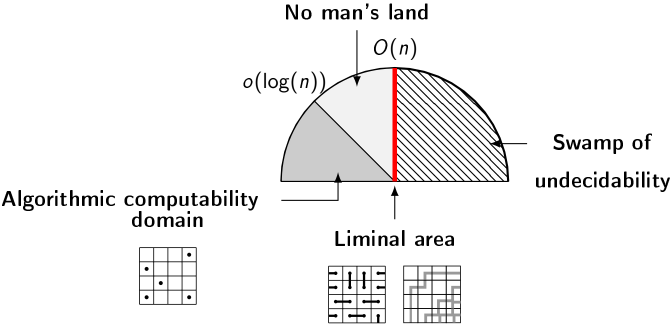
we have:

$$h(\mathcal{T}_r'(X)) = h(X) + \frac{\log(1+r)}{r}$$

Abstract of the results:



Abstract of the results:



Question: how to discriminate subshift of finite type with computable entropy/non-computable entropy on the liminal area ?

The question of intermediate gap functions

Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

The question of intermediate gap functions

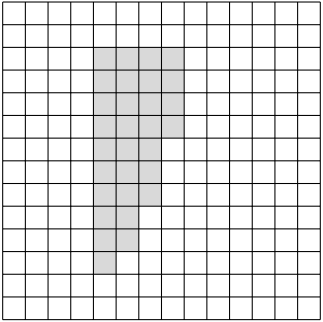
Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

Natural idea for $f(n) = \sqrt{n}$ (fails):

The question of intermediate gap functions

Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

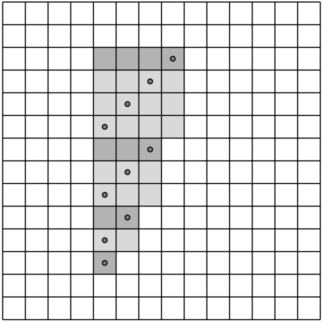
Natural idea for $f(n) = \sqrt{n}$ (fails):



The question of intermediate gap functions

Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

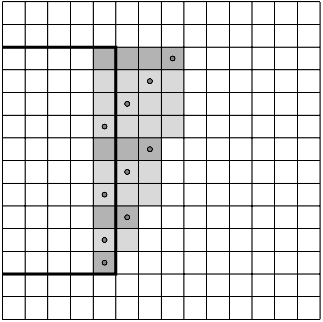
Natural idea for $f(n) = \sqrt{n}$ (fails):



The question of intermediate gap functions

Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

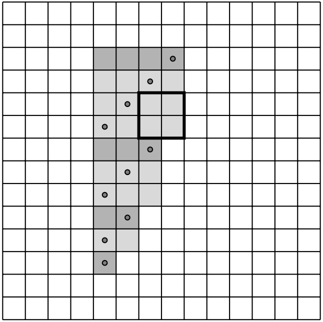
Natural idea for $f(n) = \sqrt{n}$ (fails):



The question of intermediate gap functions

Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

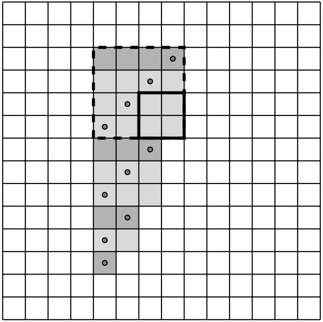
Natural idea for $f(n) = \sqrt{n}$ (fails):



The question of intermediate gap functions

Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

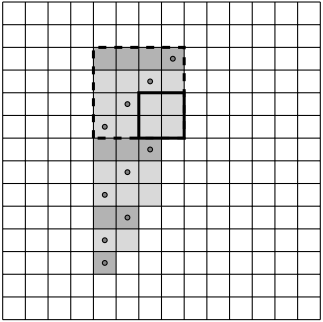
Natural idea for $f(n) = \sqrt{n}$ (fails):



The question of intermediate gap functions

Question[G.,Sablik, also related by M.Hochman]: does there exist some f-block gluing subshift with undecidable language and such that $\log(n) << f(n) = o(n)$?

Natural idea for $f(n) = \sqrt{n}$ (fails):



Problems: it is actually linear block gluing.

Universal computation in minimal multidimensional SFT

Reminder: entropy dimension (when exists):

$$D(X) = \lim_n \frac{\log(N_n(X))}{\log(n)}.$$

Universal computation in minimal multidimensional SFT

Reminder: entropy dimension (when exists):

$$D(X) = \lim_{n} \frac{\log(N_n(X))}{\log(n)}.$$

Minimal subshift: every pattern in $\mathcal{L}(X)$ appears in every configuration of X.

Universal computation in minimal multidimensional SFT

Reminder: entropy dimension (when exists):

$$D(X) = \lim_n \frac{\log(N_n(X))}{\log(n)}.$$

Minimal subshift: every pattern in $\mathcal{L}(X)$ appears in every configuration of X.

Theorem[G.,Sablik]: the values of entropy dimension on minimal tridimensional SFT are the numbers in $[0, 2] \cap \Delta_2$.

Universal computation in minimal multidimensional SFT

Reminder: entropy dimension (when exists):

$$D(X) = \lim_n \frac{\log(N_n(X))}{\log(n)}.$$

Minimal subshift: every pattern in $\mathcal{L}(X)$ appears in every configuration of X.

Theorem[G.,Sablik]: the values of entropy dimension on minimal tridimensional SFT are the numbers in $[0, 2] \cap \Delta_2$.

Ideas of proof: **1**. Computing machines control sparse random bits.

Universal computation in minimal multidimensional SFT

Reminder: entropy dimension (when exists):

$$D(X) = \lim_n \frac{\log(N_n(X))}{\log(n)}.$$

Minimal subshift: every pattern in $\mathcal{L}(X)$ appears in every configuration of X.

Theorem[G.,Sablik]: the values of entropy dimension on minimal tridimensional SFT are the numbers in $[0, 2] \cap \Delta_2$.

Ideas of proof: **1**. Computing machines control sparse random bits.

2. Counters alternative all possible behaviors.

Universal computation in minimal multidimensional SFT

Reminder: entropy dimension (when exists):

$$D(X) = \lim_n \frac{\log(N_n(X))}{\log(n)}.$$

Minimal subshift: every pattern in $\mathcal{L}(X)$ appears in every configuration of X.

Theorem[G.,Sablik]: the values of entropy dimension on minimal tridimensional SFT are the numbers in $[0, 2] \cap \Delta_2$.

Ideas of proof: **1**. Computing machines control sparse random bits.

2. Counters alternative all possible behaviors.

3. Using Fermat numbers $2^{2^n} + 1$ as periods (co-primes, encoding respecting minimality).

Some pre-formal investigations

- $\triangleright$ As we impose dynamical constraints: $\uparrow$ in complexity

Some pre-formal investigations

$\triangleright$ As we impose dynamical constraints: $\uparrow$ in complexity $\neq$ algorithmic complexity.

Some pre-formal investigations

$\triangleright$ As we impose dynamical constraints: $\uparrow$ in complexity $\neq$ algorithmic complexity.

How to formalise it ?

Some pre-formal investigations

$\triangleright$ As we impose dynamical constraints: $\uparrow$ in complexity $\neq$ algorithmic complexity.

How to formalise it ?

How does the dynamical system work ? (for instance human brain) $\rightarrow$ identification of *objects* [structures, signals, machines,...] in the system and their relations.

Some pre-formal investigations

$\triangleright$ As we impose dynamical constraints: $\uparrow$ in complexity $\neq$ algorithmic complexity.

How to formalise it ?

How does the dynamical system work ? (for instance human brain) $\rightarrow$ identification of *objects* [structures, signals, machines,...] in the system and their relations.

Cognitively this seems to correspond to the faculty of **understanding**.

Some pre-formal investigations

$\triangleright$ As we impose dynamical constraints: $\uparrow$ in complexity $\neq$ algorithmic complexity.

How to formalise it ?

How does the dynamical system work ? (for instance human brain) $\rightarrow$ identification of *objects* [structures, signals, machines,...] in the system and their relations.

Cognitively this seems to correspond to the faculty of **understanding**.

Reverse approach: given a system, how to identify *functional parts*.

Some pre-formal investigations

Relation with the notion of *organisation*.

Some pre-formal investigations

Relation with the notion of *organisation*.

Hypothesis: optimisation between algorithmic complexity of invariants/dynamical constraints $\Rightarrow$ organisedness.

Some pre-formal investigations

Relation with the notion of *organisation*.

Hypothesis: optimisation between algorithmic complexity of invariants/dynamical constraints $\Rightarrow$ organisedness.

Formalising these phenomena $\rightarrow$ development of a formalism for:

Some pre-formal investigations

Relation with the notion of *organisation*.

Hypothesis: optimisation between algorithmic complexity of invariants/dynamical constraints $\Rightarrow$ organisedness.

Formalising these phenomena $\rightarrow$ development of a formalism for:

1. limits on information transport imposed by space-time.

Some pre-formal investigations

Relation with the notion of *organisation*.

Hypothesis: optimisation between algorithmic complexity of invariants/dynamical constraints $\Rightarrow$ organisedness.

Formalising these phenomena $\rightarrow$ development of a formalism for:

1. limits on information transport imposed by space-time.
2. how dynamical constraints prevent enforcing universal computation *in* configurations.