Hardness of Conjugacy, Embedding and Factorization in multidimensional SFTs

Emmanuel Jeandel, Pascal Vanier

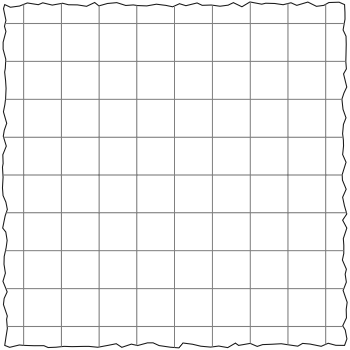
Einstein Institute of Mathematics, Hebrew University of Jerusalem
LIF, Aix-Marseille Université

Automata theory and Symbolic Dynamics Workshop

Subshifts

A **finite** alphabet :

$$\Sigma = \{\blacksquare,\blacksquare\}$$

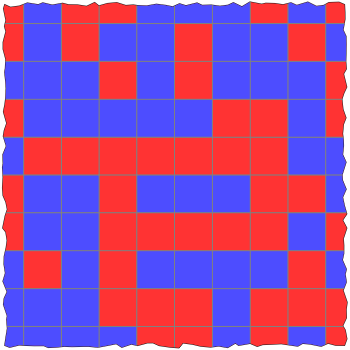


Subshifts

A **finite** alphabet :

$\Sigma = \{$  ,  $\}$

A **tiling** or **configuration**:



Subshifts

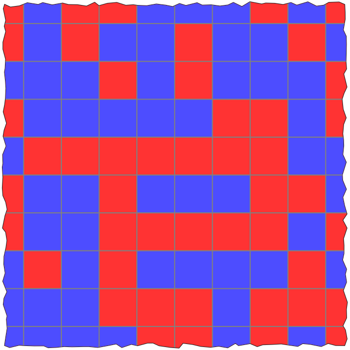
A finite alphabet :

$\Sigma = \{$  $\}$

A family of forbidden patterns:

$\mathcal{F} = \{$  $\}$

A tiling or configuration:



Subshifts

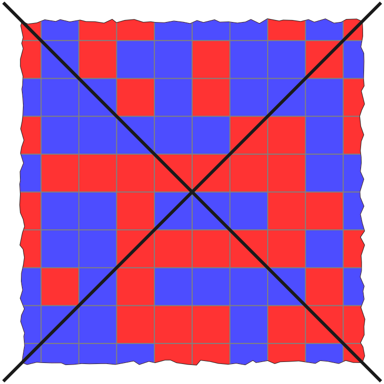
A finite alphabet :

$$\Sigma = \{$$  $$\}$$

A family of forbidden patterns:

$$\mathcal{F} = \left\{ \right.$$  $$\left. \right\}$$

A tiling or configuration:



Subshifts

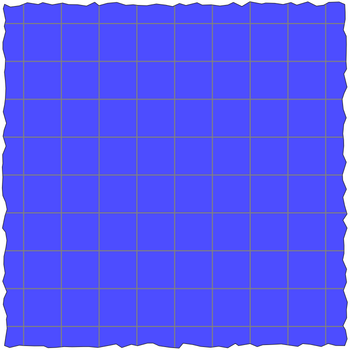
A **finite** alphabet :

$$\Sigma = \{$$  $$\}$$

A family of **forbidden patterns**:

$$\mathcal{F} = \left\{ \right.$$  $$\left. \right\}$$

A **tiling** or **configuration**:



Subshifts

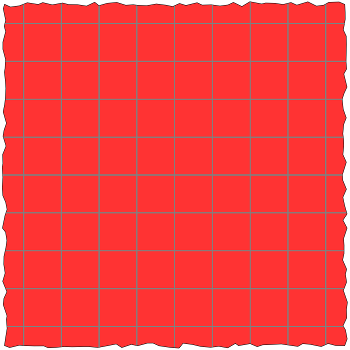
A **finite** alphabet :

$$\Sigma = \{ \quad \}$$

A family of **forbidden patterns**:

$$\mathcal{F} = \left\{ \quad \right\}$$

A **tiling** or **configuration**:



Subshifts

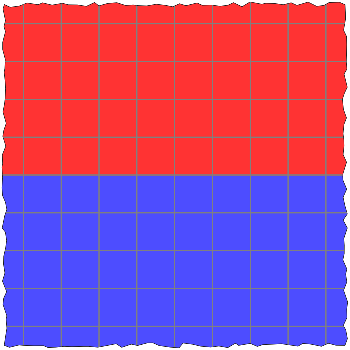
A **finite** alphabet :

$$\Sigma = \{ \ \}$$

A family of **forbidden patterns**:

$$\mathcal{F} = \left\{ \ \right\}$$

A **tiling** or **configuration**:



Subshifts

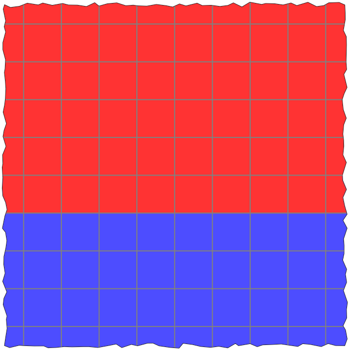
A **finite** alphabet :

$$\Sigma = \{$$  $$\}$$

A family of **forbidden patterns**:

$$\mathcal{F} = \Big\{$$  $$\Big\}$$

A **tiling** or **configuration**:



Subshifts

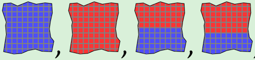
A **finite** alphabet :

$$\Sigma = \{$$  $$\}$$

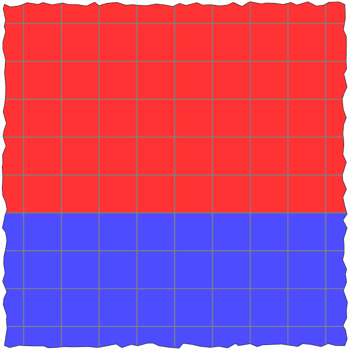
A family of **forbidden patterns**:

$$\mathcal{F} = \Big\{$$  $$\Big\}$$

Subshift: set of all configurations avoiding $\mathcal{F}$, denoted $\mathcal{X}_{\mathcal{F}}$:

$$\mathcal{X}_{\mathcal{F}} = \Big\{$$  $$, \ldots \Big\}$$

A **tiling** or **configuration**:



Subshifts

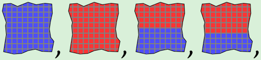
A **finite** alphabet :

$$\Sigma = \{$$  $$\}$$

A family of **forbidden patterns**:

$$\mathcal{F} = \Big\{$$  $$\Big\}$$

Subshift: set of all configurations avoiding $\mathcal{F}$, denoted $\mathcal{X}_{\mathcal{F}}$:

$$\mathcal{X}_{\mathcal{F}} = \Big\{$$  $$, \ldots \Big\}$$

The number of **forbidden patterns** may be **finite**, the generated space is then a **subshift of finite type (SFT)**.

Subshifts

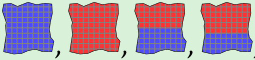
A **finite** alphabet :

$$\Sigma = \{$$  $$\}$$

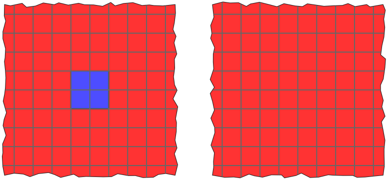
A family of **forbidden patterns**:

$$\mathcal{F} = \{$$  $$\}$$

Subshift: set of all configurations avoiding $\mathcal{F}$, denoted $\mathcal{X}_{\mathcal{F}}$:

$$\mathcal{X}_{\mathcal{F}} = \{$$  $$,\ldots \}$$

The number of **forbidden patterns** may be **finite**, the generated space is then a **subshift of finite type (SFT)**.



NOT an SFT

Block codes and sofic shifts

A **block code** is a shift invariant map defined **locally**.

block code = **continuous map**.

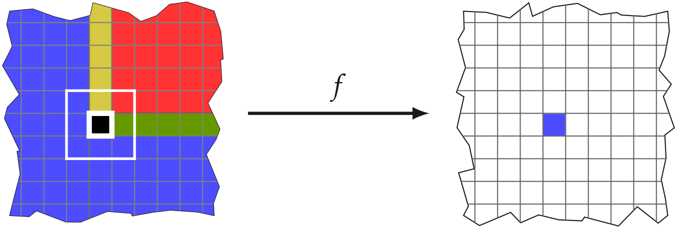
The **image of a subshift** by a block code is called a **factor**.

Factors of SFTs form the class of **sofic shifts**.

SFTs are sofic, but sofic shifts are not necessarily SFTs.

Block codes and sofic shifts

A **block code** is a shift invariant map defined **locally**.



block code = **continuous map**.

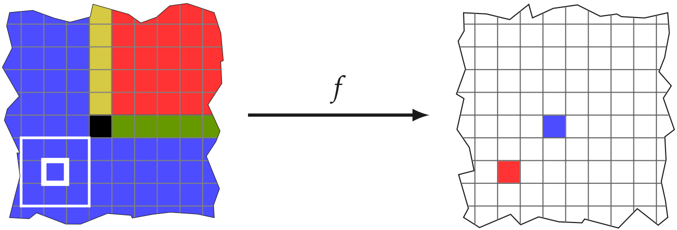
The **image of a subshift** by a block code is called a **factor**.

Factors of SFTs form the class of **sofic shifts**.

SFTs are sofic, but sofic shifts are not necessarily SFTs.

Block codes and sofic shifts

A **block code** is a shift invariant map defined **locally**.



block code = **continuous map**.

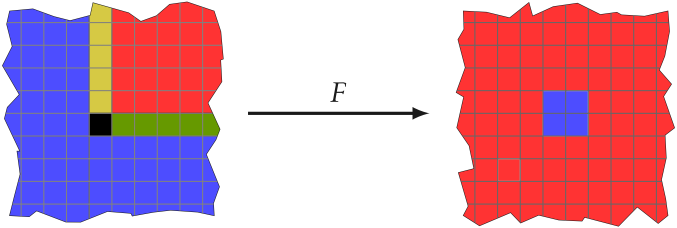
The **image of a subshift** by a block code is called a **factor**.

Factors of SFTs form the class of **sofic shifts**.

SFTs are sofic, but sofic shifts are not necessarily SFTs.

Block codes and sofic shifts

A **block code** is a shift invariant map defined **locally**.



block code = **continuous map**.

The **image of a subshift** by a block code is called a **factor**.

Factors of SFTs form the class of **sofic shifts**.

SFTs are sofic, but sofic shifts are not necessarily SFTs.

Effective shifts

An **effective subshift** is a subshift definable by a **recursively enumerable set of forbidden patterns**.

Sofic shifts are effective, but effective shifts are not necessarily sofic.

Remember Emmanuel's talk's example.

Example (1d): the forbidden patterns are the words $awawa$ for any word w and letter a, this is the Thue-Morse shift (aperiodic).

What I'm going to talk about

In this talk, we will investigate the difficulty of the relations induced by several block code types:

- Conjugacy
- Factorization
- Embedding

Don't worry, I'll (re[1])define them all!

[1] For most of you.

1. Conjugacy
2. Factorization
3. Embedding

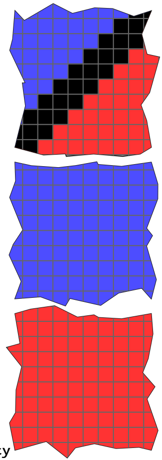
Conjugacy

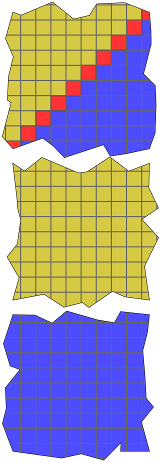
What is the “right notion” of isomorphism for subshifts ?

Conjugacy

What is the “right notion” of isomorphism for subshifts ?

Take these two subshifts:

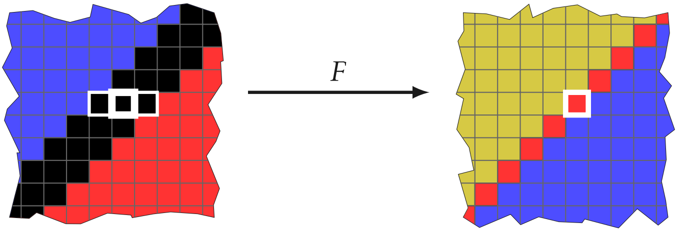




Conjugacy

What is the "right notion" of isomorphism for subshifts ?

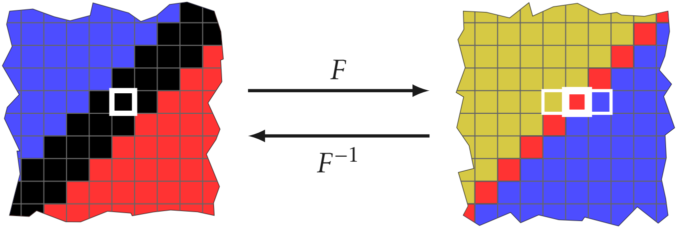
A **conjugacy** is a **bijective** block code whose **inverse is also a block code**.



Conjugacy

What is the "right notion" of isomorphism for subshifts ?

A **conjugacy** is a **bijective** block code whose **inverse is also a block code**.



(Un)Decidability of conjugacy

Can we decide whether two SFTs are conjugate?

(Un)Decidability of conjugacy

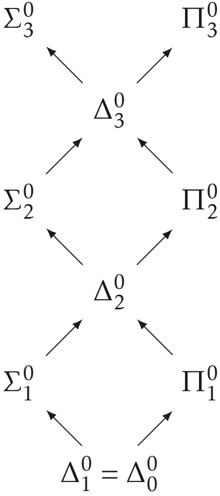
Can we decide whether two SFTs are conjugate?

- The problem is **undecidable in dimension 2**.
- The problem is **decidable** in dimension 1 **on $\mathbb{N}$**.
- The problem is **open** in dimension 1 **on $\mathbb{Z}$**.

But how hard?

Definition A problem $P \subseteq \mathbb{N}$ is Π^0_n if there exists a total Turing machine M such that

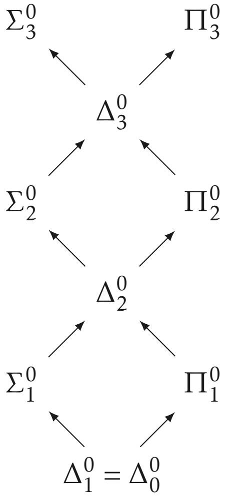
$$n \in P \Leftrightarrow \forall m_1, \exists m_2, \ldots, \Theta m_n, M(n, m_1, \ldots, m_n)$$



But how hard?

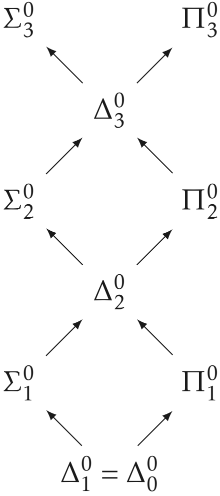
Definition A problem $P \subseteq \mathbb{N}$ is Σ_n^0 if there exists a total Turing machine M such that

$$n \in P \Leftrightarrow \exists m_1, \forall m_2, \ldots, \Theta m_n, M(n, m_1, \ldots, m_n)$$



But how hard?

- Σ_1^0: recursively enumerable
- Π_1^0: **co**-recursively enumerable
- Σ_n^0: recursively enumerable with some Π_{n-1}^0 oracle.
- Π_n^0: co-recursively enumerable with some Σ_{n-1}^0 oracle.

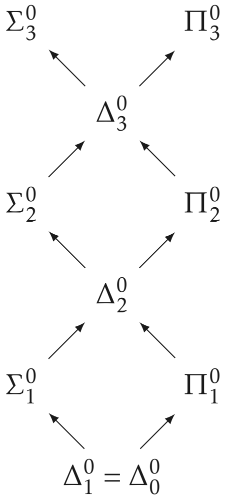


But how hard?

Reduction: $A \leq B$ iff there exists a total computable function f such that:

$$\forall x, \qquad x \in B \Leftrightarrow f(x) \in A$$

Definition A problem is **complete** if it can **solve all problems of the class**.



Some complete problems:

- Σ_1^0 : knowing if a Turing machine halts (HP)
- Π_2^0 : knowing if a Turing machine halts on all inputs (TOT)
- Σ_3^0 : knowing if the number of inputs on which a Turing machine does not halt is finite (COFIN)

Complexity of conjugacy

Theorem For any fixed SFT X, given an SFT Y deciding whether X is **conjugate** to Y is Σ_1^0**-complete**.

Remark SFTs are represented by integers

Remark Block codes are represented by integers

The inputs are these integers.

Complexity of conjugacy

Theorem For any fixed SFT X, given an SFT Y deciding whether X is **conjugate** to Y is **Σ_1^0-complete**.

Idea of the proof :

Conjugacy is Σ_1^0 :

$$\underbrace{\exists F, G, \underbrace{(F(X) \subseteq Y)}_{\Sigma_1^0} \wedge \underbrace{(G(Y) \subseteq X)}_{\Sigma_1^0} \wedge \underbrace{(F \circ G = \mathrm{id}_X)}_{\Sigma_1^0} \wedge \underbrace{(G \circ F = \mathrm{id}_Y)}_{\Sigma_1^0}}_{\Sigma_1^0}$$

- Guess two block codes F and G.
- Check if they form a conjugacy function.

Complexity of conjugacy

Theorem For any fixed SFT X, given an SFT Y deciding whether X is **conjugate** to Y is **Σ_1^0-complete**.

Idea of the proof :

Conjugacy is Σ_1^0-hard, reduction from the halting problem :

- R_M an SFT which is empty iff M halts.
- n greater than the size of the alphabet of X.

$$X \overset{?}{\cong} X \sqcup R_M \times \{0,...,n\}^{\mathbb{Z}^2}$$

- If R_M is empty, then X and $X \sqcup R_M \times \{0,\ldots,n\}^{\mathbb{Z}^2}$ are equal.
- Otherwise X and $X \sqcup R_M \times \{0,\ldots,n\}^{\mathbb{Z}^2}$ are not conjugate.

$\square$

Complexity of conjugacy (sofic & effective)

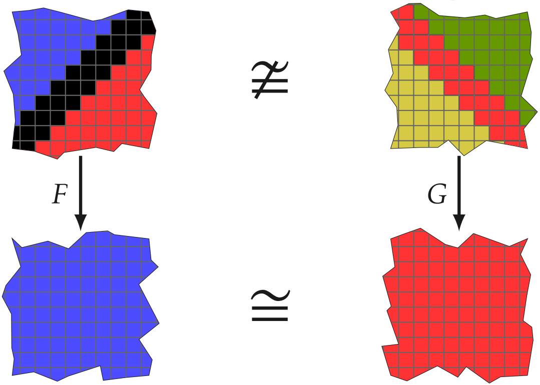
Theorem Given X, Y two **effective** (resp. **sofic** of dimension $d \geq 2$) subshifts, deciding whether X is **conjugate** to Y is Σ_3^0**-complete**.

Deciding if $F(X) \subseteq Y$ is now Π_2^0.

Complexity of conjugacy (sofic & effective)

Theorem Given X, Y two **effective** (resp. **sofic** of dimension $d \geq 2$) subshifts, deciding whether X is **conjugate** to Y is Σ_3^0**-complete**.

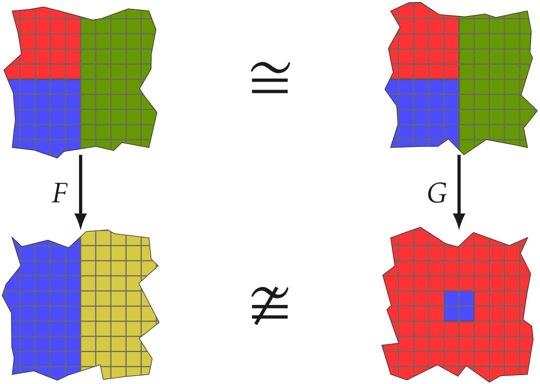
Deciding if $F(X) \subseteq Y$ is now Π_2^0.



Complexity of conjugacy (sofic & effective)

Theorem Given X, Y two **effective** (resp. **sofic** of dimension $d \geq 2$) subshifts, deciding whether X is **conjugate** to Y is **Σ_3^0-complete**.

Deciding if $F(X) \subseteq Y$ is now Π_2^0.

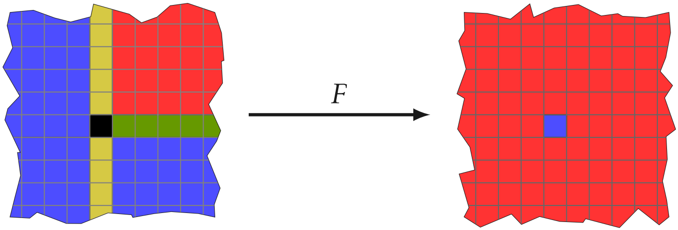


1. Conjugacy

2. Factorization

3. Embedding

Factorization



Definition A subshift Y is a factor of a subshift X, if there exists a surjective block code $F : X \to Y$.

i.e. $F(X) = Y$

Remark Factorization can be seen as a sort of simulation.

Complexity of factorization

How hard is factorization?

- At least Σ^0_1-hard:

  Factorization to the empty subshift.

- At least Π^0_1-hard:

  Factorization to the single configuration subshift.

Theorem Given two SFTs X, Y (resp. effective, sofic), deciding whether X **factorizes** onto Y is Σ^0_3**-complete**.

Upper-Bound

Theorem Factorization is Σ^0_3.

Proof scheme :

$$\exists F, \underbrace{F(X) \subseteq Y}_{\Sigma^0_1} \wedge \underbrace{Y \subseteq F(X)}_{\Pi^0_2}$$

Manipulation of logical formulae using compactness of shift spaces.

Lower-Bound

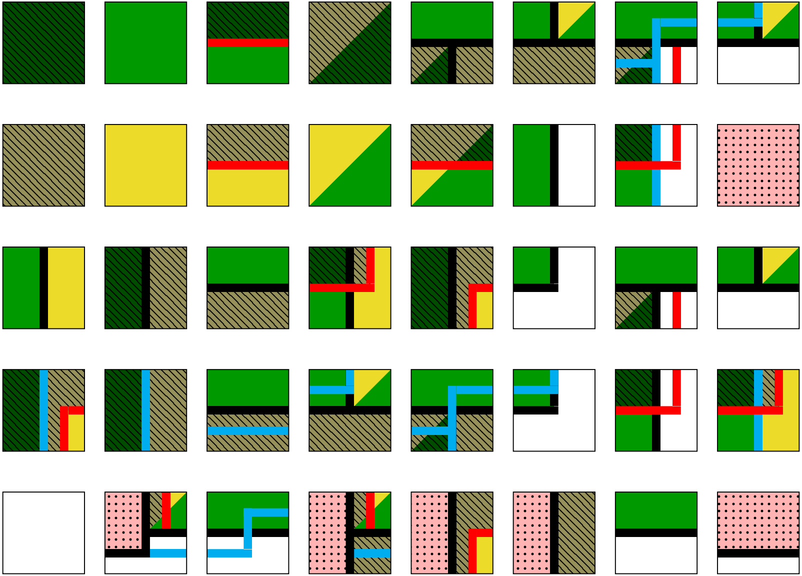
Theorem Factorization is Σ_3^0-hard.

Proof by reduction from COFIN : the set of Turing machines that run infinitely on a finite number of inputs only.

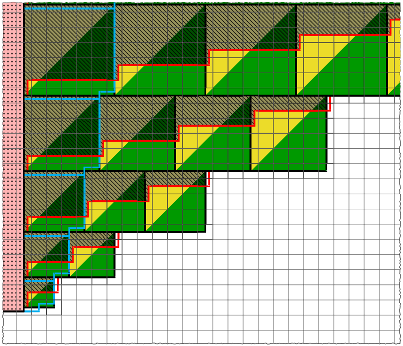
From any Turing machine M, we construct two SFTs X_M, Y_M such that X_M factors on Y_M iff $M \in$ COFIN

- We need to be able to embed some computation in X_M, Y_M
- We need some control on the structure

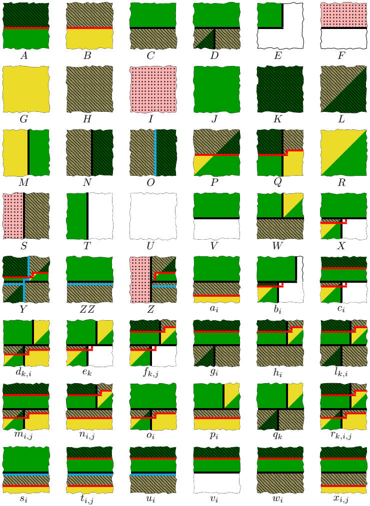
Lower-Bound : the Construction



Lower-Bound : the Construction



α-configuration



Why is such a construction interesting?

Definition A subshift has T-structure if it is formed of this SFT with something on the grid only.

Let X, Y be **two subshifts with T-structure** with $F(X) = Y$, then

$$\alpha\text{-configuration} \xrightarrow{F} \alpha\text{-configuration}.$$

Shifted at most by the radius of F.

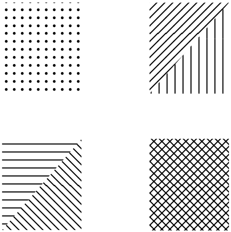
Why is such a construction interesting?

Definition A subshift has T-structure if it is formed of this SFT with something on the grid only.

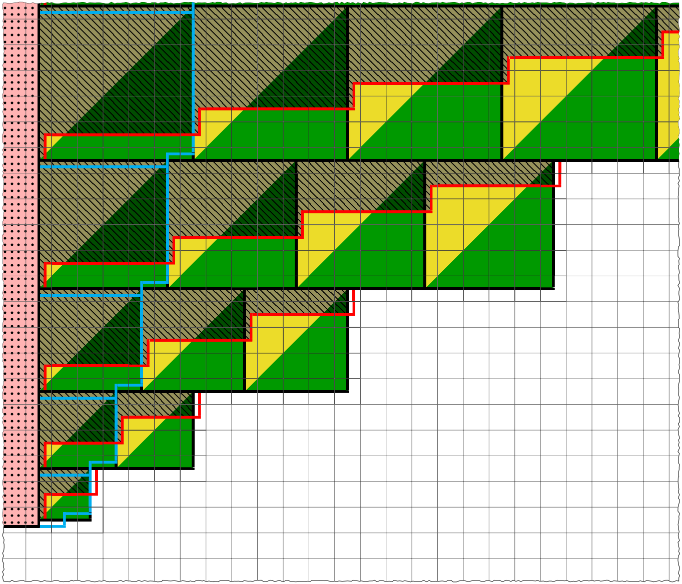
Let X, Y be **two subshifts with T-structure** with $F(X) = Y$, then

$$\alpha\text{-configuration} \xrightarrow{F} \alpha\text{-configuration}.$$

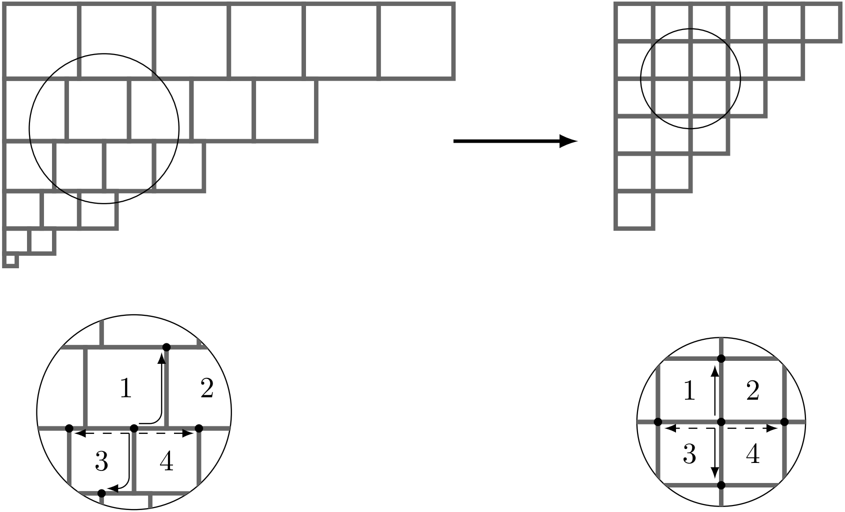
Shifted at most by the radius of F.



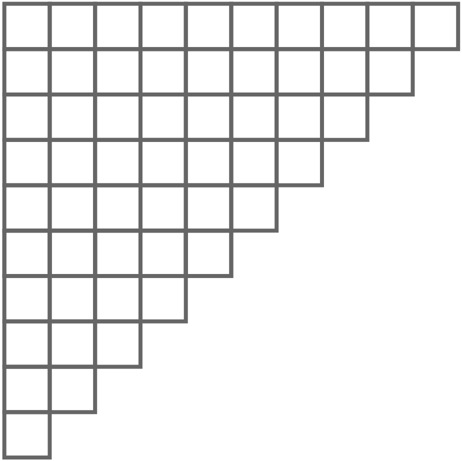
Lower Bound: the reduction



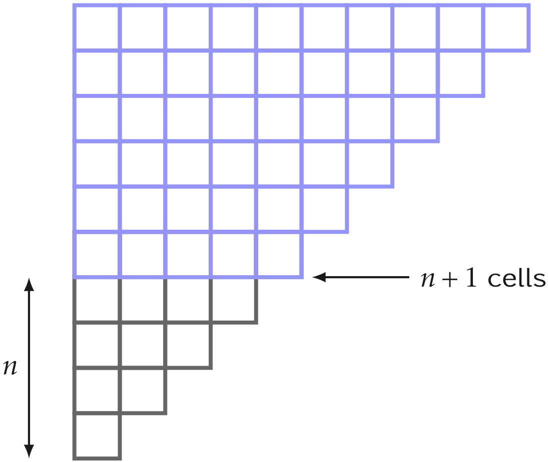
Lower Bound: the reduction



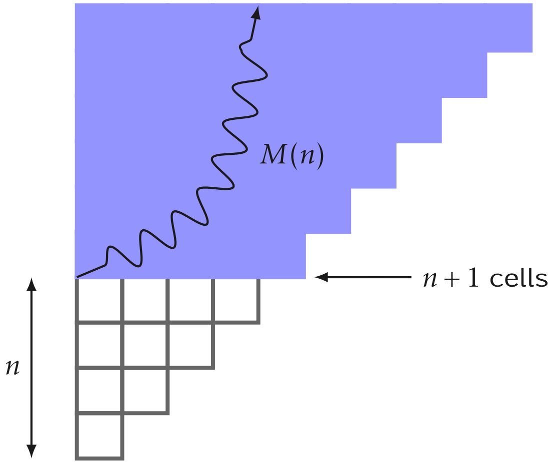
Lower Bound: the reduction



Lower Bound: the reduction



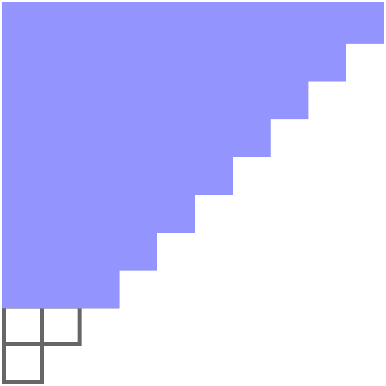
Lower Bound: the reduction



Lower Bound: the reduction

When $\{n \mid M(n)\uparrow\}$ **is infinite**, there are points with **computation** starting **arbitrarily far** from the start.

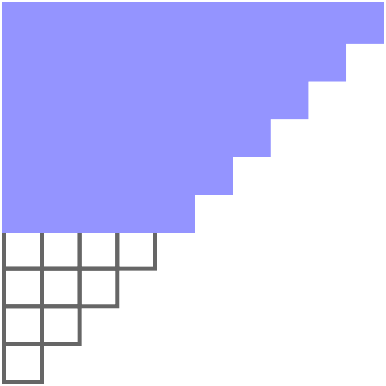
When $\{n \mid M(n)\uparrow\}$ **is finite**, there is an N such that **no computation starts after N**.



Lower Bound: the reduction

When $\{n \mid M(n)\uparrow\}$ **is infinite**, there are points with **computation** starting **arbitrarily far** from the start.

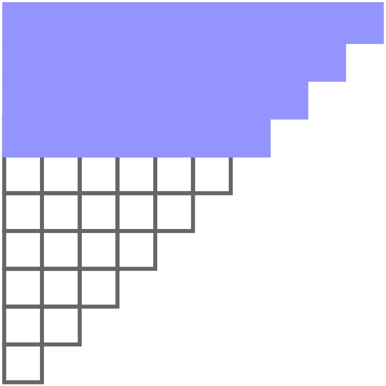
When $\{n \mid M(n)\uparrow\}$ **is finite**, there is an N such that **no computation starts after N**.



Lower Bound: the reduction

When $\{n \mid M(n)\uparrow\}$ **is infinite**, there are points with **computation** starting **arbitrarily far** from the start.

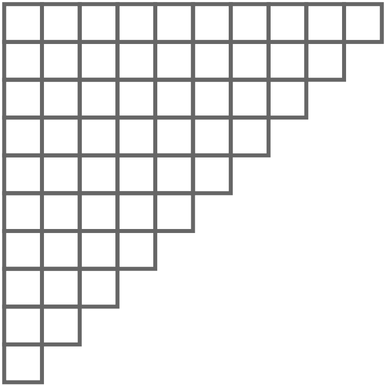
When $\{n \mid M(n)\uparrow\}$ **is finite**, there is an N such that **no computation starts after** N.



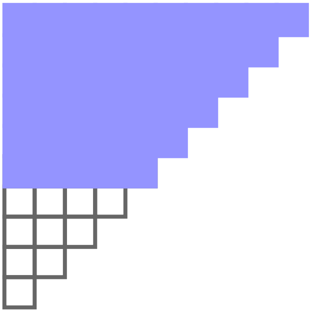
Lower Bound: the reduction

When $\{n \mid M(n)\uparrow\}$ **is infinite**, there are points with **computation** starting **arbitrarily far** from the start.

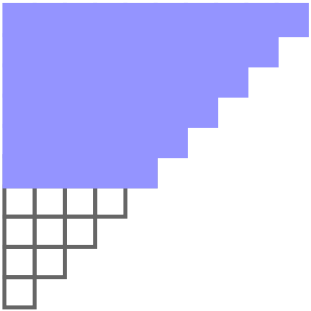
When $\{n \mid M(n)\uparrow\}$ **is finite**, there is an N such that **no computation starts after N**.



Lower Bound: the reduction

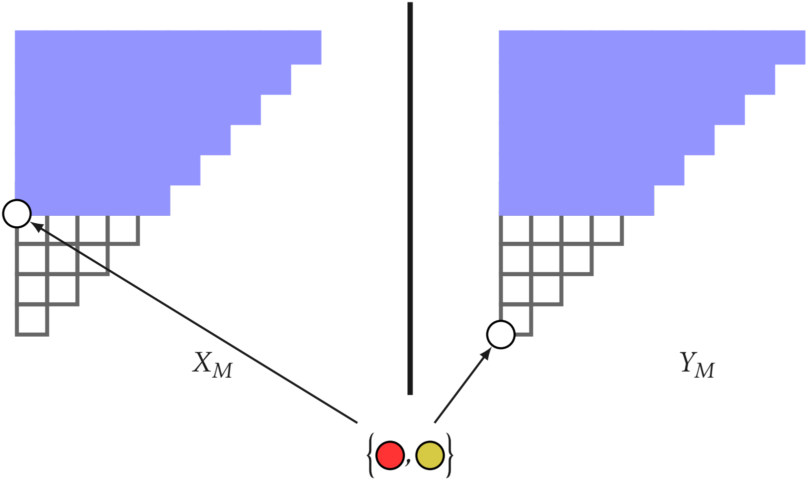


X_M

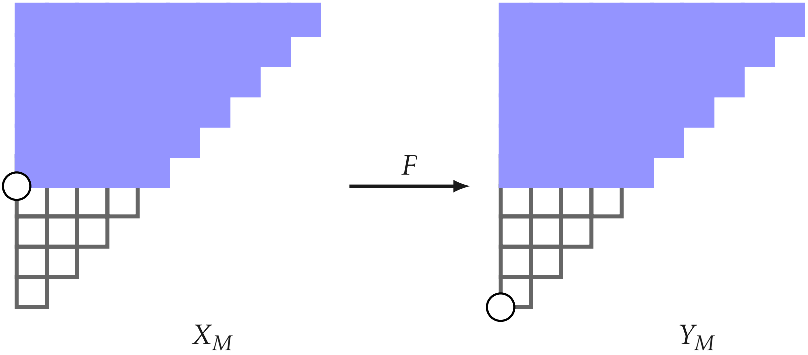


Y_M

Lower Bound: the reduction

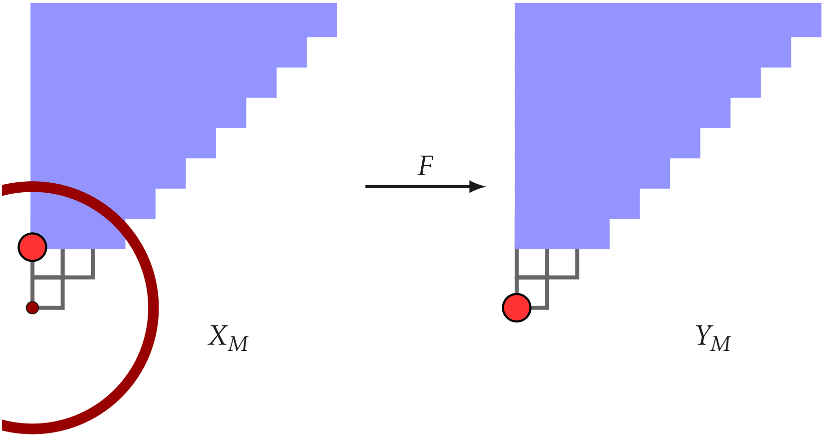


Lower Bound: the reduction

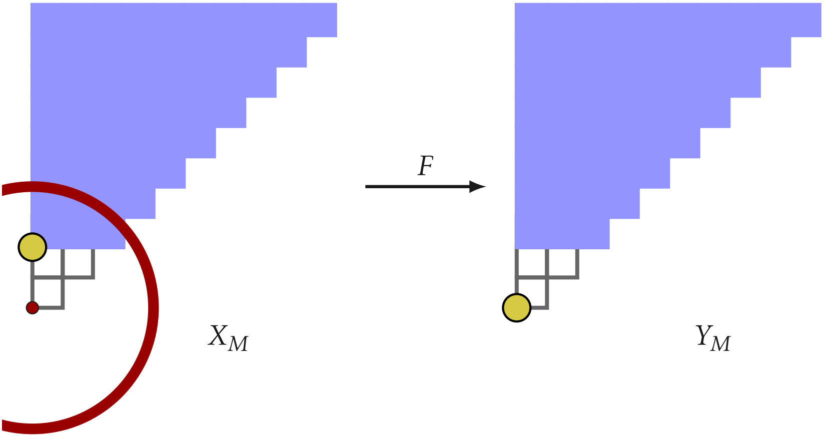


Suppose $\{n \mid M(n)\uparrow\}$ is **infinite** and that there exists a **factor map** $F : X \to Y$ of **radius** r.

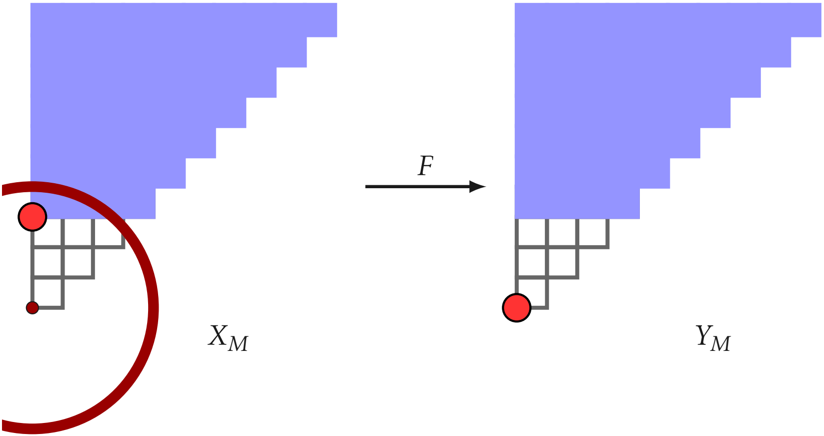
Lower Bound: the reduction



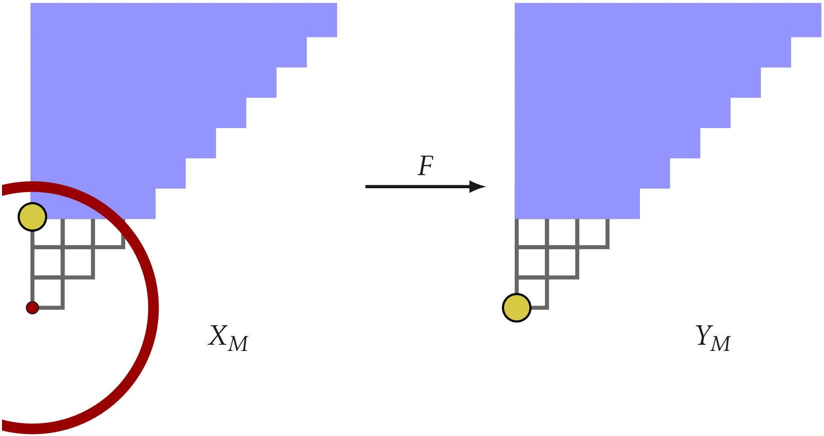
Lower Bound: the reduction



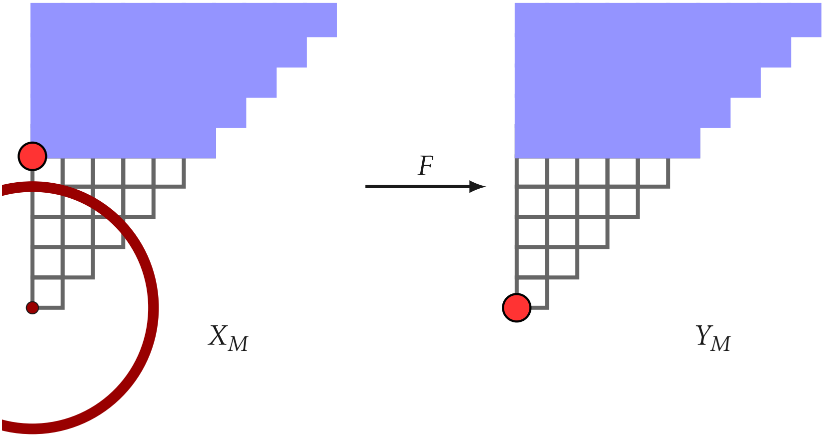
Lower Bound: the reduction



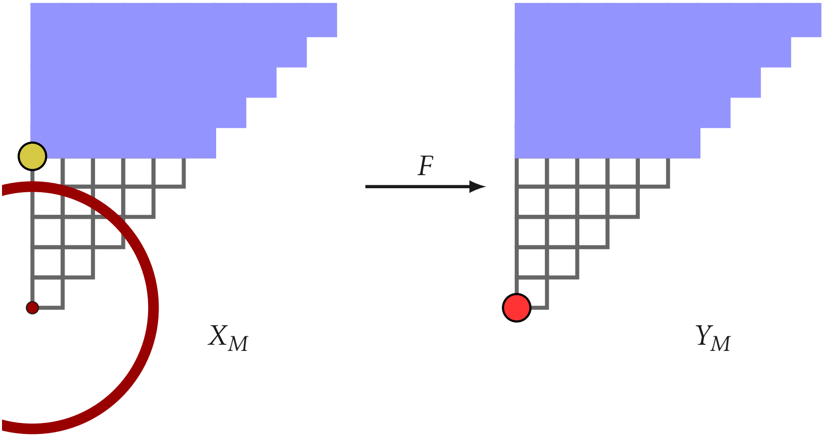
Lower Bound: the reduction



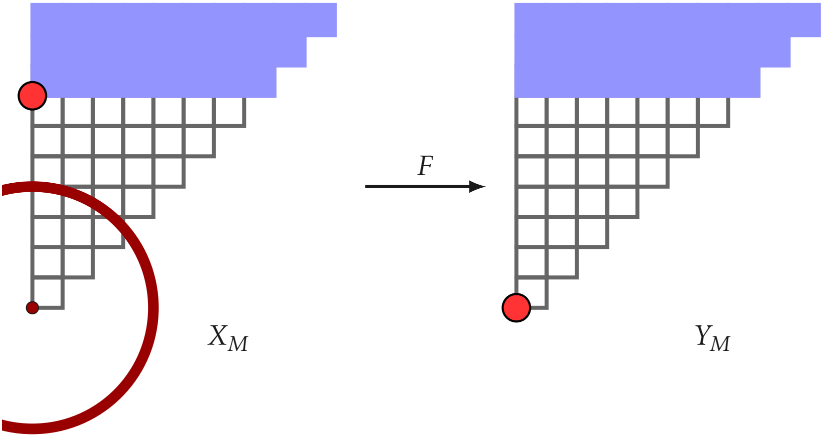
Lower Bound: the reduction



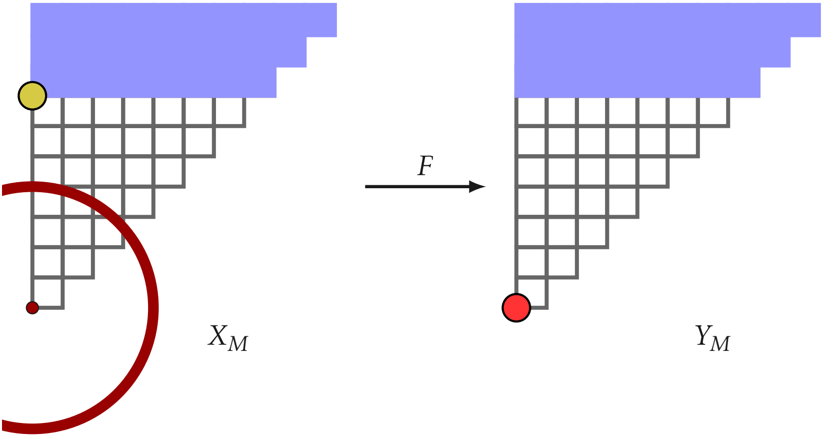
Lower Bound: the reduction



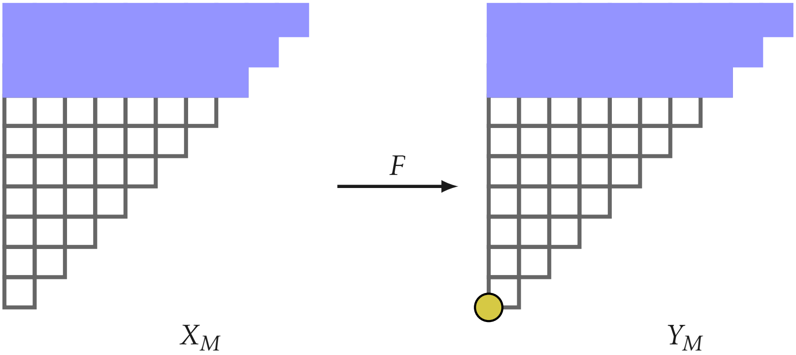
Lower Bound: the reduction



Lower Bound: the reduction

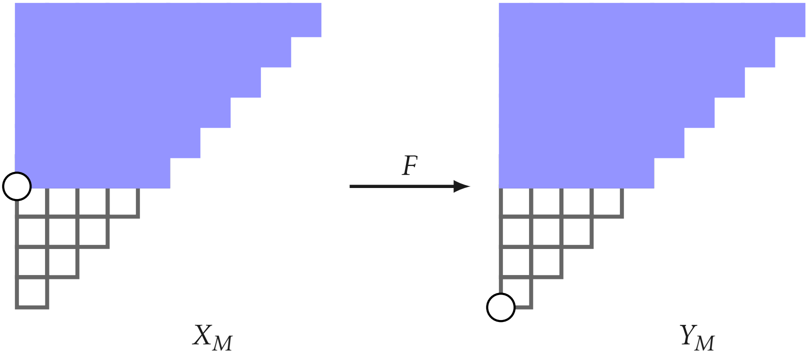


Lower Bound: the reduction



An infinity of ◯ in Y_M don't have a preimage.

Lower Bound: the reduction



When it is **finite**, just take the radius so that it covers the **largest element**.

1. Conjugacy
2. Factorization
3. Embedding

Embedding

Definition A subshift X **embeds** into a subshift Y if there exists an **injective block code** $F : X \to Y$.

Theorem Given two SFTs X, Y, deciding whether X embeds into Y is Σ_1^0.

Again, the proof is a mix of compactness and formula manipulations.

Embedding

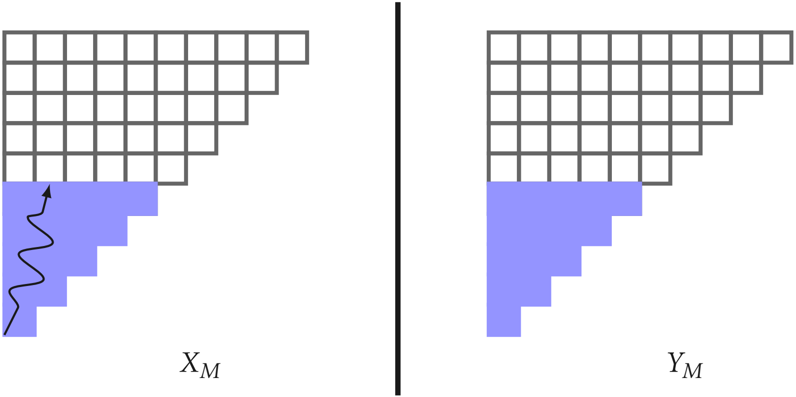
Theorem Given two SFTs X, Y, deciding whether X embeds into Y is Σ_1^0-hard.

Let X, Y be **two subshifts with T-structure** with $X \rightsquigarrow Y$, then

$$\alpha\text{-}\mathbf{configuration} \xrightarrow{F} \alpha\text{-}\mathbf{configuration}.$$

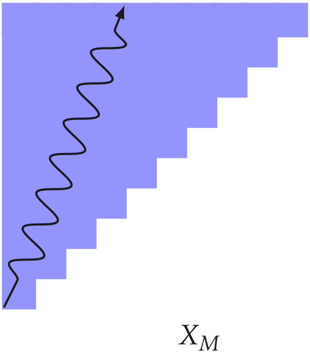
Shifted at most by the radius of F.

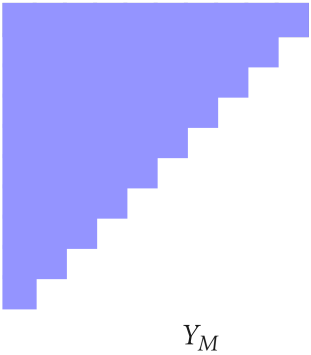
Embedding: the reduction



M **halts** on no input.

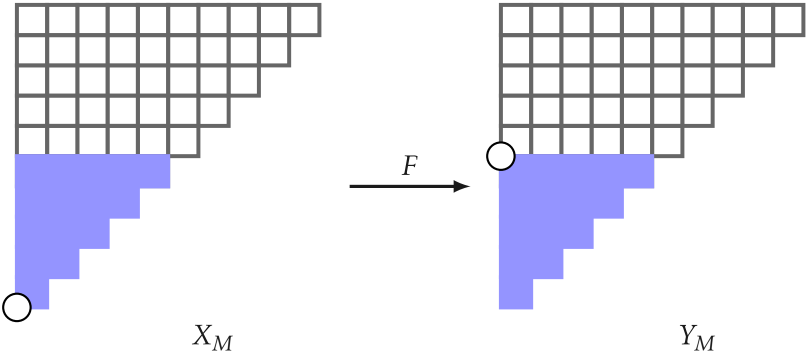
Embedding: the reduction



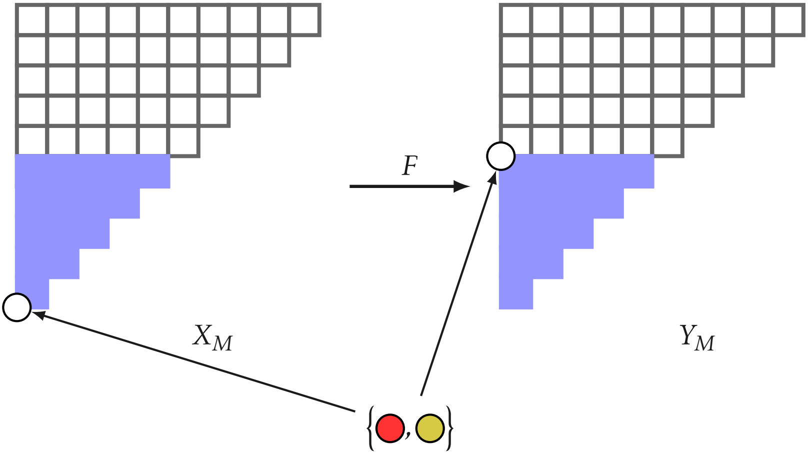


M **does not halt** with no input.

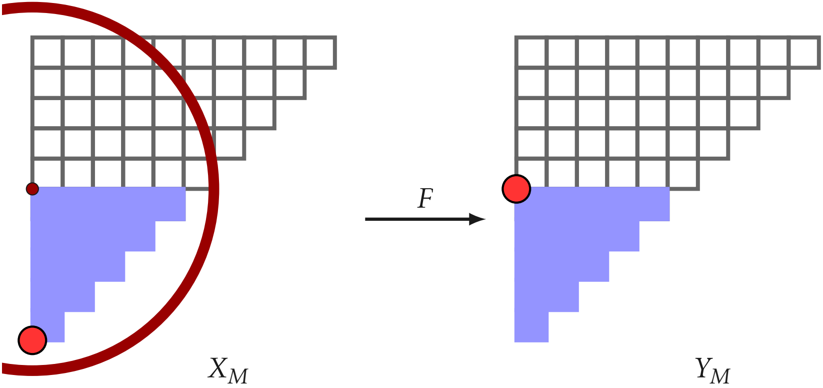
Embedding: the reduction



Embedding: the reduction

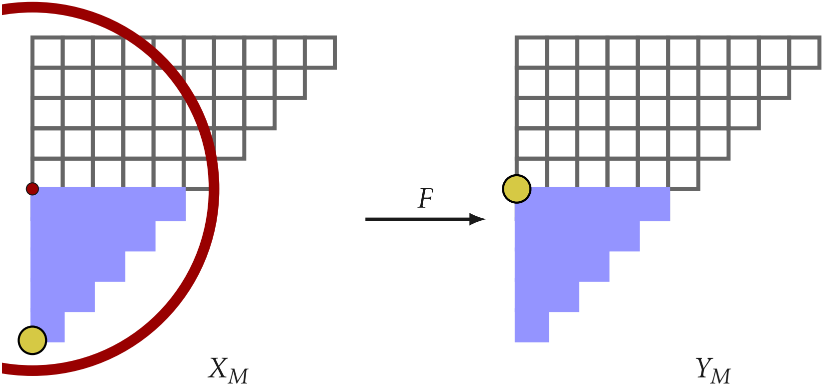


Embedding: the reduction



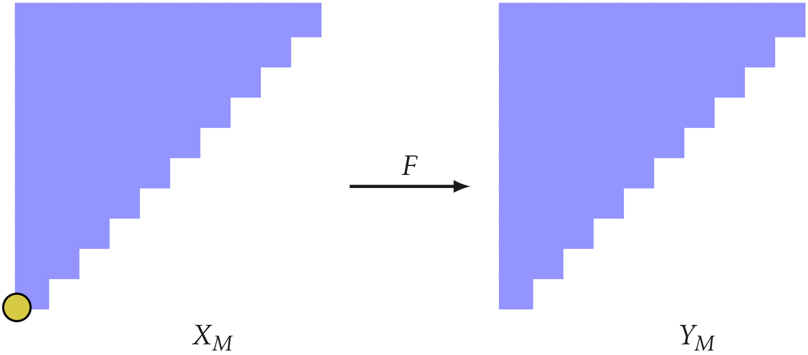
When the **machine halts** take F to have as **radius** the **time** that the machine takes **to halt**.

Embedding: the reduction



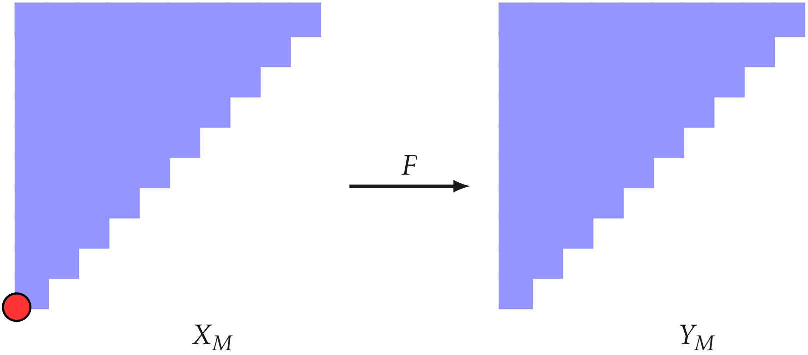
When the **machine halts** take F to have as **radius** the **time** that the machine takes **to halt**.

Embedding: the reduction



When the **machine does not halt**, the two grids have the same image.

Embedding: the reduction



When the **machine does not halt**, the two grids have the same image.

Conclusion

	Conjugacy	Factorization	Embedding
SFTs	Σ_1^0-complete	Σ_3^0-complete	Σ_1^0-complete
Sofic	Σ_3^0-complete	Σ_3^0-complete	?
Effective	Σ_3^0-complete	Σ_3^0-complete	?