

The one-marked-component shift

Julien Cassaigne and Pierre Guillon

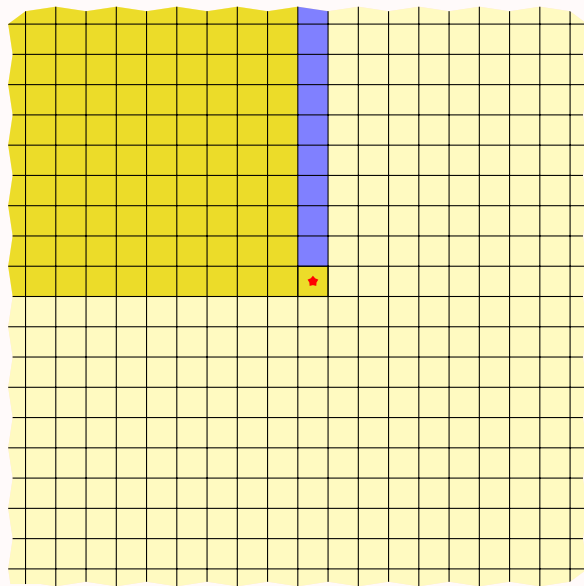
I2Marseille, CNRS & AMU

january 2021

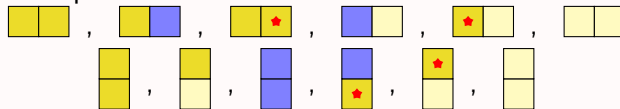
2D shifts

- **alphabet:** finite set $\mathcal{A} (= \{\square, \blacksquare, \blacksquare, \blacksquare\})$
- **configuration:** $x \in \mathcal{A}^{\mathbb{Z}^2}$
- **pattern:** $u \in \mathcal{A}^F, F \subseteq \mathbb{Z}^2$
- **u appears** in x ; $u \sqsubset x$: $u = x_{\mathbf{i}+F}$ for some $\mathbf{i} \in \mathbb{Z}^2$
- **shift:**

$$\mathcal{X}_{\mathcal{F}} := \left\{ x \in \mathcal{A}^{\mathbb{Z}^2} \mid \forall u \in \mathcal{F}, u \not\sqsubset x \right\}$$
 for some $\mathcal{F} \subset \bigcup_{F \subseteq \mathbb{Z}^2} \mathcal{A}^F$
- **of finite type:** $\mathcal{X}_{\mathcal{F}}$, for $|\mathcal{F}| < \infty$



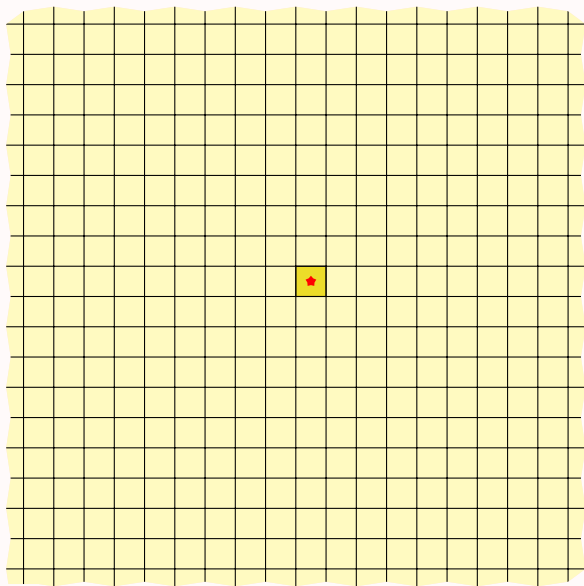
allowed patterns:



2D shifts

- **alphabet:** finite set $\mathcal{A} (= \{\square, \blacksquare, \blacksquare, \blacksquare\})$
- **configuration:** $x \in \mathcal{A}^{\mathbb{Z}^2}$
- **pattern:** $u \in \mathcal{A}^F, F \subseteq \mathbb{Z}^2$
- **u appears** in x ; $u \sqsubset x$: $u = x_{\mathbf{i}+F}$ for some $\mathbf{i} \in \mathbb{Z}^2$
- **shift:**
$$\mathcal{X}_{\mathcal{F}} := \left\{ x \in \mathcal{A}^{\mathbb{Z}^2} \mid \forall u \in \mathcal{F}, u \not\sqsubset x \right\}$$

for some $\mathcal{F} \subset \bigcup_{F \subseteq \mathbb{Z}^2} \mathcal{A}^F$
- **of finite type:** $\mathcal{X}_{\mathcal{F}}$, for $|\mathcal{F}| < \infty$
- **sofic:** image of SFT by alphabet projection
 $\pi : \mathcal{A} \rightarrow \mathcal{B}$



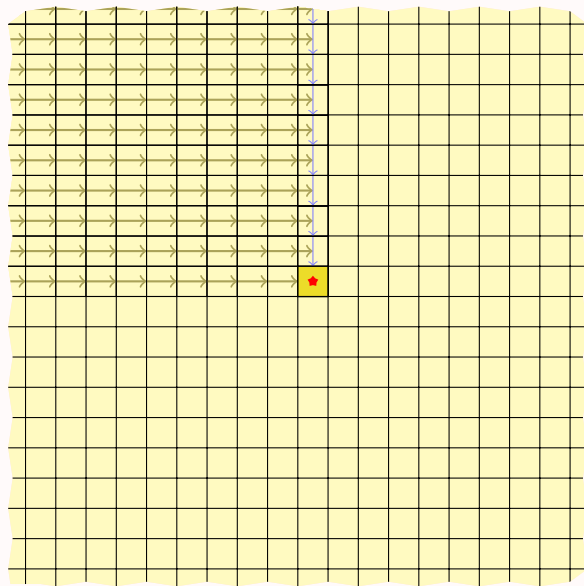
e.g. shift of configurations which have ≤ 1 .

2D shifts

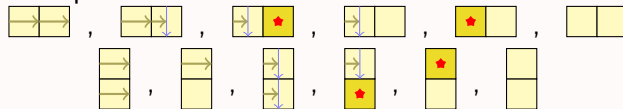
- **alphabet**: finite set $\mathcal{A} (= \{\square, \blacksquare, \blacksquare, \blacksquare\})$
- **configuration**: $x \in \mathcal{A}^{\mathbb{Z}^2}$
- **pattern**: $u \in \mathcal{A}^F, F \subseteq \mathbb{Z}^2$
- u **appears** in x ; $u \sqsubset x$: $u = x_{\mathbf{i}+F}$ for some $\mathbf{i} \in \mathbb{Z}^2$
- **shift**:

$$\mathcal{X}_{\mathcal{F}} := \left\{ x \in \mathcal{A}^{\mathbb{Z}^2} \mid \forall u \in \mathcal{F}, u \not\sqsubset x \right\}$$
 for some $\mathcal{F} \subset \bigcup_{F \subseteq \mathbb{Z}^2} \mathcal{A}^F$
- **of finite type**: $\mathcal{X}_{\mathcal{F}}$, for $|\mathcal{F}| < \infty$
- **sofic**: image of SFT by alphabet projection
 $\pi : \mathcal{A} \rightarrow \mathcal{B}$

e.g. shift of configurations which have ≤ 1 .



allowed patterns:

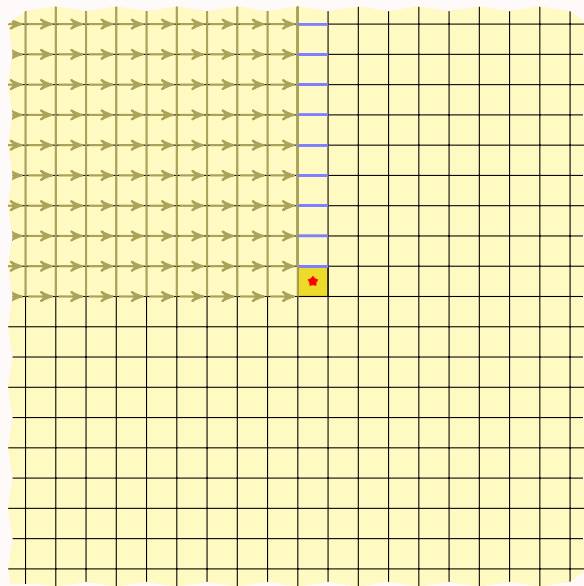


2D shifts

- **alphabet:** finite set $\mathcal{A} (= \{\square, \blacksquare, \square, \star\})$
- **configuration:** $x \in \mathcal{A}^{\mathbb{Z}^2}$
- **pattern:** $u \in \mathcal{A}^F, F \subseteq \mathbb{Z}^2$
- **u appears** in x ; $u \sqsubset x$: $u = x_{\mathbf{i}+F}$ for some $\mathbf{i} \in \mathbb{Z}^2$
- **shift:**

$$\mathcal{X}_{\mathcal{F}} := \left\{ x \in \mathcal{A}^{\mathbb{Z}^2} \mid \forall u \in \mathcal{F}, u \not\sqsubset x \right\}$$
 for some $\mathcal{F} \subset \bigcup_{F \subseteq \mathbb{Z}^2} \mathcal{A}^F$
- **of finite type:** $\mathcal{X}_{\mathcal{F}}$, for $|\mathcal{F}| < \infty$
- **sofic:** image of SFT by alphabet projection
 $\pi : \mathcal{A} \rightarrow \mathcal{B}$

e.g. shift of configurations which have ≤ 1 $\star$.

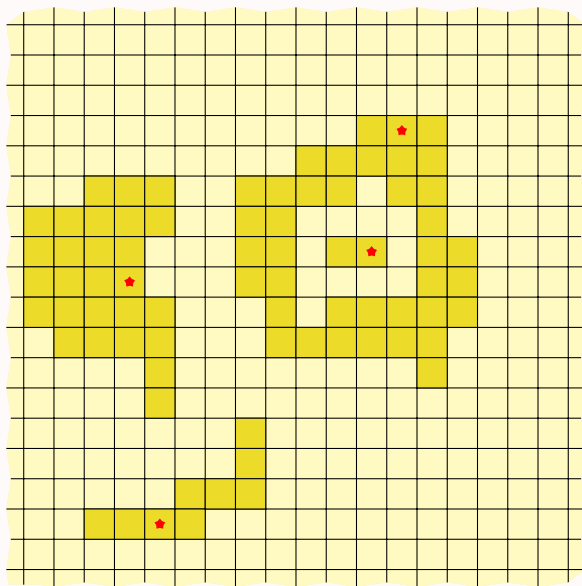


tiles:     

The one-marked-component shift

- alphabet: $\mathcal{A} = \{\text{white square}, \text{yellow square}, \text{yellow square with red star}\}$
- **component**: maximal connected subgraph of the grid colored in $\{\text{yellow square}, \text{yellow square with red star}\}$.
- $\mathcal{Y}$: configurations whose finite components have **exactly 1** yellow square with red star.

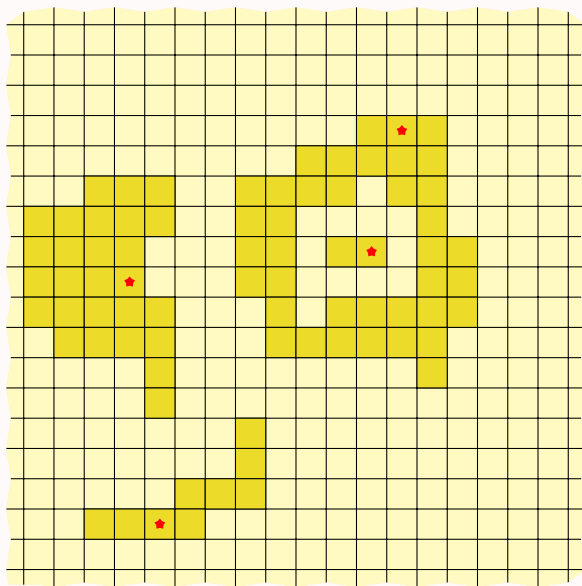
Thm: $\mathcal{Y}$ is sofic.



The one-marked-component shift

- alphabet: $\mathcal{A} = \{\square, \blacksquare, \star\}$
- **component**: maximal connected subgraph of the grid colored in $\{\blacksquare, \star\}$.
- $\mathcal{Y}$: configurations whose finite components have **exactly 1** $\star$.
- $\mathcal{Y}'$: configurations from $\mathcal{Y}$ avoiding .

Thm: $\mathcal{Y}'$ is **sofic**.



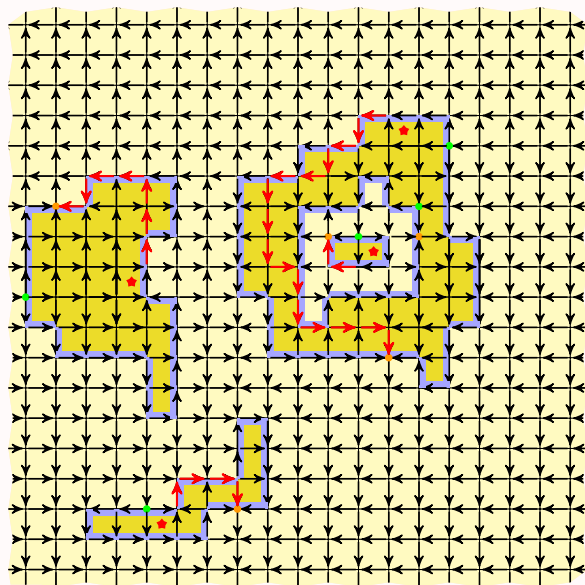
The one-marked-component shift

- alphabet: $\mathcal{A} = \{\square, \blacksquare, \star\}$
- **component**: maximal connected subgraph of the grid colored in $\{\blacksquare, \star\}$.
- $\mathcal{Y}$: configurations whose finite components have **exactly 1** $\star$.
- $\mathcal{Y}'$: configurations from $\mathcal{Y}$ avoiding .

Thm: $\mathcal{Y}'$ is **sofic**.

We build:

- a SFT $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \star\}^{\mathbb{Z}^2} \times \mathcal{T} \xrightarrow{\pi_2} \mathcal{Y}'$



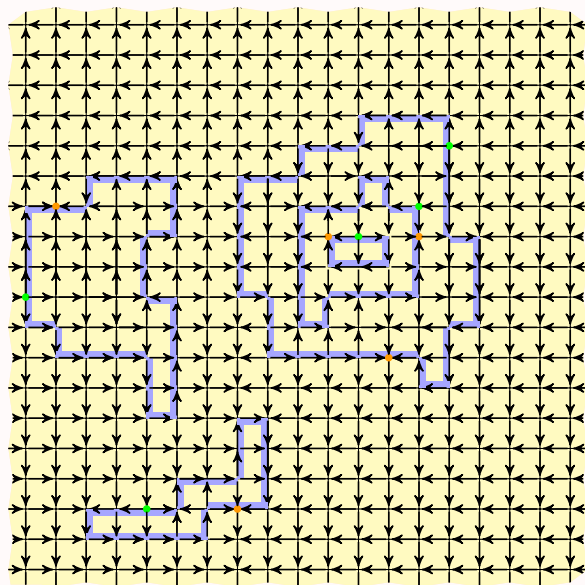
The one-marked-component shift

- alphabet: $\mathcal{A} = \{\square, \blacksquare, \star\}$
- **component**: maximal connected subgraph of the grid colored in $\{\blacksquare, \star\}$.
- $\mathcal{Y}$: configurations whose finite components have **exactly 1** $\star$.
- $\mathcal{Y}'$: configurations from $\mathcal{Y}$ avoiding .

Thm: $\mathcal{Y}'$ is **sofic**.

We build:

- a SFT $\mathcal{W} \subset \mathcal{S} \times \mathcal{W}'$
- a SFT $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \star\}^{\mathbb{Z}^2} \times \mathcal{T} \xrightarrow{\pi_2} \mathcal{Y}'$



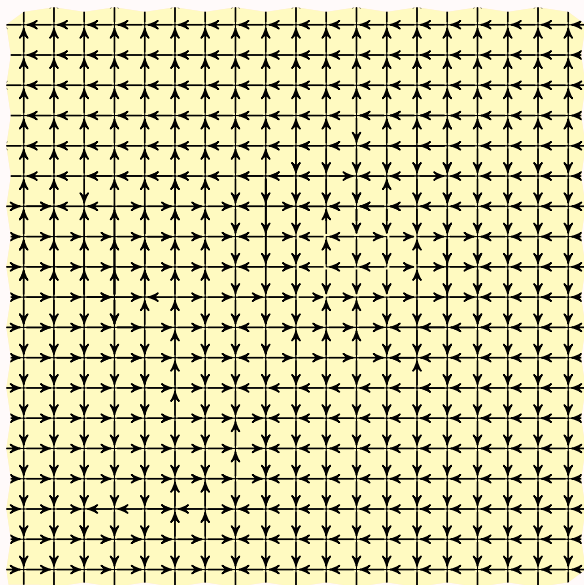
The one-marked-component shift

- alphabet: $\mathcal{A} = \{\square, \blacksquare, \star\}$
- **component**: maximal connected subgraph of the grid colored in $\{\blacksquare, \star\}$.
- $\mathcal{Y}$: configurations whose finite components have **exactly 1** $\star$.
- $\mathcal{Y}'$: configurations from $\mathcal{Y}$ avoiding .

Thm: $\mathcal{Y}'$ is **sofic**.

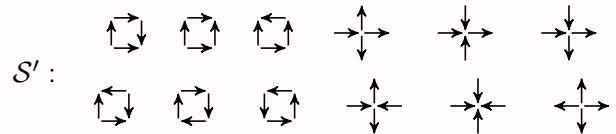
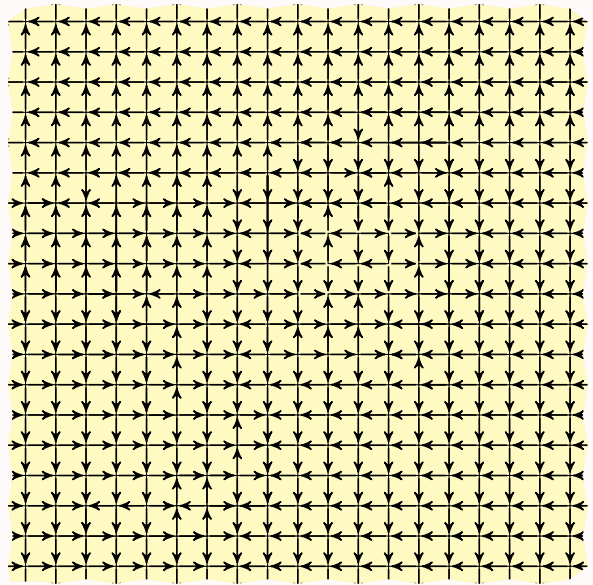
We build:

- a SFT $\mathcal{S}'$: **streams**
- a SFT $\mathcal{S} \subset \mathcal{S}'$: **chorded streams**
- a SFT $\mathcal{W} \subset \mathcal{S} \times \mathcal{W}'$: **surrounding streams**
- a SFT $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \star\}^{\mathbb{Z}^2} \times \mathcal{T} \xrightarrow{\pi_2} \mathcal{Y}'$



SFT of streams

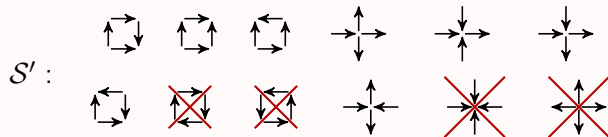
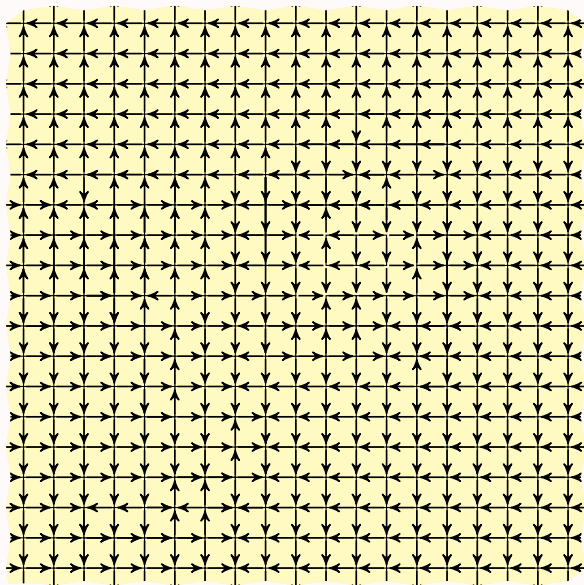
- orientation of the grid



SFT of streams

- orientation of the grid
- **stream**: acyclic orientation of the grid with no source and no sink.

Lem: $\mathcal{S}'$ is the set of streams.



SFT of streams

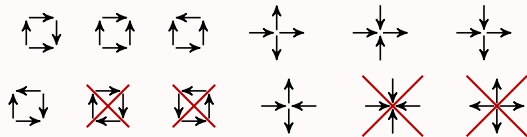
- orientation of the grid
- **stream**: acyclic orientation of the grid with no source and no sink.

Lem: $\mathcal{S}'$ is the set of streams.

Proof:

-  is forbidden.

$\mathcal{S}'$:



SFT of streams

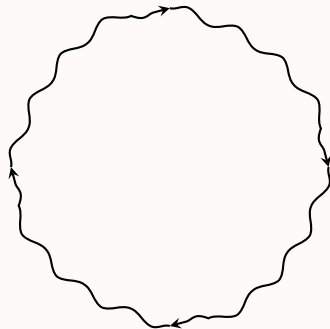
- orientation of the grid
- **stream**: acyclic orientation of the grid with no source and no sink.

Lem: $\mathcal{S}'$ is the set of streams.

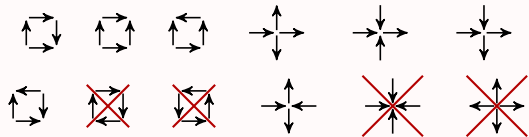
Proof:

-  is forbidden.
- Larger cycles

□



$\mathcal{S}'$:



SFT of streams

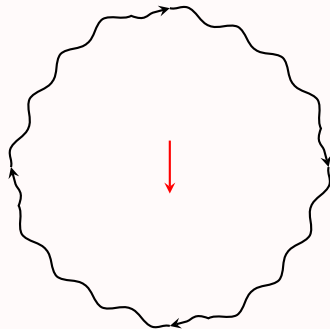
- orientation of the grid
- **stream**: acyclic orientation of the grid with no source and no sink.

Lem: $\mathcal{S}'$ is the set of streams.

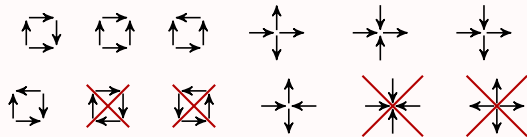
Proof:

-  is forbidden.
- Larger cycles

□



$\mathcal{S}'$:



SFT of streams

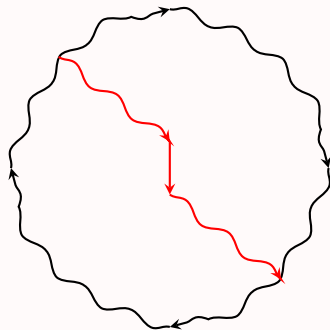
- orientation of the grid
- **stream**: acyclic orientation of the grid with no source and no sink.

Lem: $\mathcal{S}'$ is the set of streams.

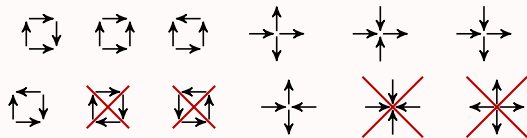
Proof:

-  is forbidden.
- Larger cycles surround a dipath
→ smaller cycle.

□



$\mathcal{S}'$:

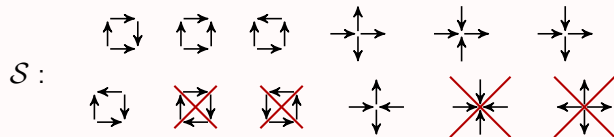
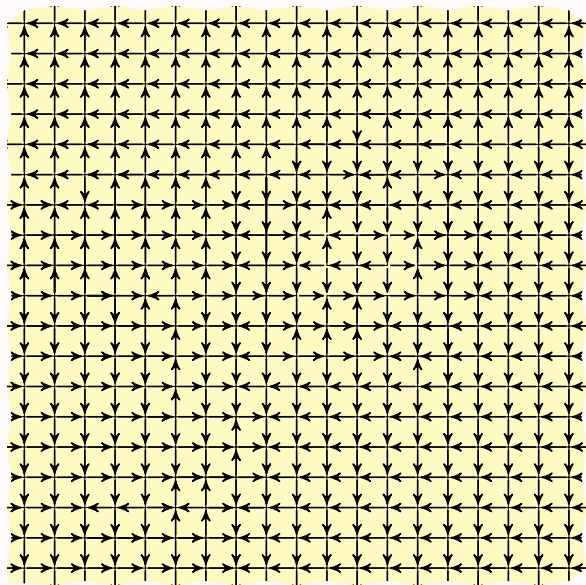


SFT of chorded streams

in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

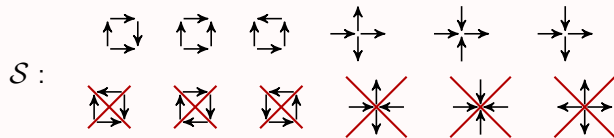
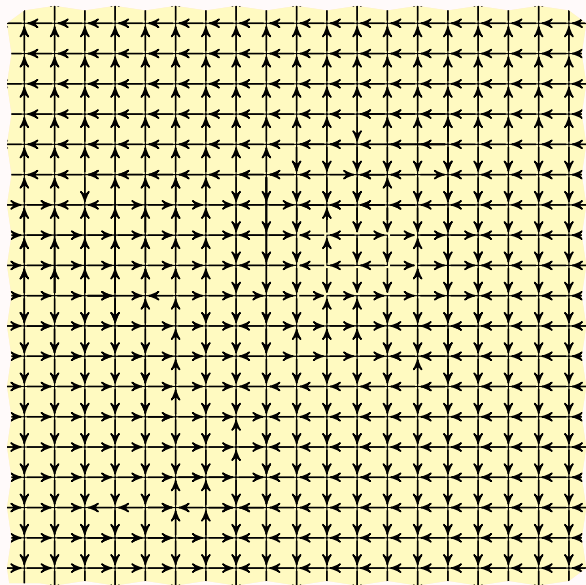


SFT of chorded streams

in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.



SFT of chorded streams

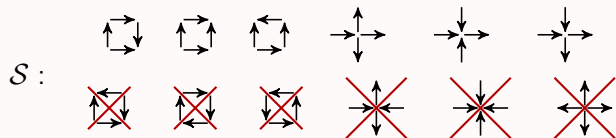
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.



SFT of chorded streams

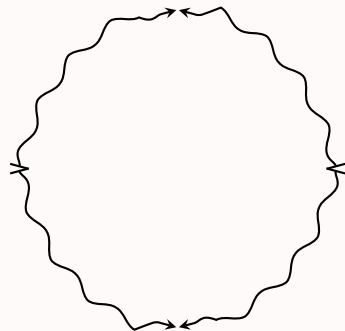
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

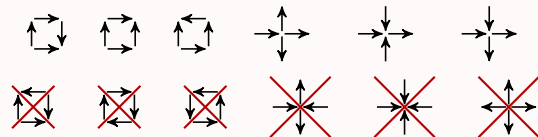
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c



$\mathcal{S}$:



SFT of chorded streams

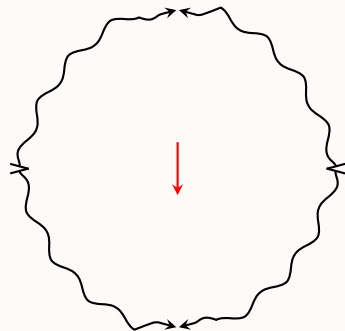
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

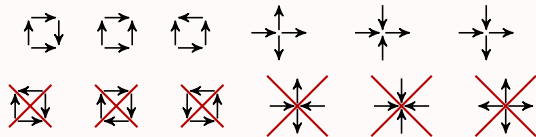
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

- is forbidden.
- Larger cycle c



$\mathcal{S}$:



SFT of chorded streams

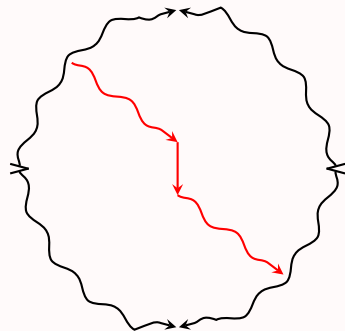
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath joining 2 changes of direction of same type, staying inside the surrounded region

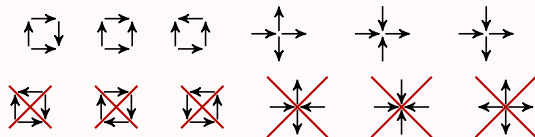
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.



$\mathcal{S}$:



SFT of chorded streams

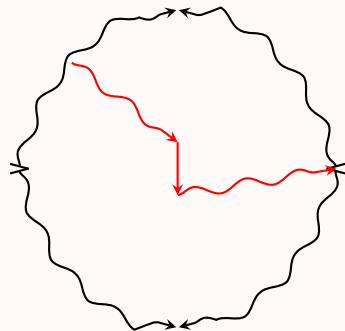
in a **nondirected** cycle:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath joining 2 changes of direction of same type, staying inside the surrounded region

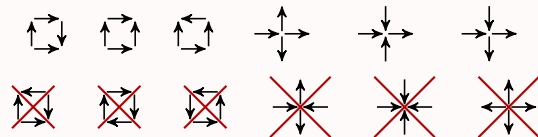
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .



$\mathcal{S}$:



SFT of chorded streams

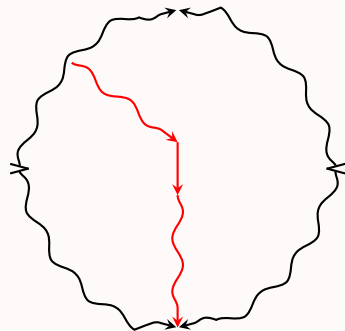
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath joining 2 changes of direction of same type, staying inside the surrounded region

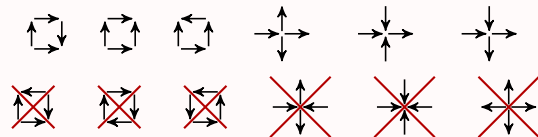
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.



$\mathcal{S}$:



SFT of chorded streams

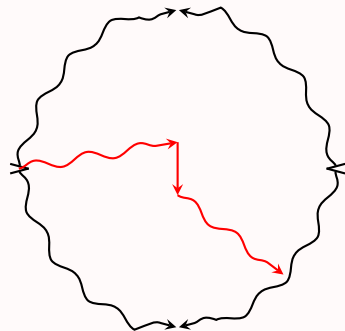
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath joining 2 changes of direction of same type, staying inside the surrounded region

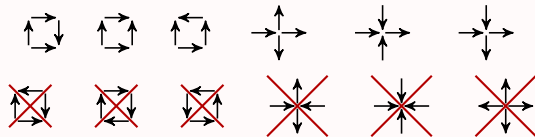
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.



$\mathcal{S}$:



SFT of chorded streams

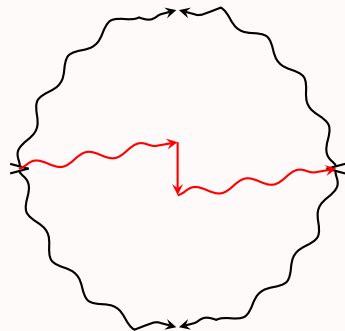
in a **nondirected** cycle:

- **change of direction:** $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord:** dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

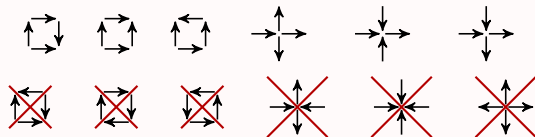
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.



$\mathcal{S}$:



SFT of chorded streams

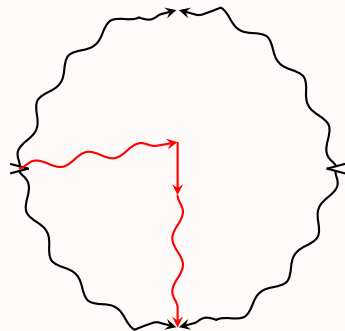
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

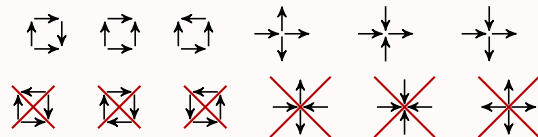
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

- is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.



$\mathcal{S}$:



SFT of chorded streams

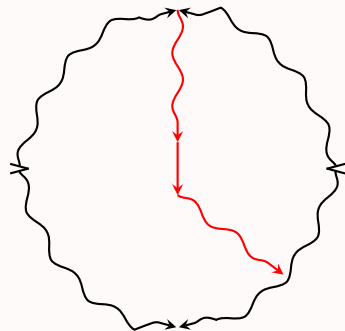
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath joining 2 changes of direction of same type, staying inside the surrounded region

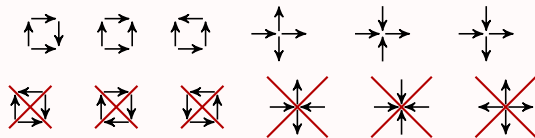
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.



$\mathcal{S}$:



SFT of chorded streams

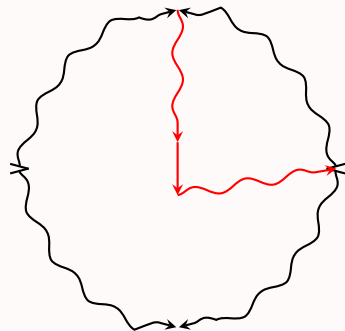
in a **nondirected** cycle:

- **change of direction:** $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord:** dipath joining 2 changes of direction of same type, staying inside the surrounded region

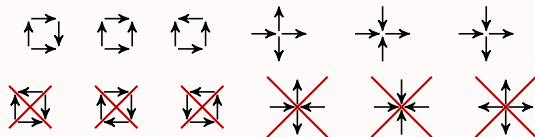
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.



$\mathcal{S}$:



SFT of chorded streams

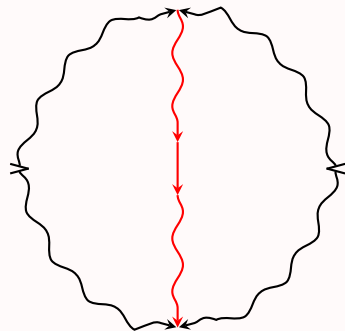
in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath joining 2 changes of direction of same type, staying inside the surrounded region

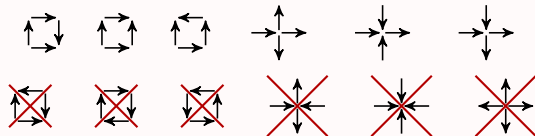
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.



$\mathcal{S}$:



SFT of chorded streams

in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

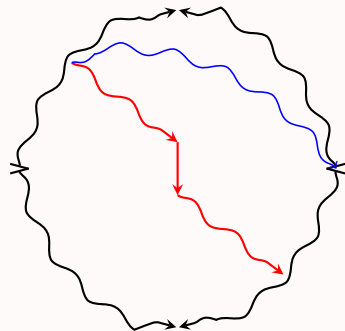
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

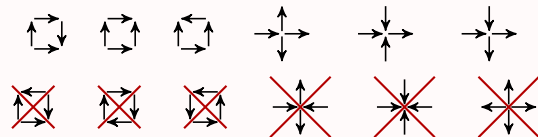
-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.

A chord of c' , not forbidden by ,

extends to a chord of c . □



$\mathcal{S}$:



SFT of chorded streams

in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

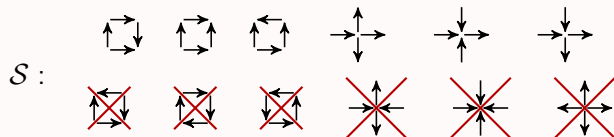
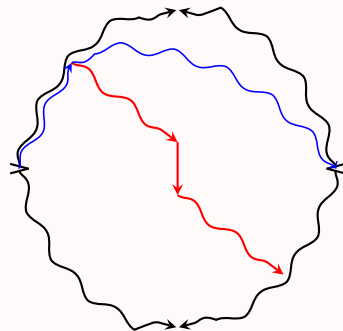
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.

A chord of c' , not forbidden by ,

extends to a chord of c . □



SFT of chorded streams

in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath **joining 2 changes** of direction of same type, staying inside the surrounded region

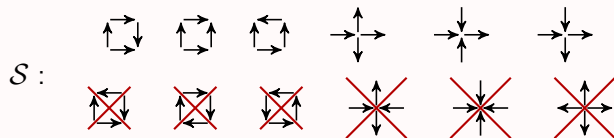
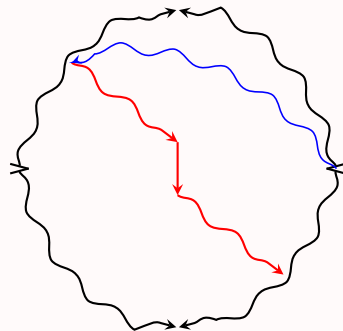
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.

A chord of c' , not forbidden by ,

extends to a chord of c . □



SFT of chorded streams

in a **nondirected cycle**:

- **change of direction**: $\leftarrow \rightarrow$ or $\rightarrow \leftarrow$
- **chord**: dipath joining 2 changes of direction of same type, staying inside the surrounded region

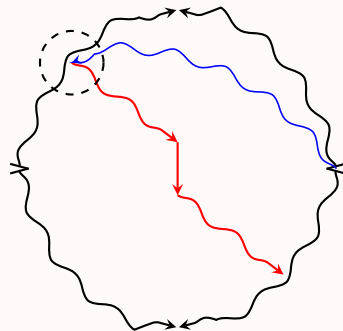
Lem: $\mathcal{S}$ is the set of **streams** whose (nondirected) cycles with ≥ 4 changes of direction admit **chords**.

Proof:

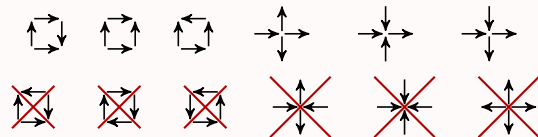
-  is forbidden.
- Larger cycle c surrounds a dipath.
 - ▶ $\rightarrow$ either extends to a chord of c .
 - ▶ $\rightarrow$ or cuts smaller cycle c' with ≥ 4 changes.

A chord of c' , not forbidden by ,

extends to a chord of c . □

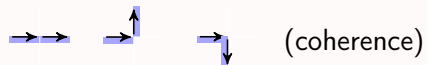
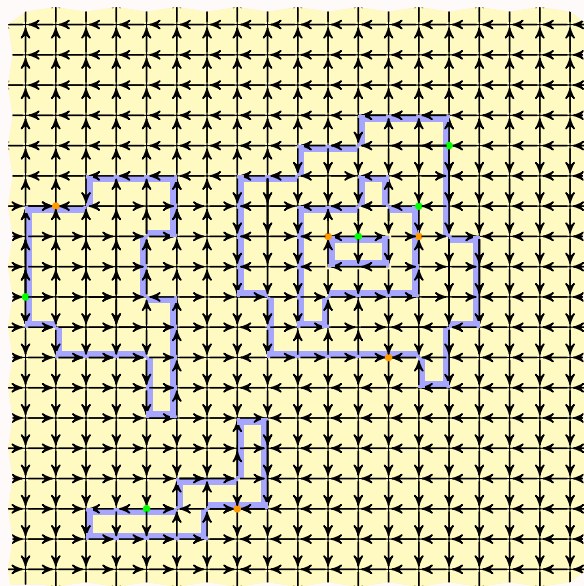


$\mathcal{S}$:



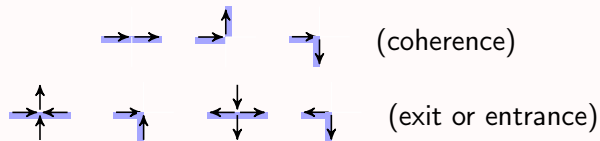
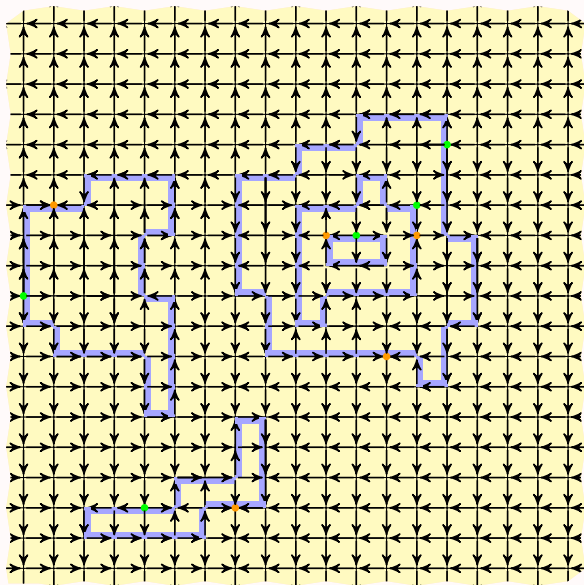
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.

- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.



Surrounding streams

- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**



(coherence)

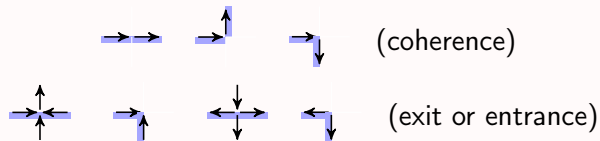
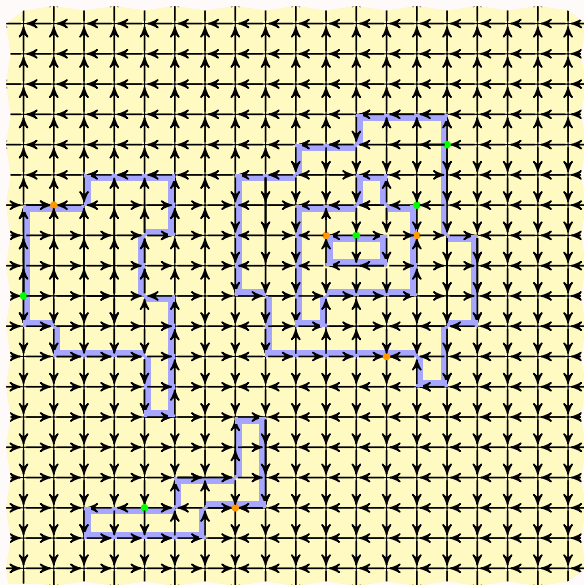
(exit or entrance)

Surrounding streams

- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.



Surrounding streams

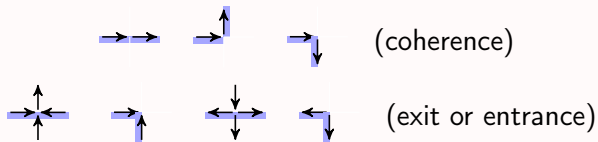
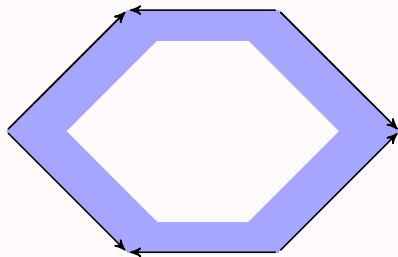
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction,



Surrounding streams

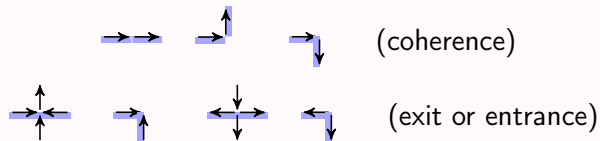
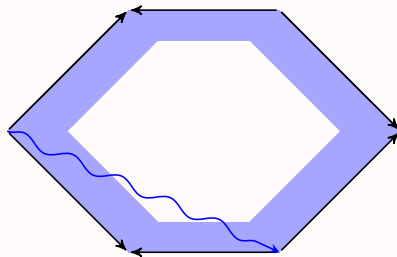
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord



Surrounding streams

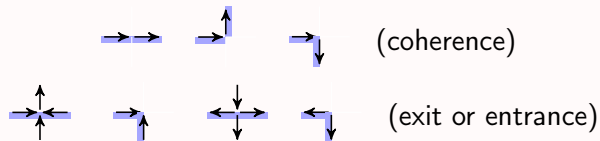
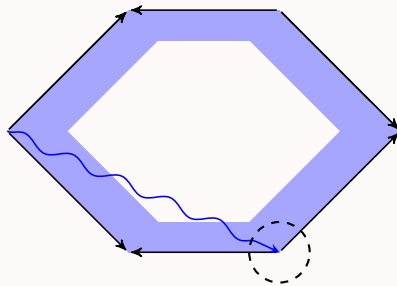
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord
→ violates entrance or exit constraint.



Surrounding streams

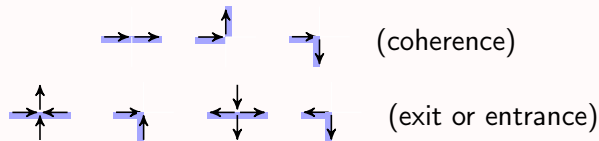
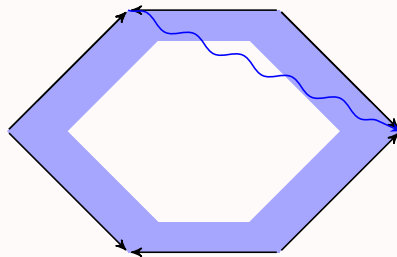
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord
→ violates entrance or exit constraint.



Surrounding streams

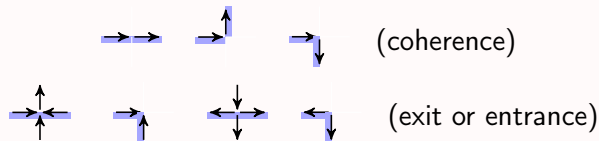
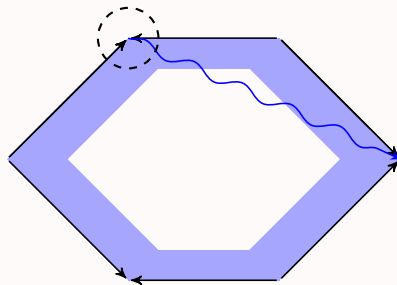
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord
→ violates entrance or exit constraint.



Surrounding streams

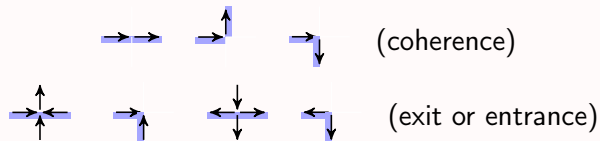
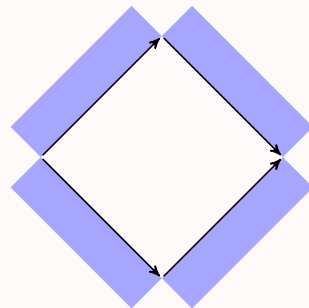
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord
→ violates entrance or exit constraint.
- If the finite region is not shaded,



Surrounding streams

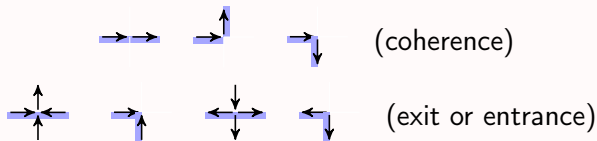
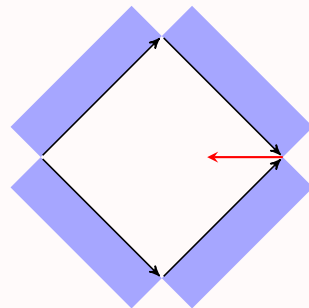
- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord
→ violates entrance or exit constraint.
- If the finite region is not shaded, the outgoing
arrow from the exit



Surrounding streams

- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

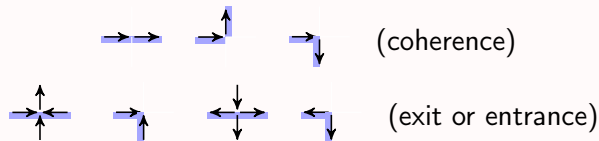
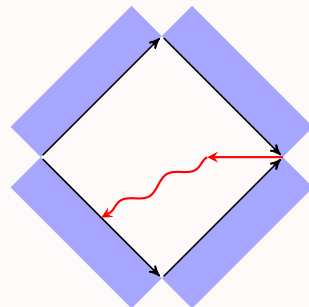
Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord
→ violates entrance or exit constraint.
- If the finite region is not shaded, the outgoing
arrow from the exit can be extended into a
cycle.

□



Surrounding streams

- Arrows can **shade** one side.
- Shading arrows form **coherent** paths.
- **entrance**:  forces arrow **from non-shaded**
- **exit**:  forces arrow **to non-shaded**

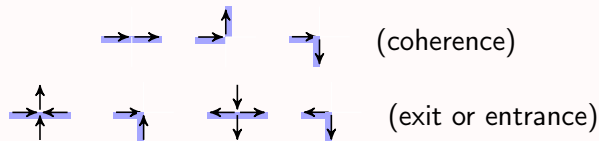
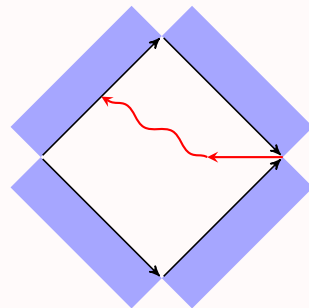
Lem: Any shading cycle:

- shades towards the **surrounded** region.
- admits exactly one **entrance** and one **exit**.

Proof:

- If > 2 changes of direction, it admits a chord
→ violates entrance or exit constraint.
- If the finite region is not shaded, the outgoing
arrow from the exit can be extended into a
cycle.

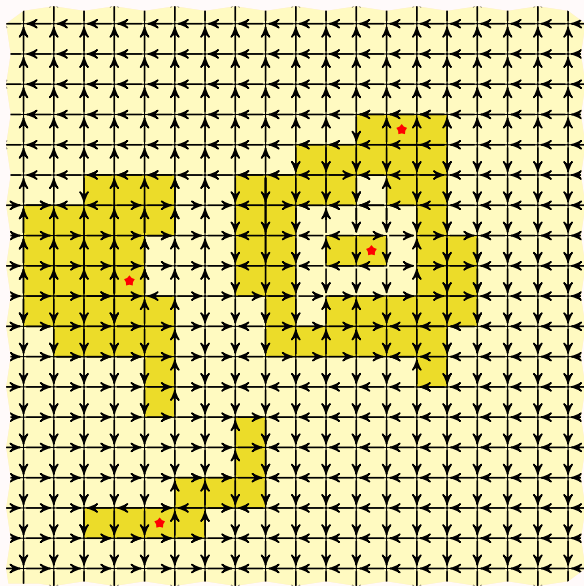
□



Mark-checking trees

Define $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^2} \times \mathcal{T}$:

- Superimpose shades with boundaries between $\square$ and $\{\blacksquare, \blacksquare\}$



Mark-checking trees

Define $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^2} \times \mathcal{T}$:

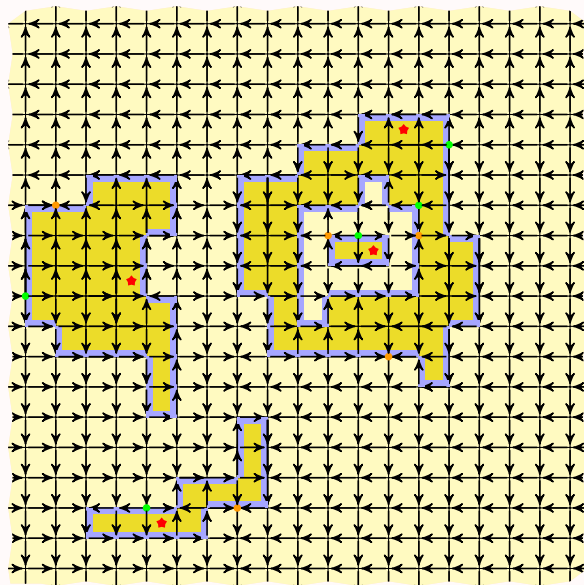
- Superimpose **shades with boundaries** between $\square$ and $\{\blacksquare, \blacksquare\}$
- Decorate arrows around $\blacksquare$ and $\blacksquare$ so that they **count occurrences of $\blacksquare$** among 0, 1, or " ≥ 2 ".



$$a+b+c+d=-1$$



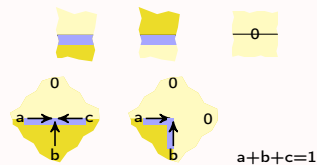
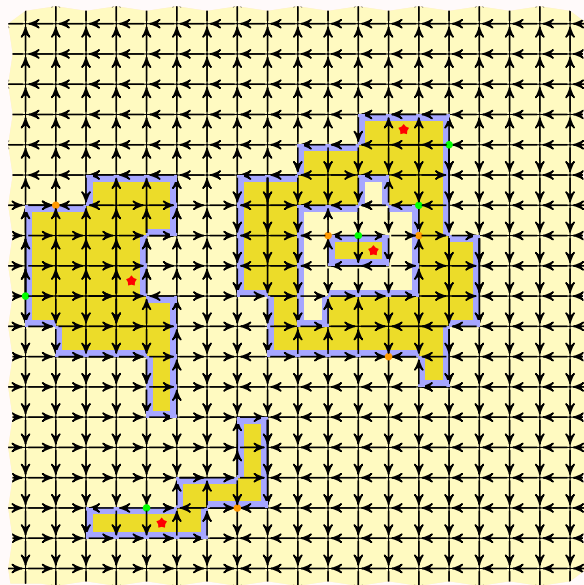
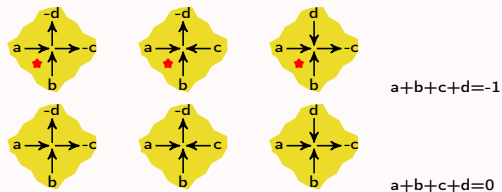
$$a+b+c+d=0$$



Mark-checking trees

Define $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^2} \times \mathcal{T}$:

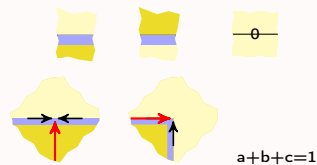
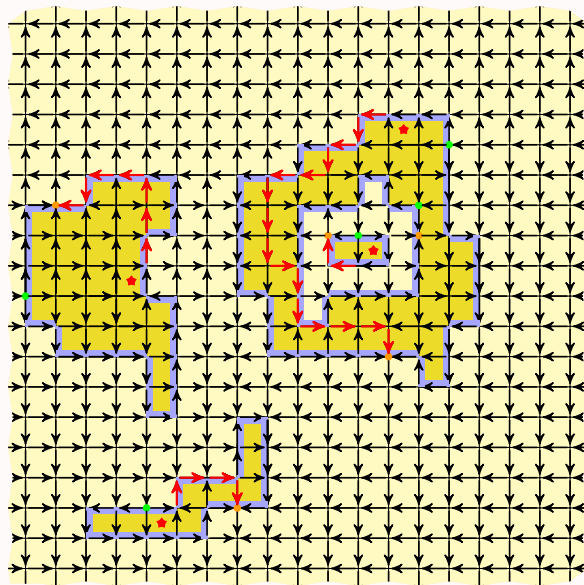
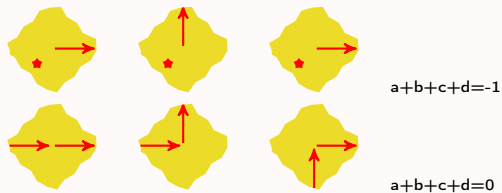
- Superimpose **shades with boundaries** between $\square$ and $\{\blacksquare, \blacksquare\}$
- Decorate arrows around $\blacksquare$ and $\blacksquare$ so that they **count occurrences of $\blacksquare$** among 0, 1, or " ≥ 2 ".
- **Exit** must **sum to 1**.



Mark-checking trees

Define $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^2} \times \mathcal{T}$:

- Superimpose **shades with boundaries** between $\square$ and $\{\blacksquare, \blacksquare\}$
- Decorate arrows around $\blacksquare$ and $\blacksquare$ so that they **count occurrences of $\blacksquare$** among 0, 1, or " ≥ 2 ".
- **Exit** must **sum to 1**.

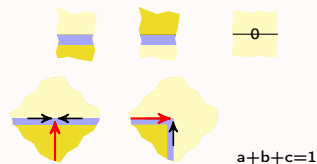
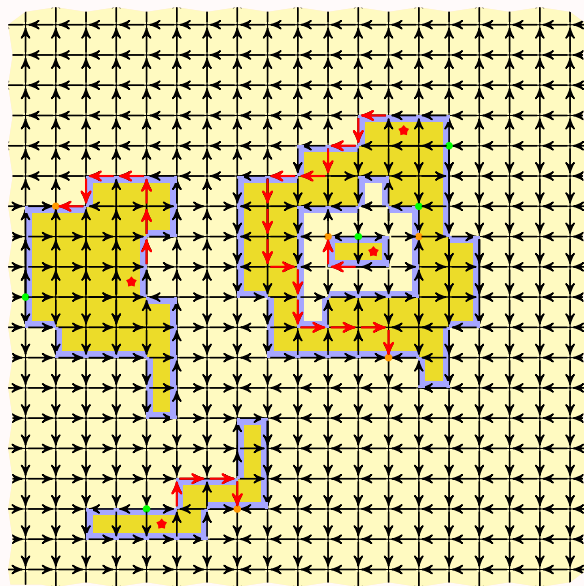
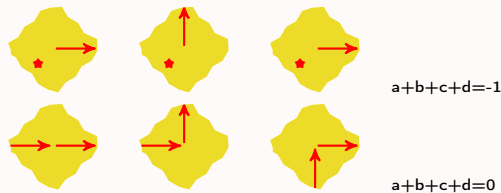


Mark-checking trees

Define $\mathcal{Z} \subset \mathcal{W} \times \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^2} \times \mathcal{T}$:

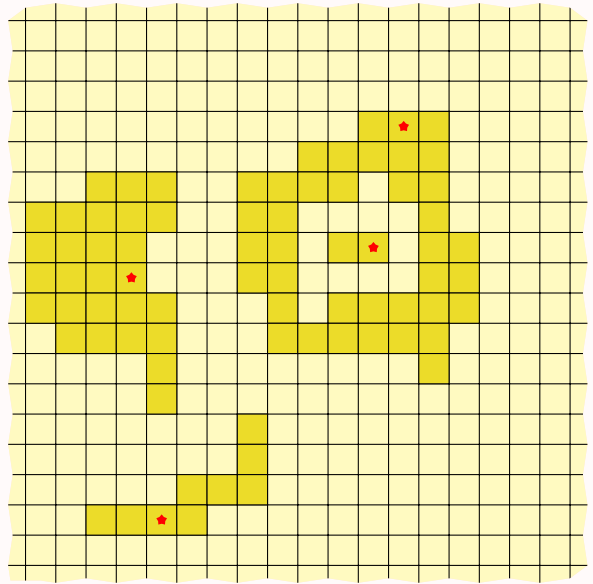
- Superimpose **shades with boundaries** between $\square$ and $\{\blacksquare, \blacksquare\}$
- Decorate arrows around $\blacksquare$ and $\blacksquare$ so that they **count occurrences of $\blacksquare$** among 0, 1, or " ≥ 2 ".
- **Exit** must **sum to 1**.

Lem: $\pi_2(\mathcal{Z}) \subseteq \mathcal{Y}'$.



Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

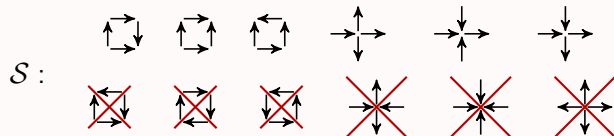
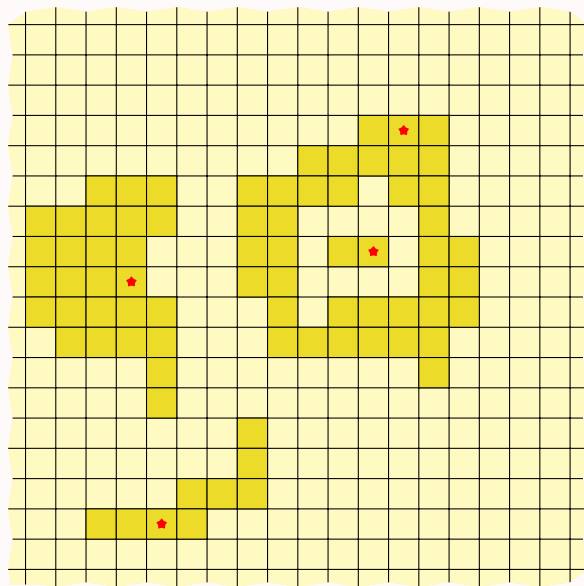


Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.

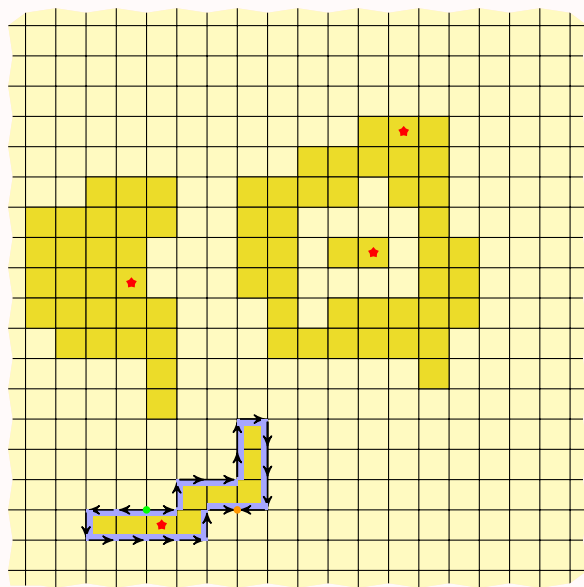


Decorating configurations

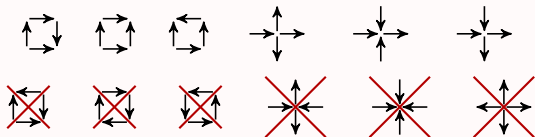
Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:



$\mathcal{S} :$



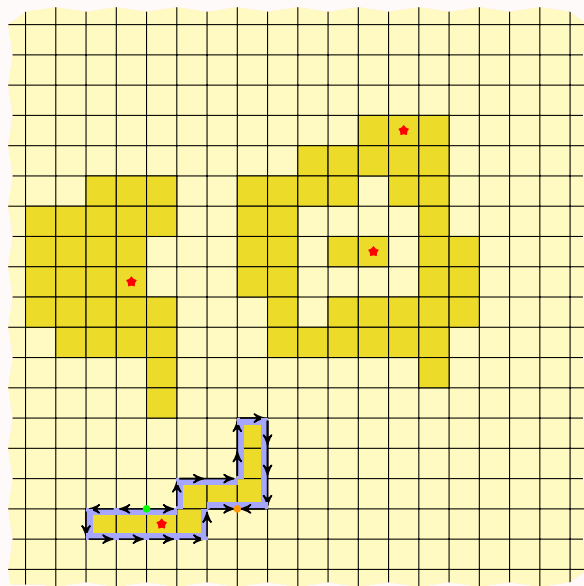
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

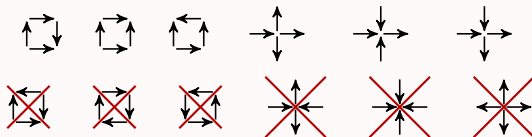
Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



$\mathcal{S} :$



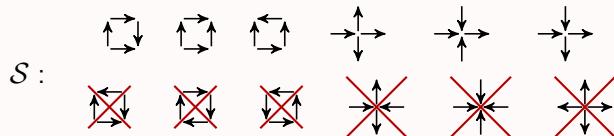
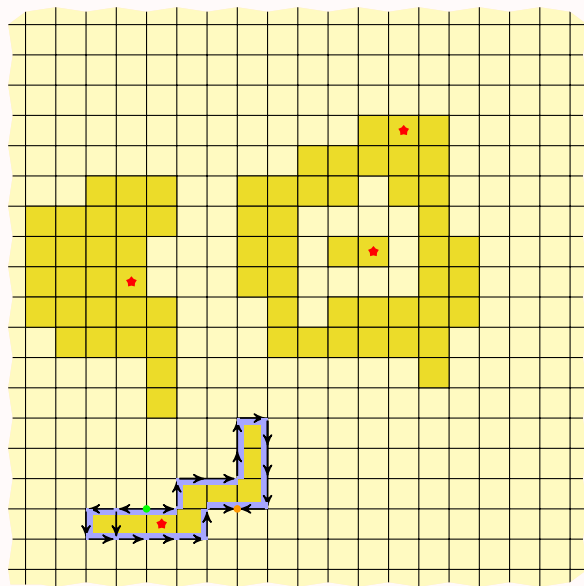
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



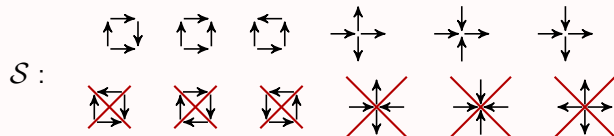
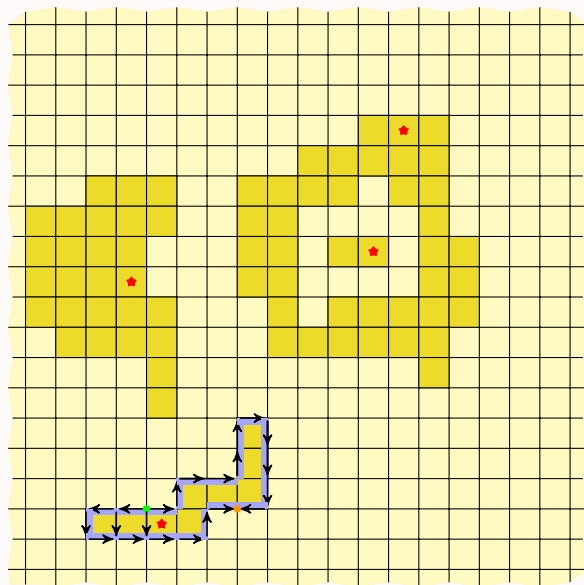
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



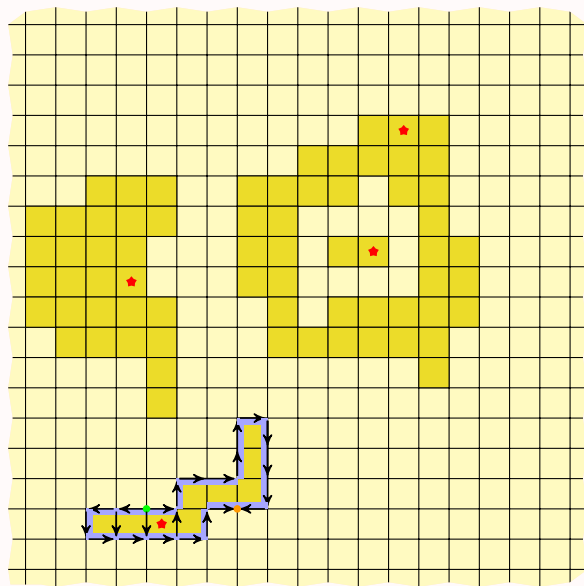
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

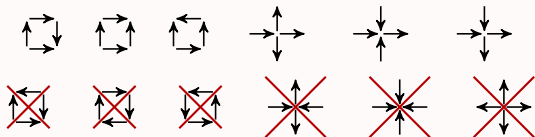
Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



$\mathcal{S} :$



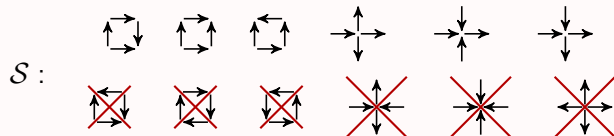
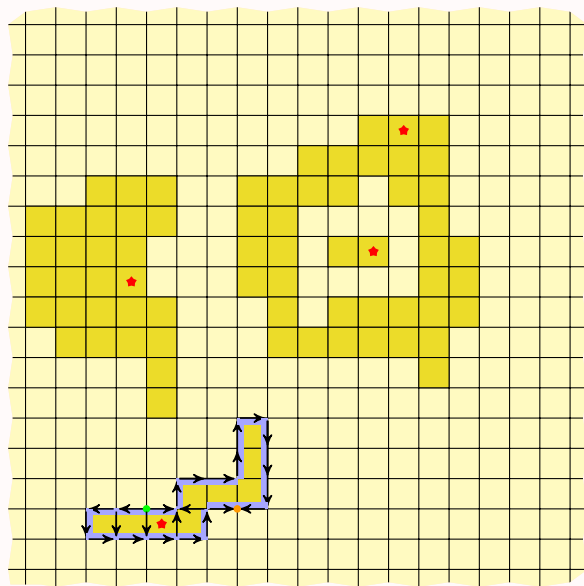
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



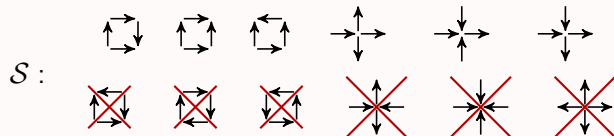
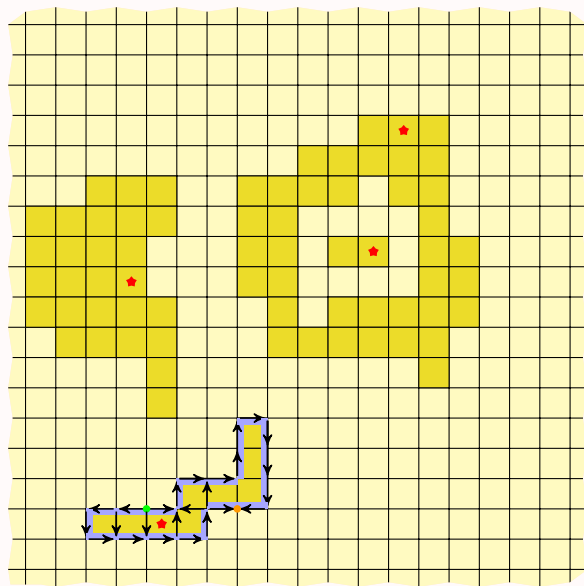
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



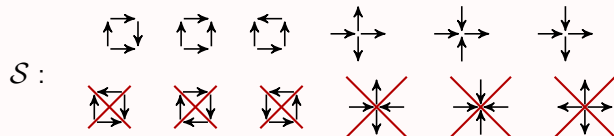
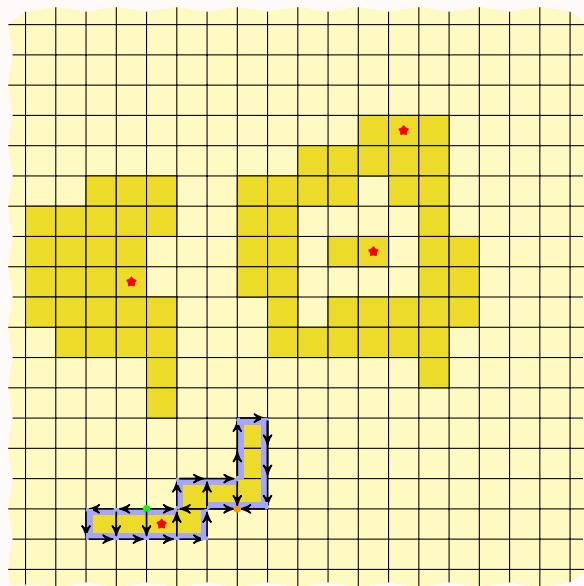
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



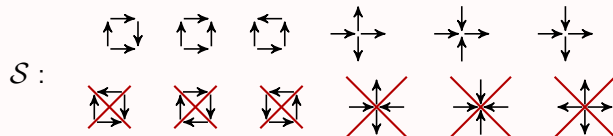
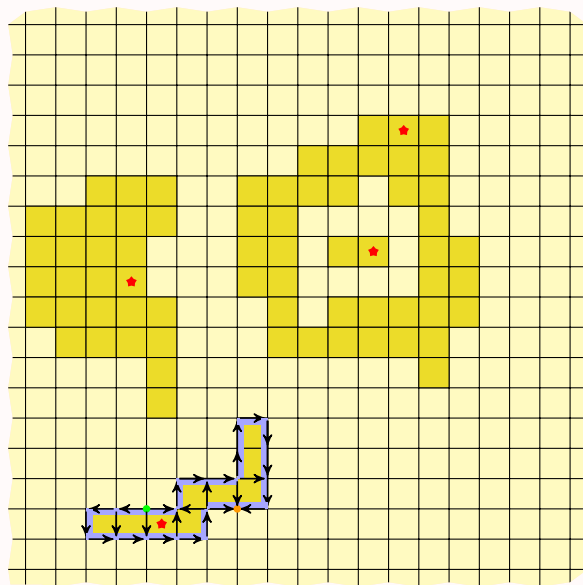
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



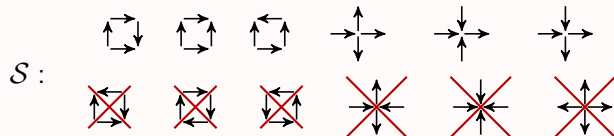
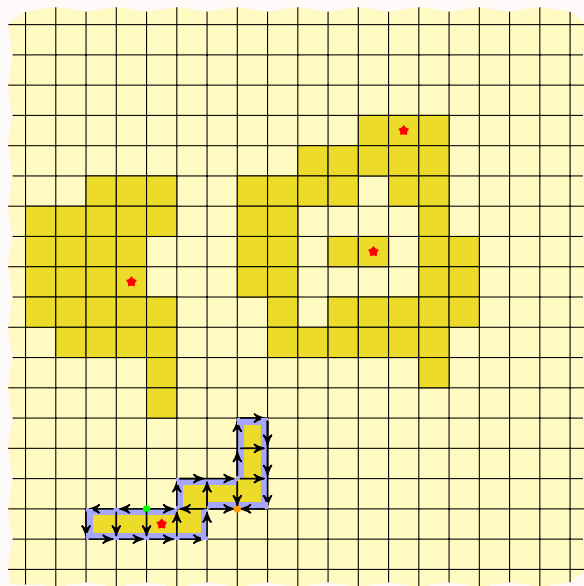
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



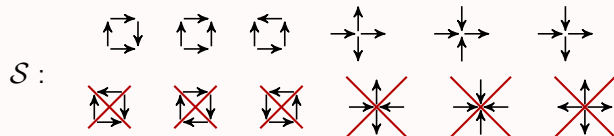
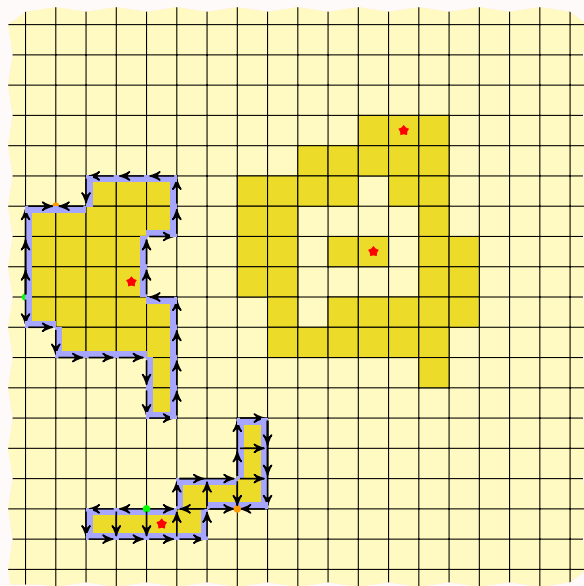
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



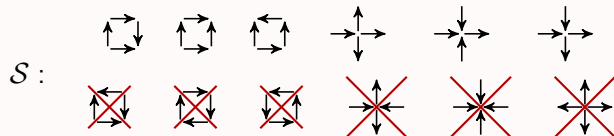
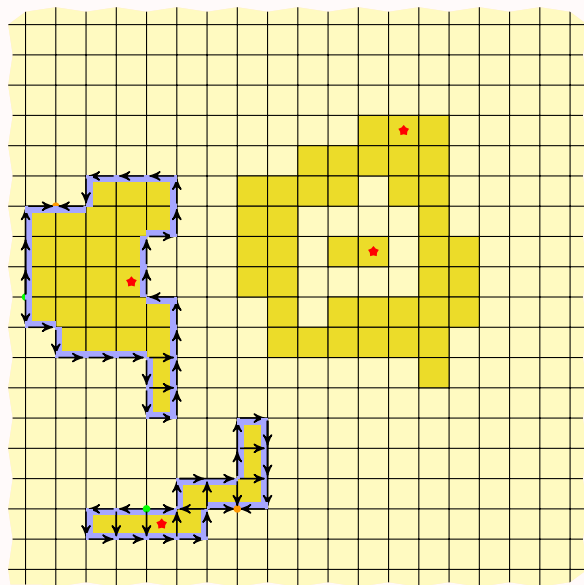
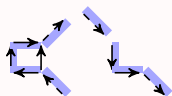
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



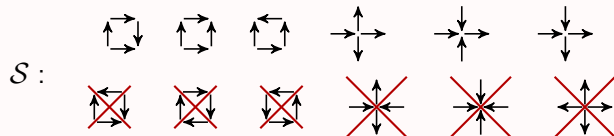
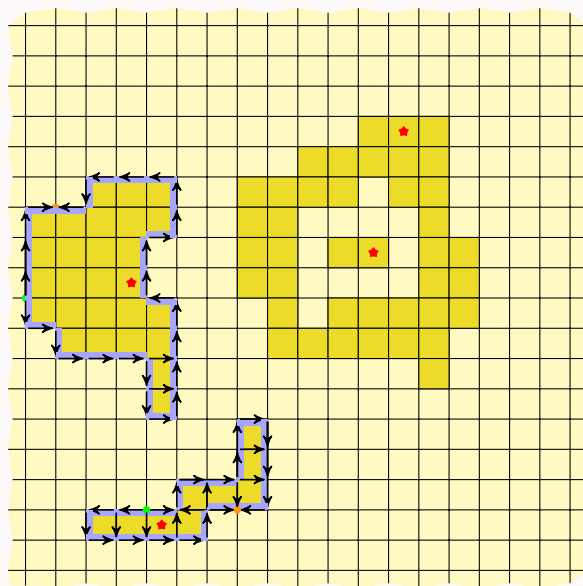
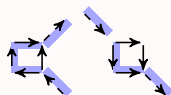
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



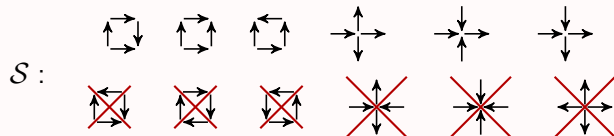
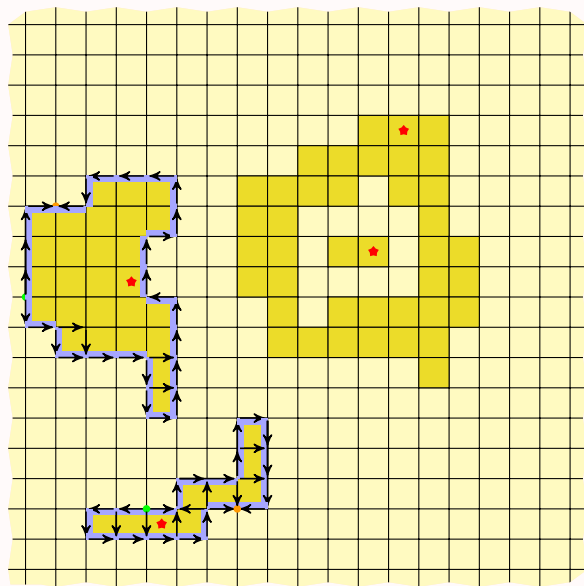
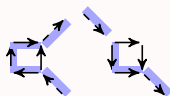
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



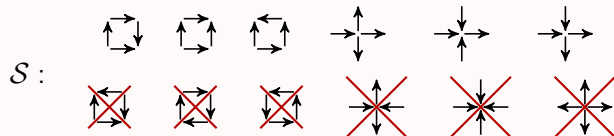
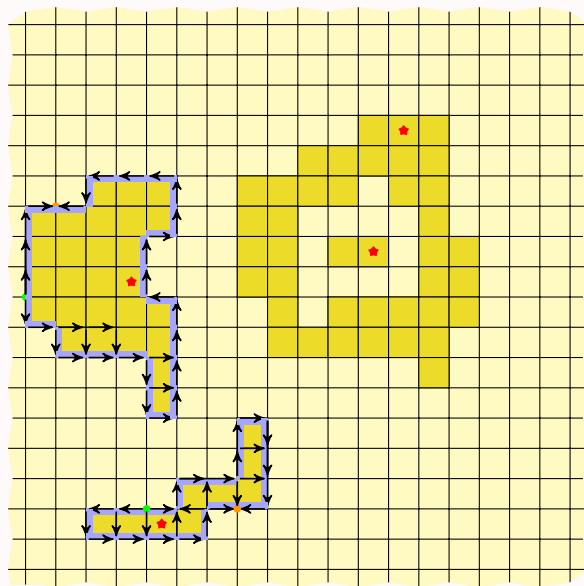
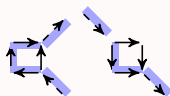
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



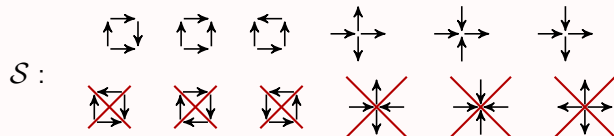
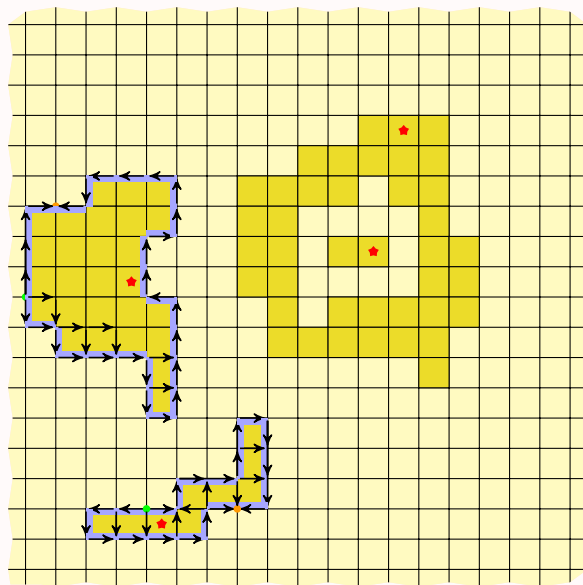
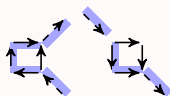
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



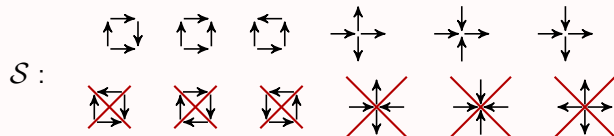
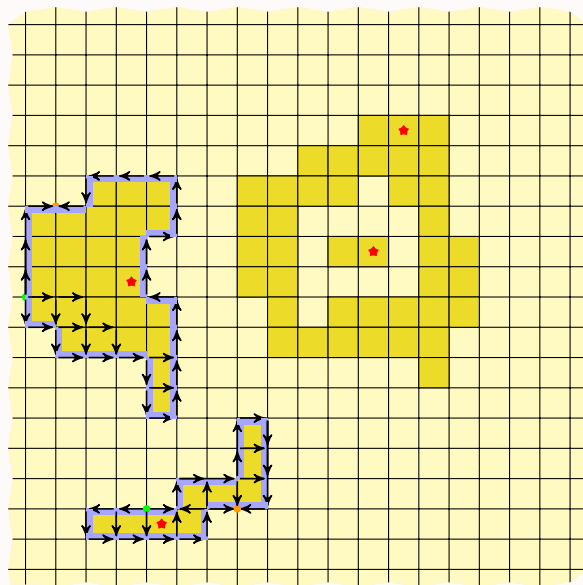
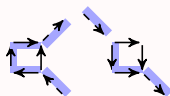
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



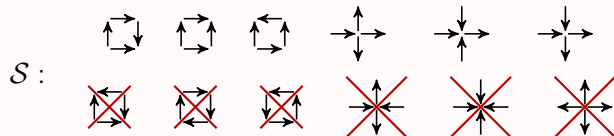
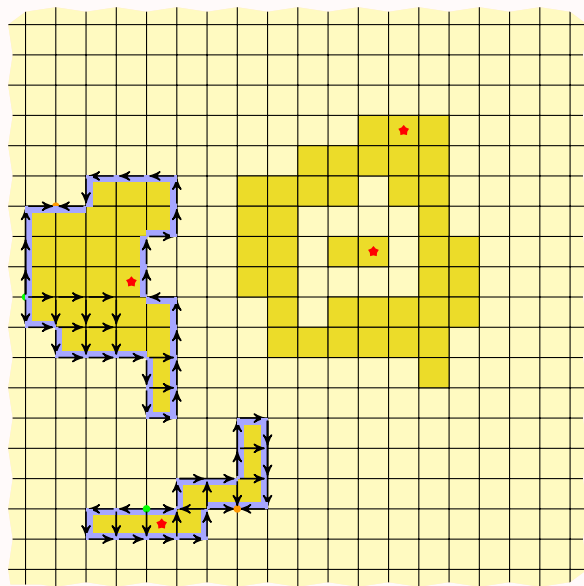
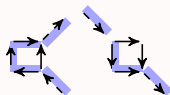
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



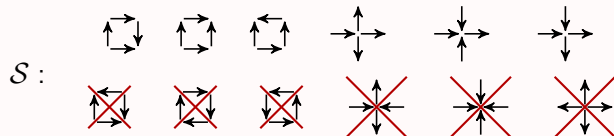
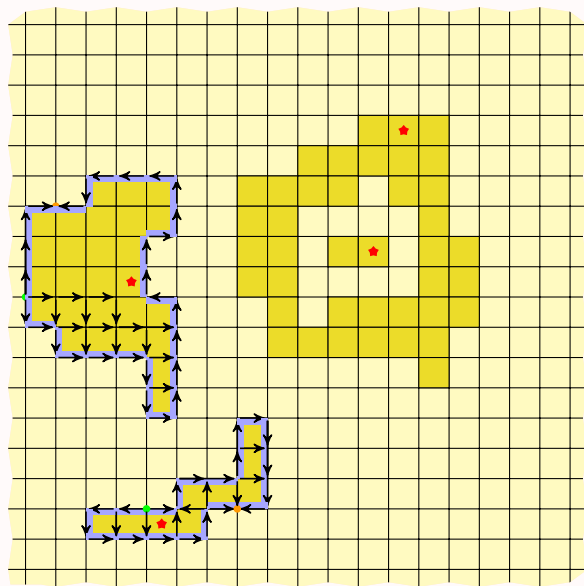
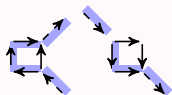
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



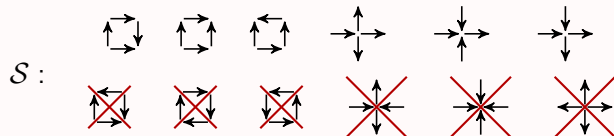
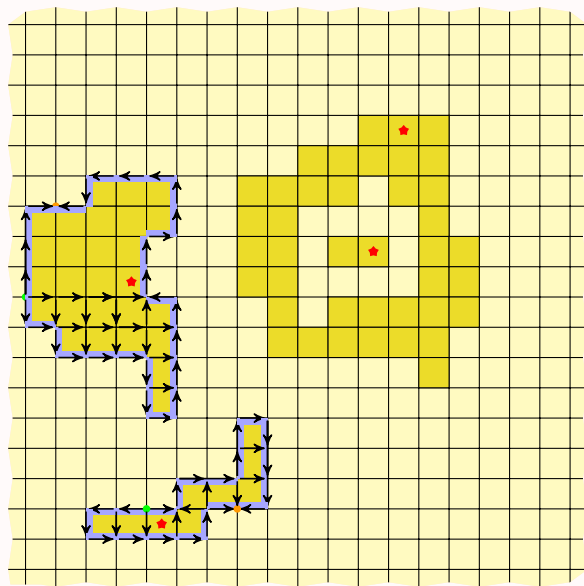
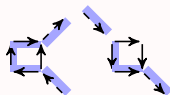
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



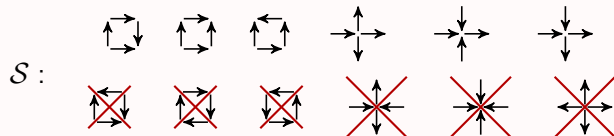
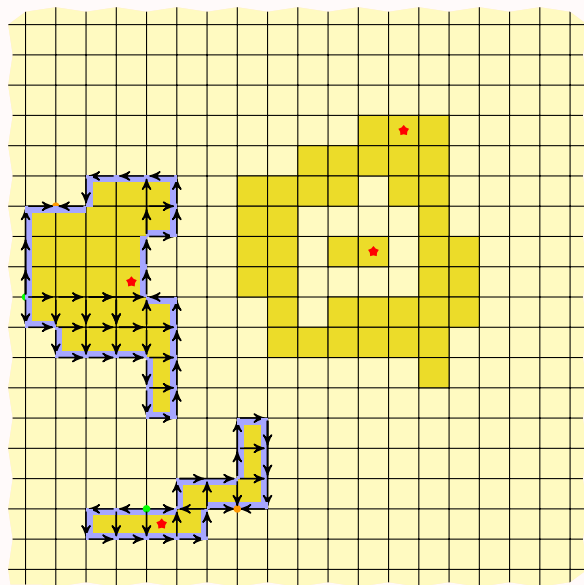
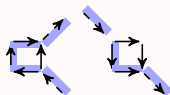
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



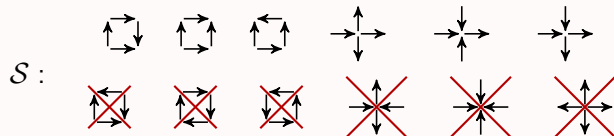
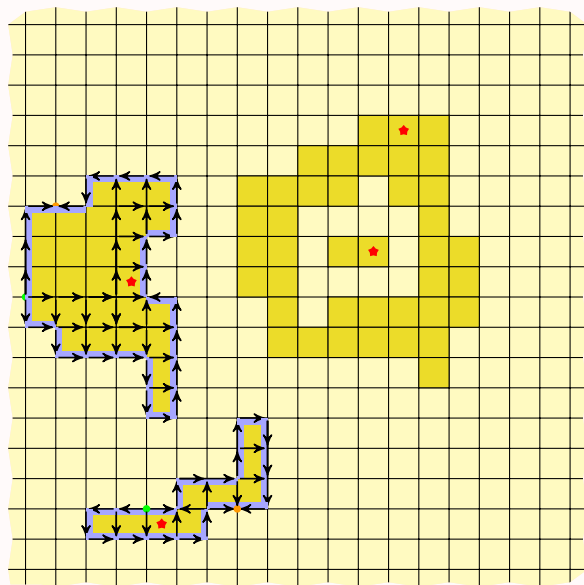
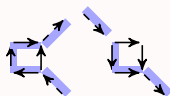
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



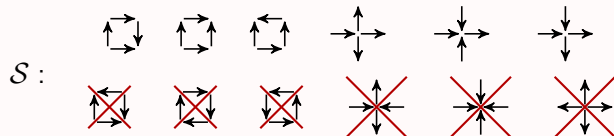
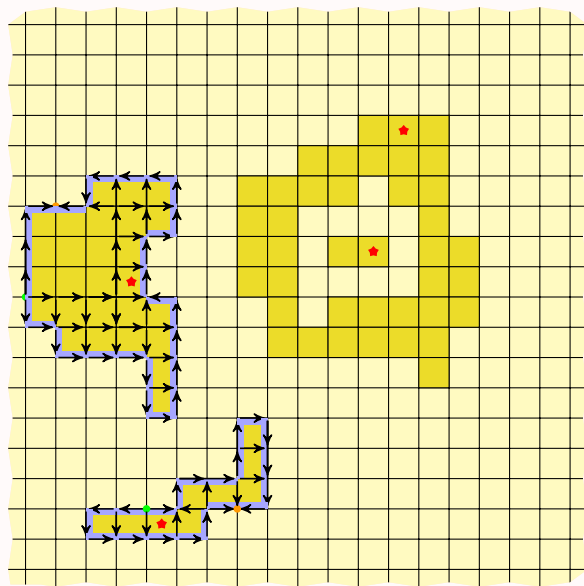
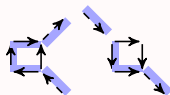
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



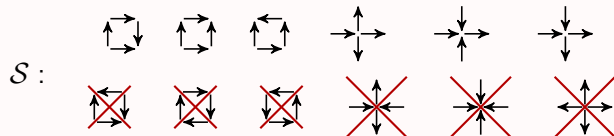
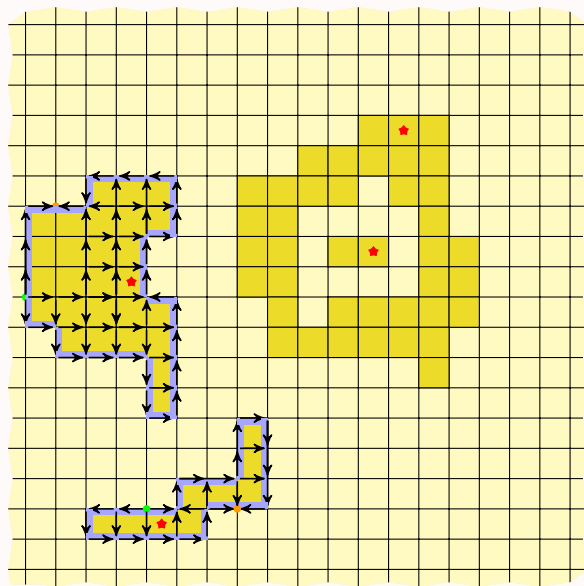
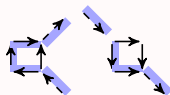
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



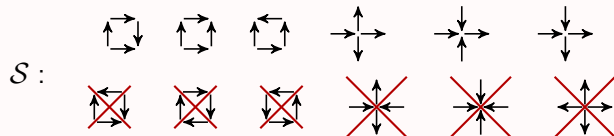
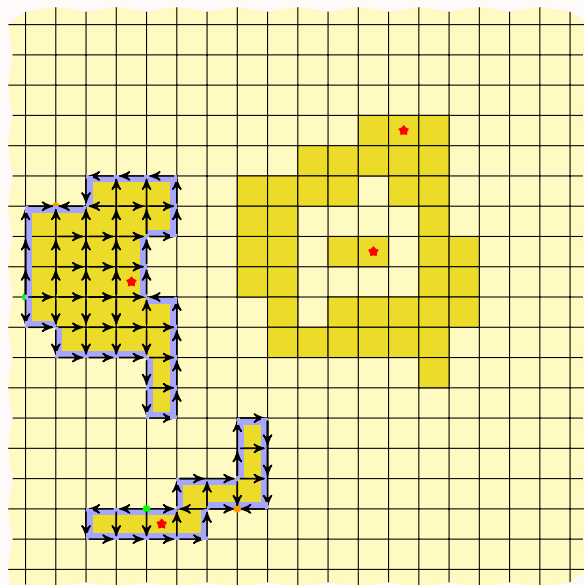
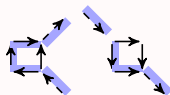
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



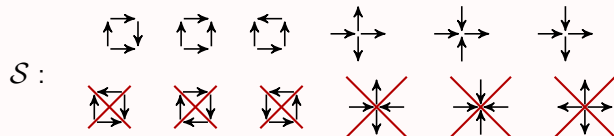
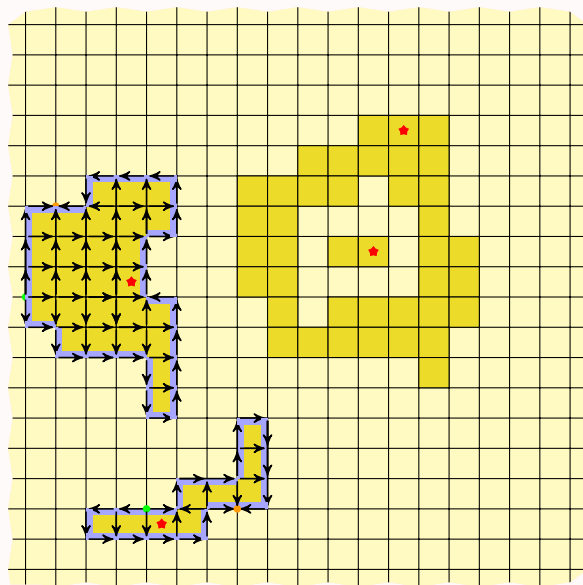
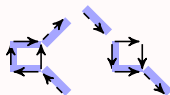
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



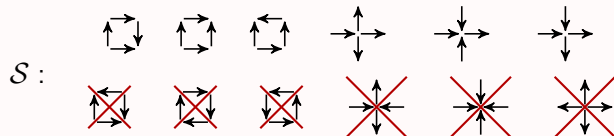
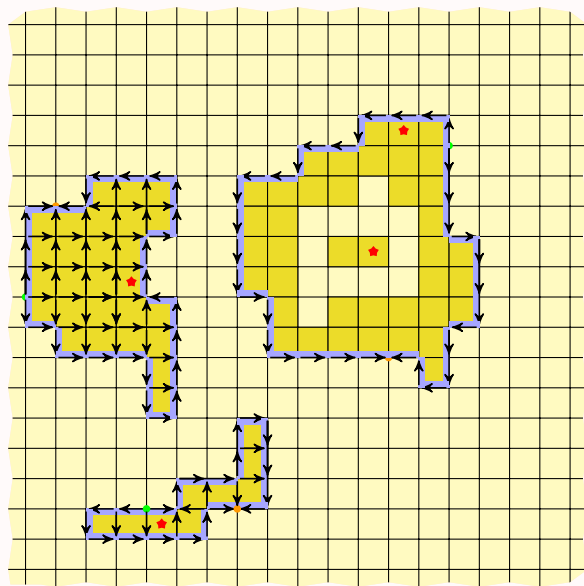
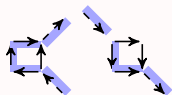
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



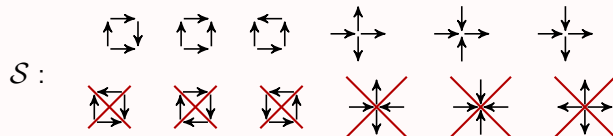
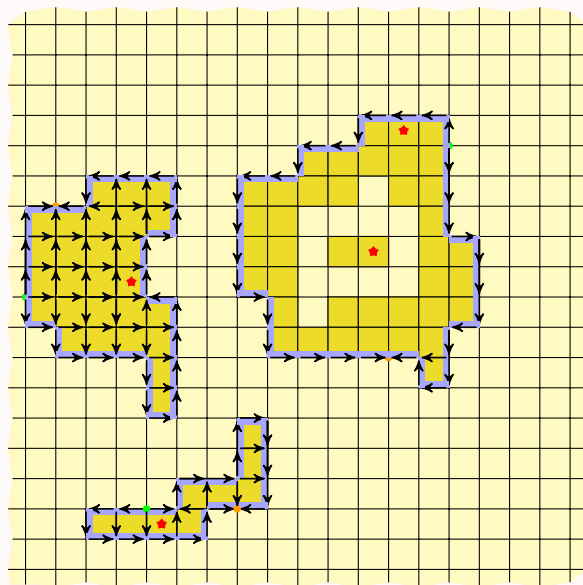
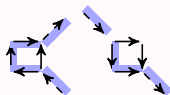
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



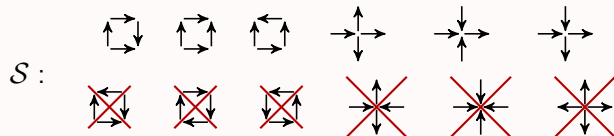
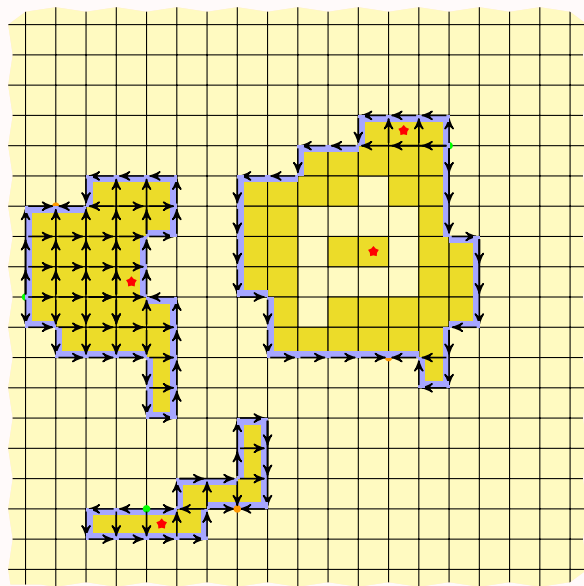
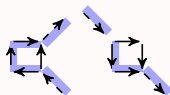
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



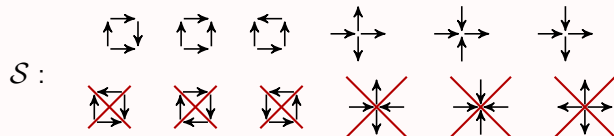
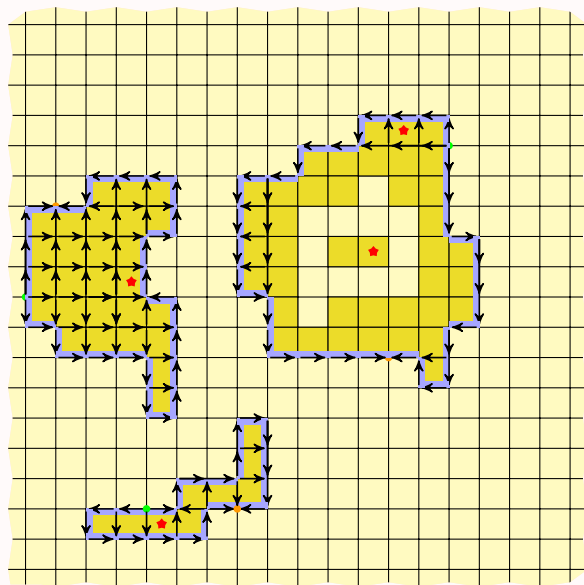
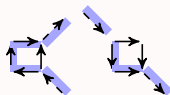
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



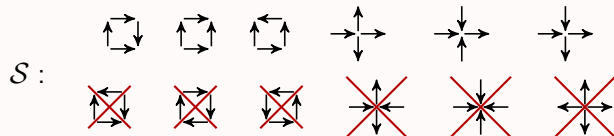
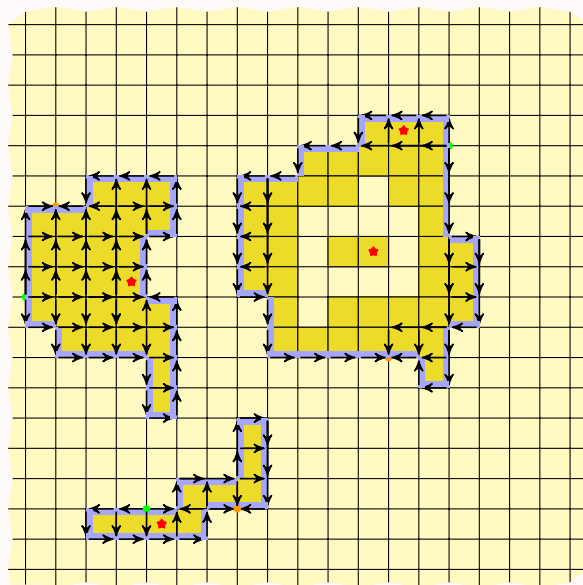
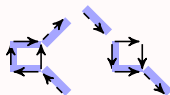
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



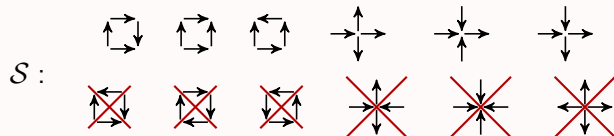
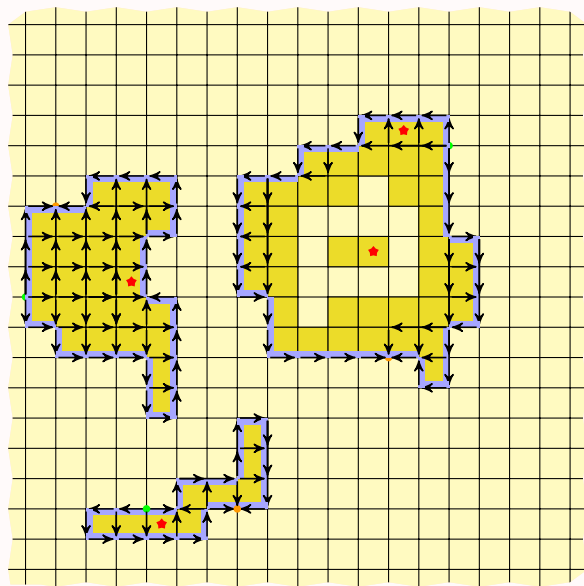
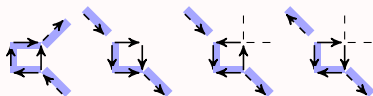
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



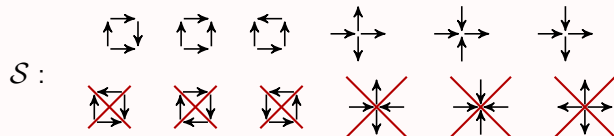
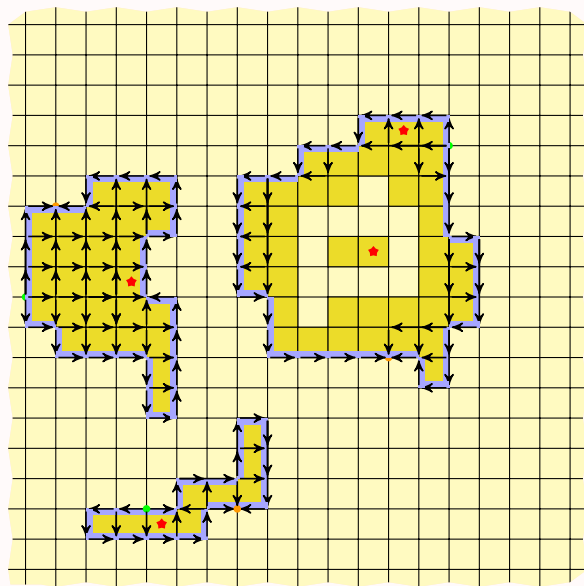
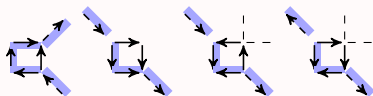
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



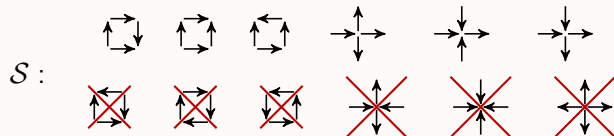
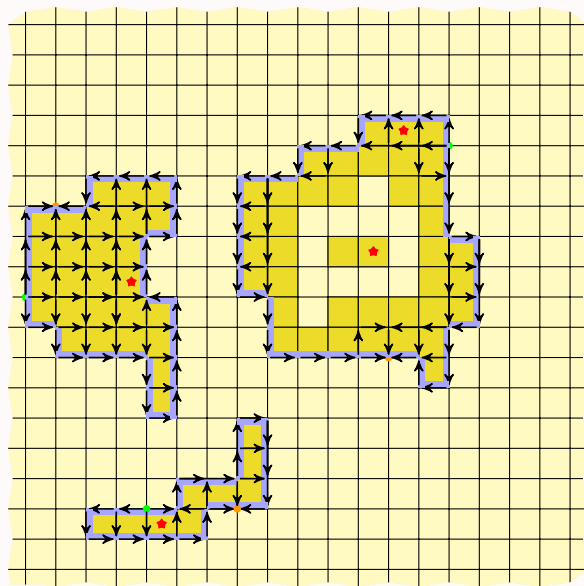
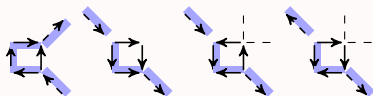
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:



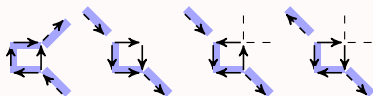
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

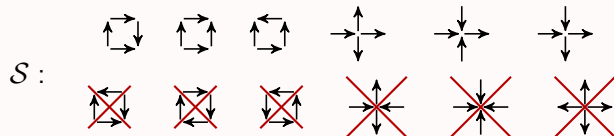
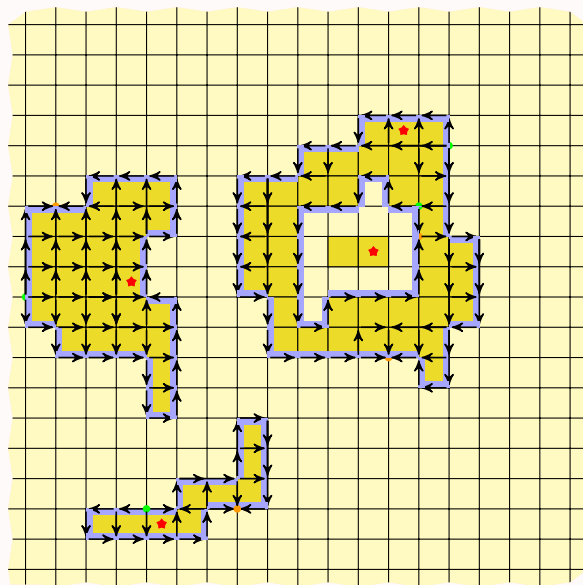
► Tile corners:



► Dig to inner region with



and impose its entrance and exit here.



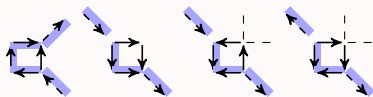
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

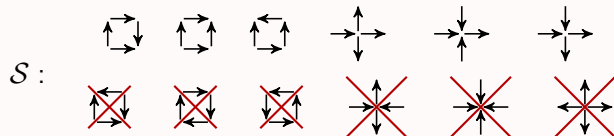
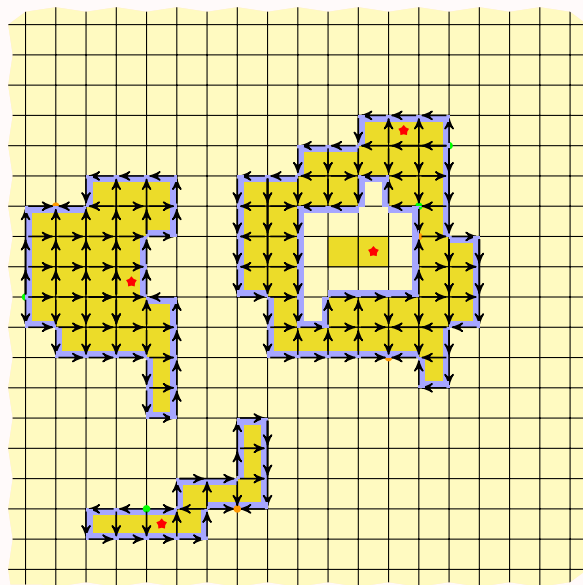
► Tile corners:



► Dig to inner region with



and impose its entrance and exit here.



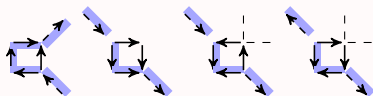
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

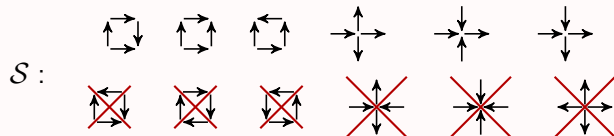
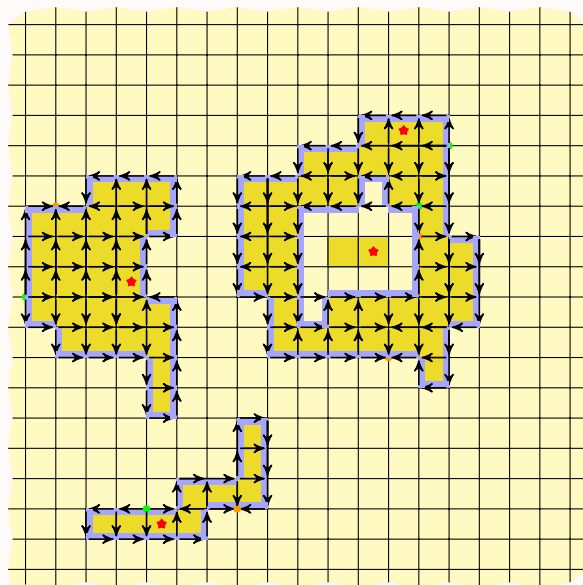
► Tile corners:



► Dig to inner region with



and impose its entrance and exit here.



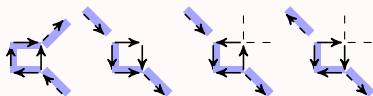
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

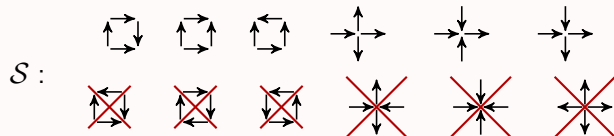
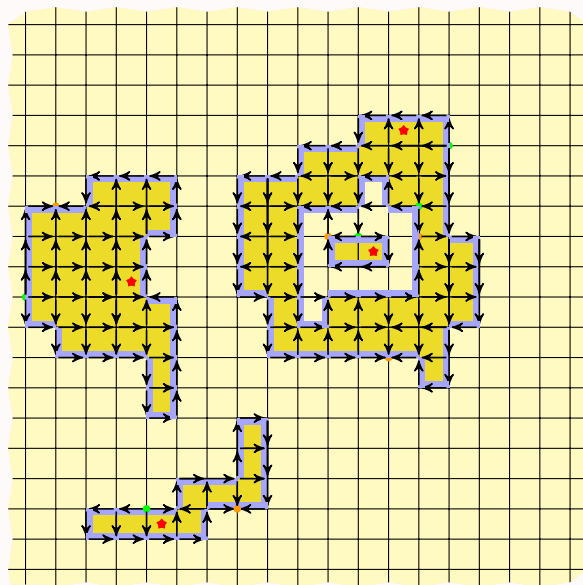
► Tile corners:



► Dig to inner region with



and impose its entrance and exit here.



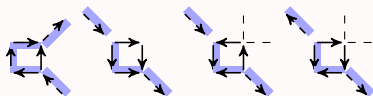
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

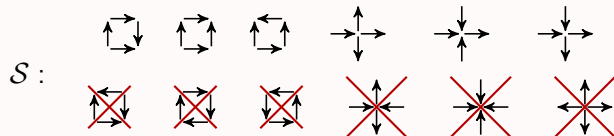
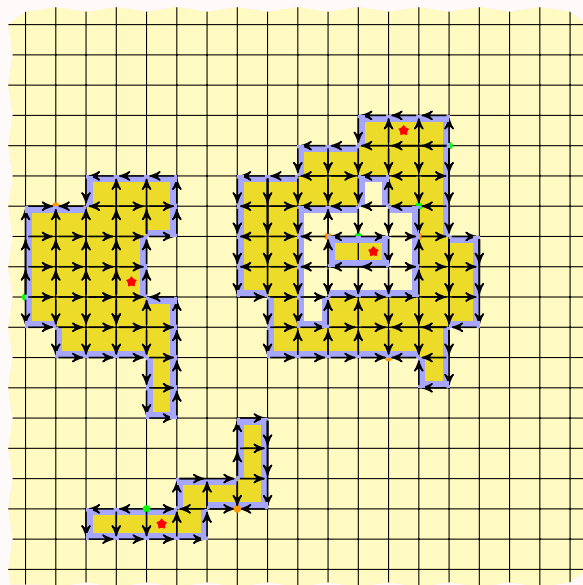
► Tile corners:



► Dig to inner region with



and impose its entrance and exit here.



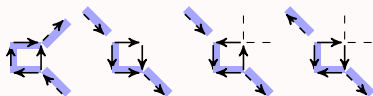
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

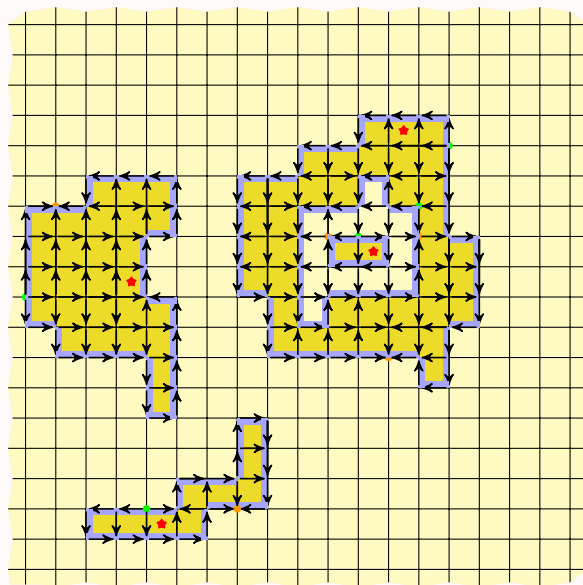
► Tile corners:



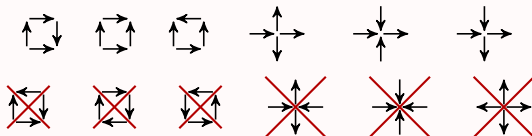
► Dig to inner region with



and impose its entrance and exit here.



$\mathcal{S}$:



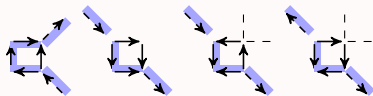
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:

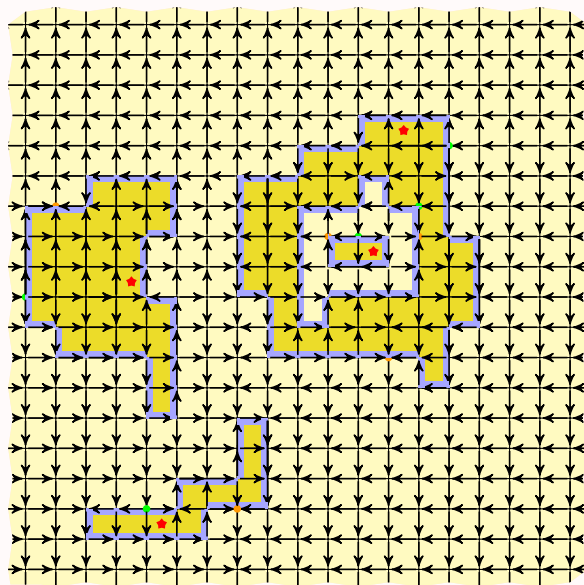


► Dig to inner region with

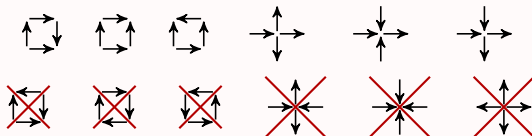


and impose its entrance and exit here.

Compactness.



$\mathcal{S}$:



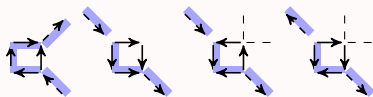
Decorating configurations

Lem: $\pi_2(\mathcal{Z}) = \mathcal{Y}'$.

Proof: Induction: any finite region, for fixed entrance and exit, is tileable.

- 1-tile region with 2 changes of directions = tile.
- For larger regions:

► Tile corners:

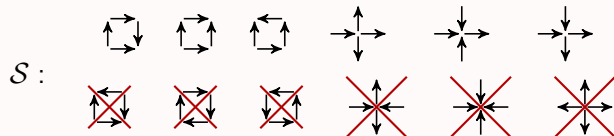
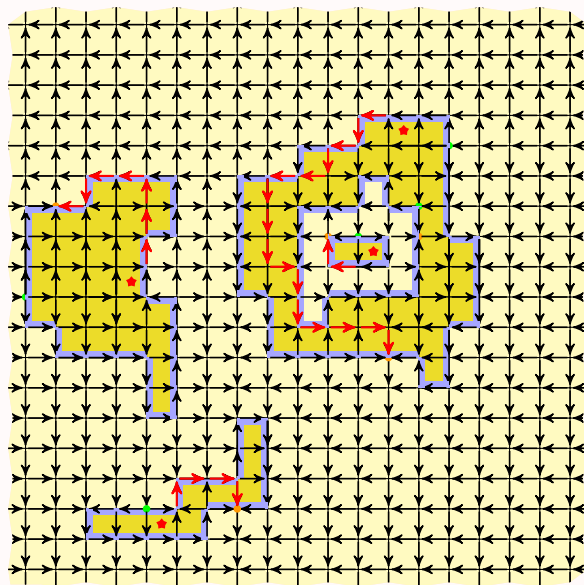


► Dig to inner region with

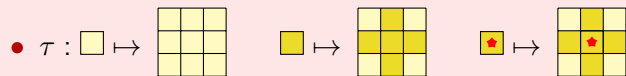


and impose its entrance and exit here.

Compactness.



Allowing checkerboard corners



- $\tau(\mathcal{Y})$ forbids vertex .

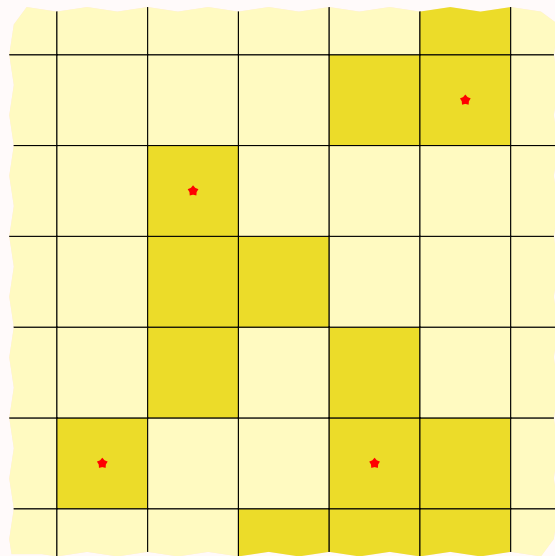
- $\tau(\mathcal{Y}) = \mathcal{Y} \cap \tau \left(\{\square, \square, \square\}^{\mathbb{Z}^2} \right).$

Lem: $\tau^{-1}(\mathcal{Y}') = \mathcal{Y}.$

Proof:

- τ sends a component to a component.
- τ sends a mark to a mark

□



Allowing checkerboard corners

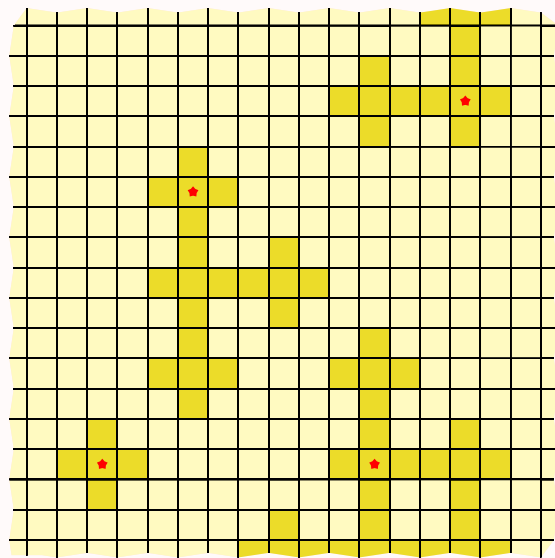
- $\tau : \square \mapsto \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \quad \square \mapsto \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \quad \star \mapsto \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \star & \square \\ \hline \square & \square & \square \\ \hline \end{array}$
- $\tau(\mathcal{Y})$ forbids vertex .
- $\tau(\mathcal{Y}) = \mathcal{Y} \cap \tau\left(\{\square, \square, \star\}^{\mathbb{Z}^2}\right)$.

Lem: $\tau^{-1}(\mathcal{Y}') = \mathcal{Y}$.

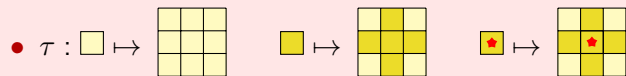
Proof:

- τ sends a component to a component.
- τ sends a mark to a mark

□



Allowing checkerboard corners



- $\tau(\mathcal{Y})$ forbids vertex .

- $\tau(\mathcal{Y}) = \mathcal{Y} \cap \tau \left(\{ \square, \square, \square \}^{\mathbb{Z}^2} \right).$

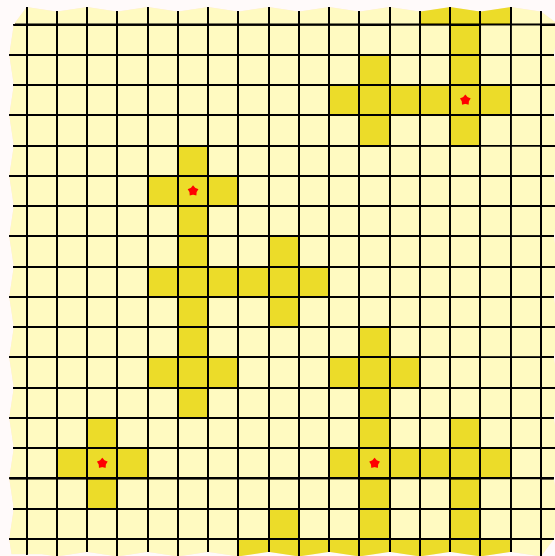
Lem: $\tau^{-1}(\mathcal{Y}') = \mathcal{Y}.$

Proof:

- τ sends a component to a component.
- τ sends a mark to a mark

□

Cor: $\mathcal{Y}$ sofic $\implies \mathcal{Y}'$ sofic.



Conclusion

Same method can handle:

- Any fixed **number of marks** per component.
- **Odd** number of marks per component.
- **Odd-component** shift [Cassaigne, 2010].
- Check that occurrences of some pattern are in some **semi-regular set**, in each component.
- Check that components are in some “**tree-regular**” picture language.
- **Genus** of components in some semi-regular set. . .

Tools:

- Force **acyclicity**.
- Simulate **tree automata**.
- Tell the **inside** from the **outside** from a component.
- “Tell” the **infiniteness** of the component.
- Tell **simple connectedness**. . .

Questions

- Shift of the configurations whose connected components have ≤ 1 ?
- $\mathbb{Z}^3$: the 1-marked-component is not sofic.
Odd-component shift?
- In which **groups** can these methods be used?
- **Equal-entropy** cover?

