Snakes and Cellular Automata Reductions and Inseparability Results

Jarkko Kari

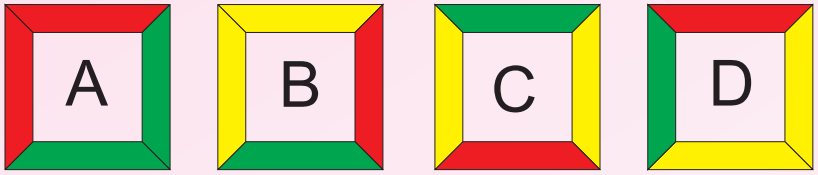
Department of Mathematics, University of Turku, Finland

TUCS (Turku Centre for Computer Science)

Wang tiles

A **Wang tile** is a unit size square tile with colored edges.

A **tile set** T is a finite collection of such tiles:

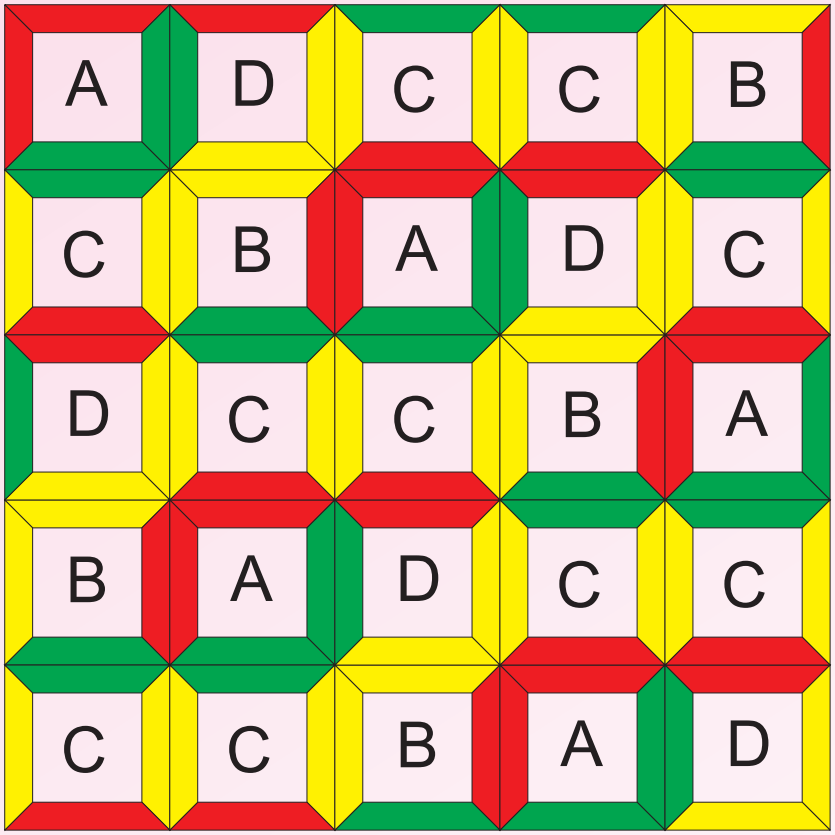


A valid tiling is an assignment

$$\mathbb{Z}^2 \longrightarrow T$$

of tiles on infinite square lattice so that the abutting edges of adjacent tiles have the same color.

For example, with copies of the given four tiles we can properly tile a 5×5 square...



... and since the colors on the borders match this square can be repeated to form a valid periodic tiling of the plane.

A decision problem:

The domino problem: Given a set T of Wang tiles, does T admit a valid tiling of the plane?

An obvious attempt to solve the domino problem:

An obvious attempt to solve the domino problem:

- Test if the 2×2 square can be tiled. If not, answer "no".

An obvious attempt to solve the domino problem:

- Test if the 2×2 square can be tiled. If not, answer "no".
- Test if the 3×3 square can be tiled. If not, answer "no".

An obvious attempt to solve the domino problem:

- Test if the 2×2 square can be tiled. If not, answer "no".
- Test if the 3×3 square can be tiled. If not, answer "no".
- Test if the 4×4 square can be tiled. If not, answer "no".
- ...

An obvious attempt to solve the domino problem:

- Test if the 2×2 square can be tiled. If not, answer "no".
- Test if the 3×3 square can be tiled. If not, answer "no".
- Test if the 4×4 square can be tiled. If not, answer "no".
- ...

If T does not admit a tiling, then eventually we encounter a number n such that T does not tile the $n \times n$ square, and we get the correct"no"-answer.

An obvious attempt to solve the domino problem:

- Test if the 2×2 square can be tiled. If not, answer "no".
- Test if the 3×3 square can be tiled. If not, answer "no".
- Test if the 4×4 square can be tiled. If not, answer "no".
- ...

If T does not admit a tiling, then eventually we encounter a number n such that T does not tile the $n \times n$ square, and we get the correct"no"-answer.

A big problem: If T admits tiling then the process never stops. We never get the positive answer. This is only a **semi-algorithm** for non-tileability.

Second attempt: For every $n = 2, 3, 4 \ldots$, construct all possible tilings of the $n \times n$ square.

- If the $n \times n$ square can not be tiled, answer "no".
- If the $n \times n$ square can be tiled in a way that can be repeated periodically to cover the plane, answer "yes".

Now we get the correct "no" -answer if no tiling exists, and the correct "yes" -answer if a periodic tiling exists.

Second attempt: For every $n = 2, 3, 4 \ldots$, construct all possible tilings of the $n \times n$ square.

- If the $n \times n$ square can not be tiled, answer "no".
- If the $n \times n$ square can be tiled in a way that can be repeated periodically to cover the plane, answer "yes".

Now we get the correct "no" -answer if no tiling exists, and the correct "yes" -answer if a periodic tiling exists.

But: there exist **aperiodic** tile sets that tile the plane only in non-periodic ways. On such cases our second attempt fails to return any answer.

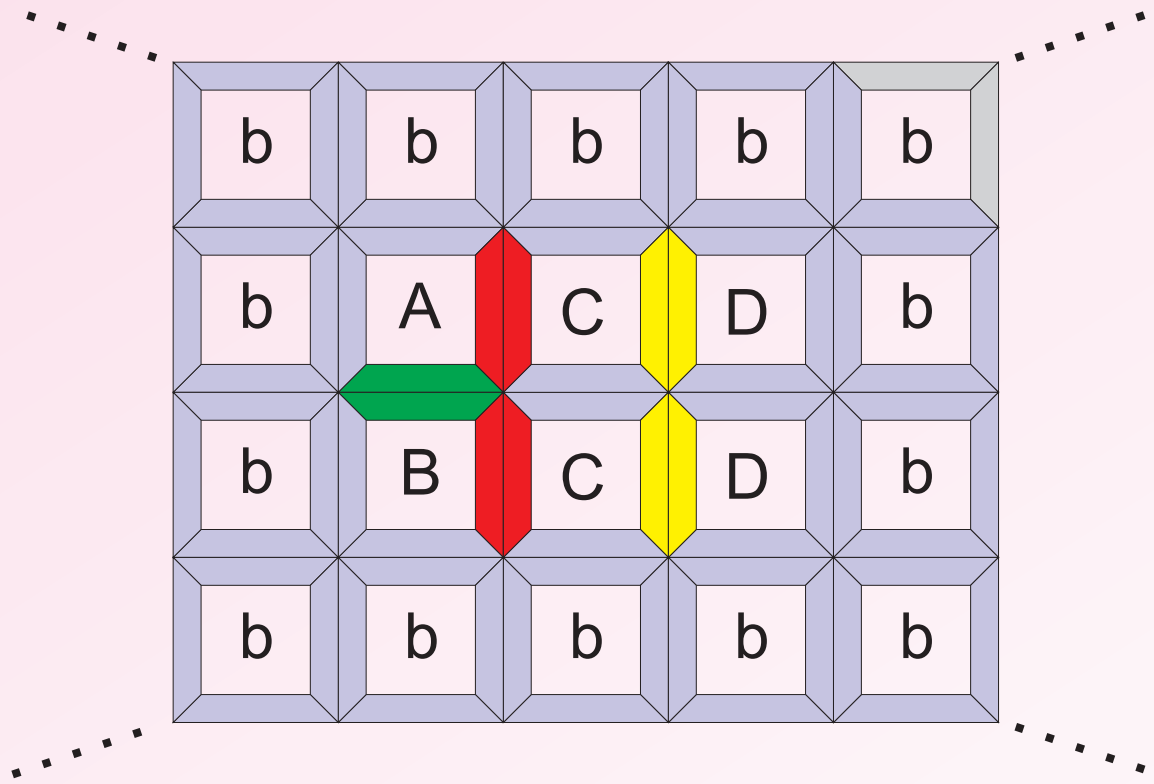
In fact, no general procedure exists:

Theorem (R. Berger 1966): The domino problem is undecidable.

As part of his proof, R. Berger also constructed the first known aperiodic tile set.

We also need the following variant of the domino problem:

The finite domino problem: Given a set T of Wang tiles, containing a specific blank tile b, does T admit a valid tiling of the plane such that all but a finite, non-zero number of tiles are blank:



Theorem. The finite domino problem is undecidable.

The proof is easy, using H.Wang's original technique to simulate Turing machines on tiles.

Remark:

- The **negative** instances of the **domino problem**
- The **positive** instances of the **finite domino problem**

are semi-decidable.

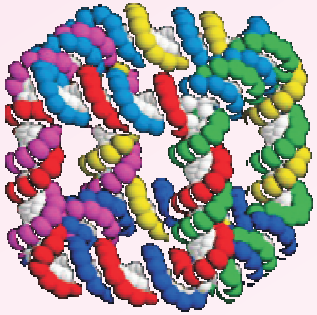
Outline of the talk

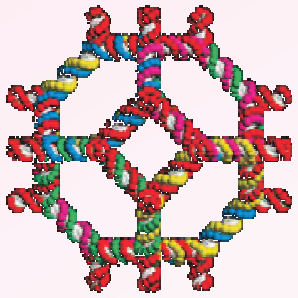
1. Wang tiles and the Domino problem ✓
2. Two applications
   (A) Self-assembly
      - Termination problem
      - Snake tiling problem
      - Loop tiling problem
      - Recursive inseparability
   (B) Cellular automata
      - Reversibility and surjectivity
      - Inseparability

Self-assembly

- **Self-assembly** = the process by which objects autonomously come together to form complex structures

- **Examples**
  - **Atoms** bind by chemical bonds to form molecules
  - **Molecules** may form crystals or macromolecules
  - **Cells** interact to form organisms





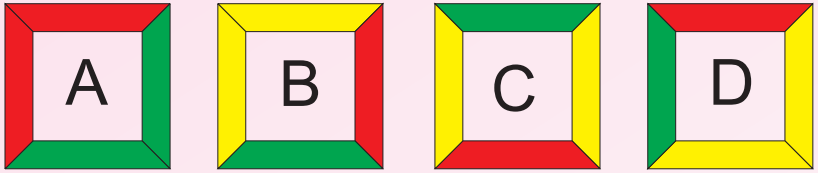
Motivation for self-assembly

- **Nanotechnology:** miniaturization in medicine, electronics, engineering, material science, manufacturing
  - **Top-down** techniques: nanolithography
  - **Bottom-up** techniques: self-assembly. Design molecules that self-assemble to form the target structure.

- **Computation using self-assembly:** In 1994 L.Adleman demonstrated that the self-assembly process of DNA molecules can be used to solve computational problems

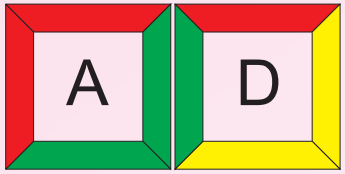
Theoretical model based on Wang tiles

A simplified theoretical model to study algorithms and complexity related to self-assembly (L. Adleman, E. Winfree).



Each **Wang tile** is a "molecule" that may participate in the assembly. Tiles are placed on the plane at integer lattice positions indexed by $\mathbb{Z}^2$. The tiles are not rotated.

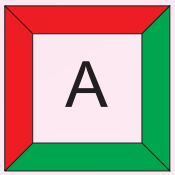
Two adjacent tiles **stick** if they have the same color at the abutting edges:



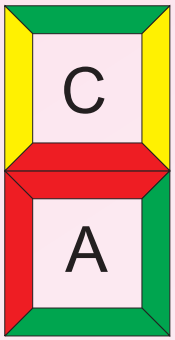
In the model,

- We have a finite tile set of different tiles. [Finite number of different "molecules" in use.]
- We have an unlimited supply of copies of the tiles available. [We do not control the precise number of molecules participating in the self-assembly process.]

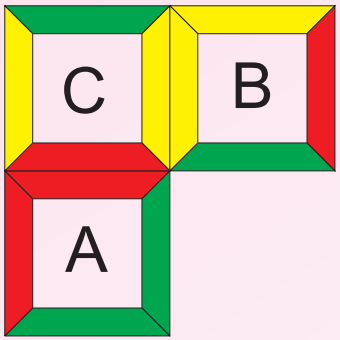
Structures self-assemble with tiles from T by incrementally adding single tiles that stick:



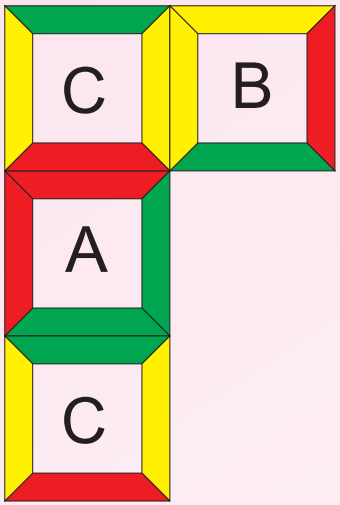
Structures self-assemble with tiles from T by incrementally adding single tiles that stick:



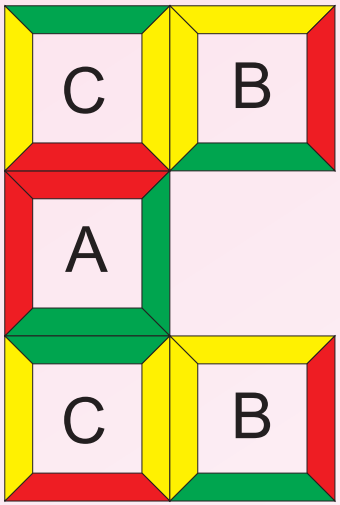
Structures self-assemble with tiles from T by incrementally adding single tiles that stick:



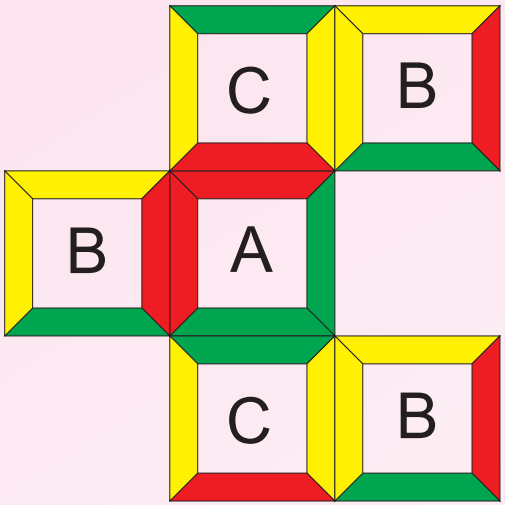
Structures self-assemble with tiles from T by incrementally adding single tiles that stick:



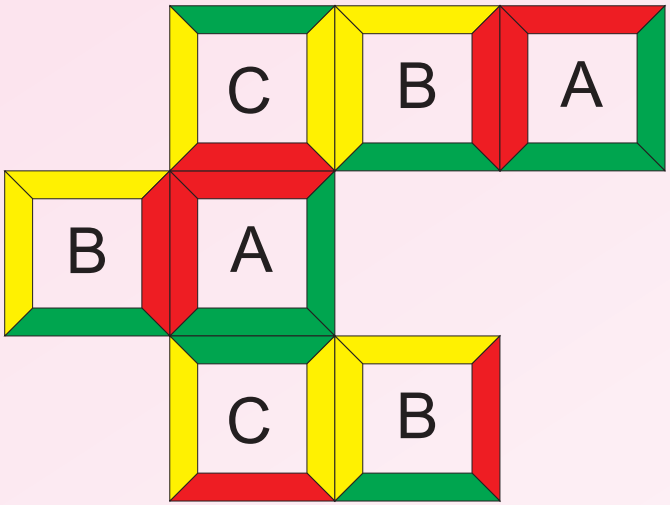
Structures self-assemble with tiles from T by incrementally adding single tiles that stick:



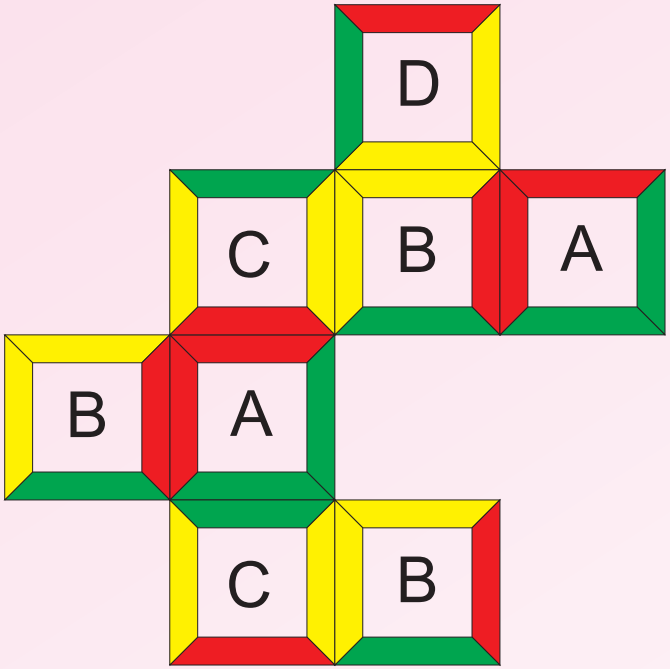
Structures self-assemble with tiles from T by incrementally adding single tiles that stick:



Structures self-assemble with tiles from T by incrementally adding single tiles that stick:

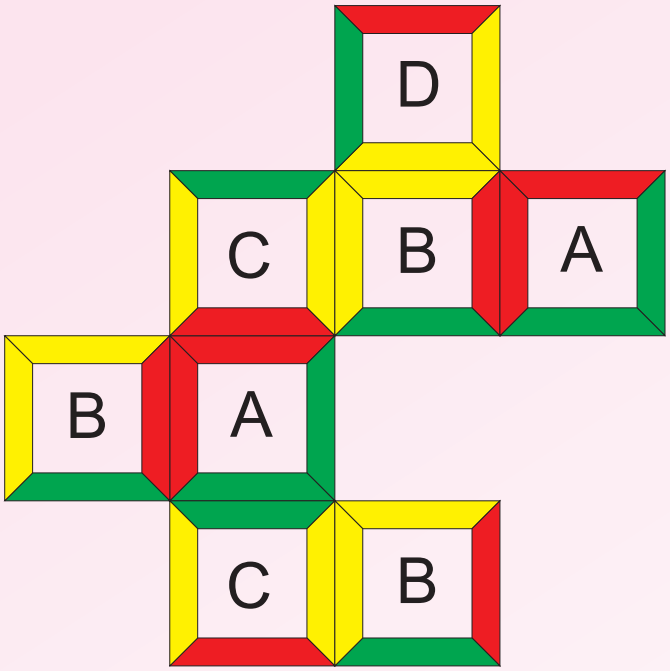


Structures self-assemble with tiles from T by incrementally adding single tiles that stick:



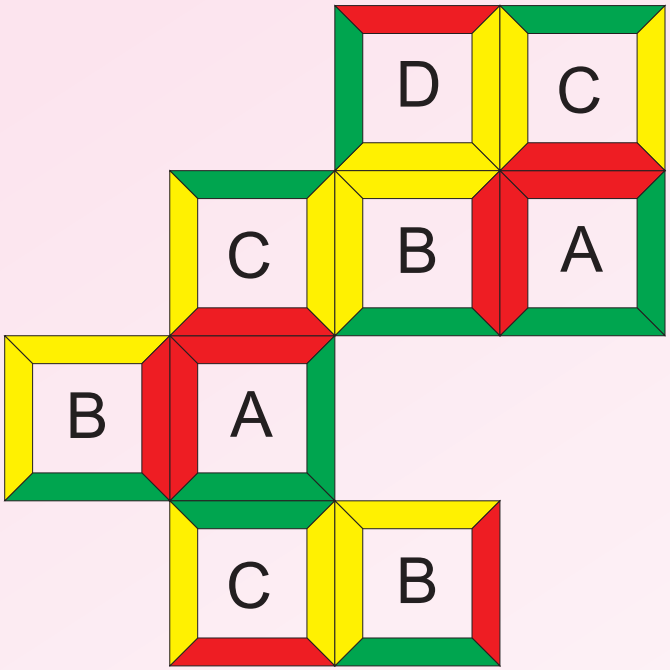
There are two natural choices of matching rules:

- Under **strong** sticking rules, each new tile must stick to **all** its neighbors that have been assembled before it.



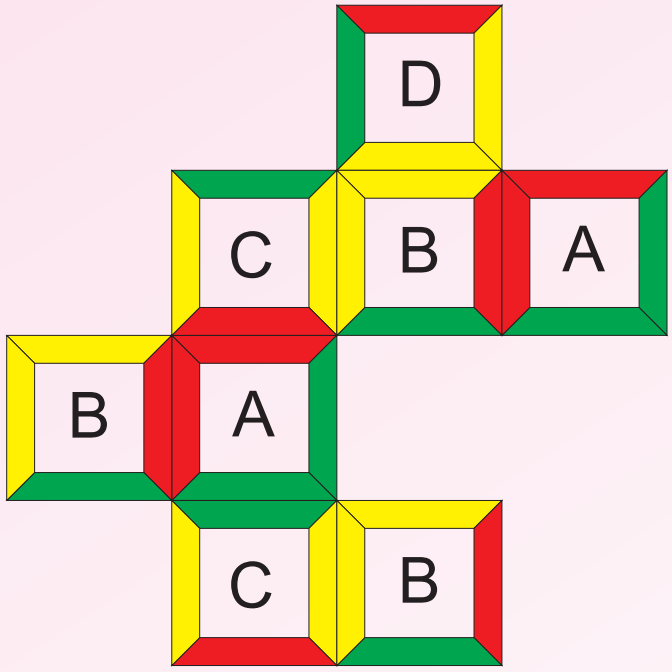
There are two natural choices of matching rules:

- Under **strong** sticking rules, each new tile must stick to **all** its neighbors that have been assembled before it.



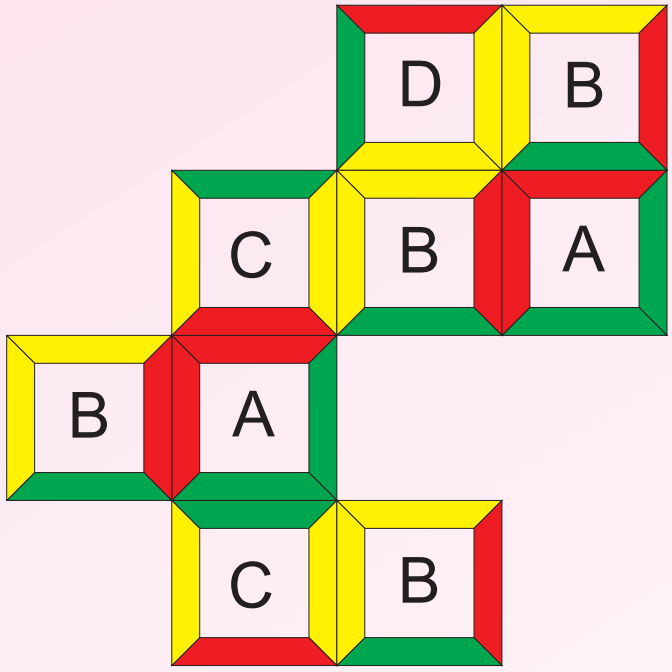
There are two natural choices of matching rules:

- Under **strong** sticking rules, each new tile must stick to **all** its neighbors that have been assembled before it.
- Under **weak** sticking rules, it is enough for a tile to stick to **one** previous tile.



There are two natural choices of matching rules:

- Under **strong** sticking rules, each new tile must stick to **all** its neighbors that have been assembled before it.
- Under **weak** sticking rules, it is enough for a tile to stick to **one** previous tile.



There are two natural choices of matching rules:

- Under **strong** sticking rules, each new tile must stick to **all** its neighbors that have been assembled before it.
- Under **weak** sticking rules, it is enough for a tile to stick to **one** previous tile.

A more general approach: assign a **strength** value to each color and say that a tile sticks if the sum of the strengths between the tile and its neighbors exceeds some given **threshold**.

The tiling model is a simplification of real self-assembly systems. However, it is already complex enough to admit universal computation.

If some questions concerning the tiling model are (computationally) difficult, then surely the same questions about real self-assembly systems are at least as difficult.

$\Longrightarrow$ The tiling model gives **Lower bounds** on computational problems on self-asssembly.

Several interesting algorithmic questions arise:

- Does a given tile set assemble a unique shape ?
- Complexity questions: How many tiles are needed in a set that self-assembles into a given shape ?
- **Can the given tile set self-assemble without limit, i.e., is unbounded growth possible ?**

The termination problem

A pattern is a **terminal assembly** if no more tiles can be added.

A tile set admits **unbounded growth** if an infinite sequence of tile additions without ever reaching a terminal assembly is possible.

The termination problem

A pattern is a **terminal assembly** if no more tiles can be added.

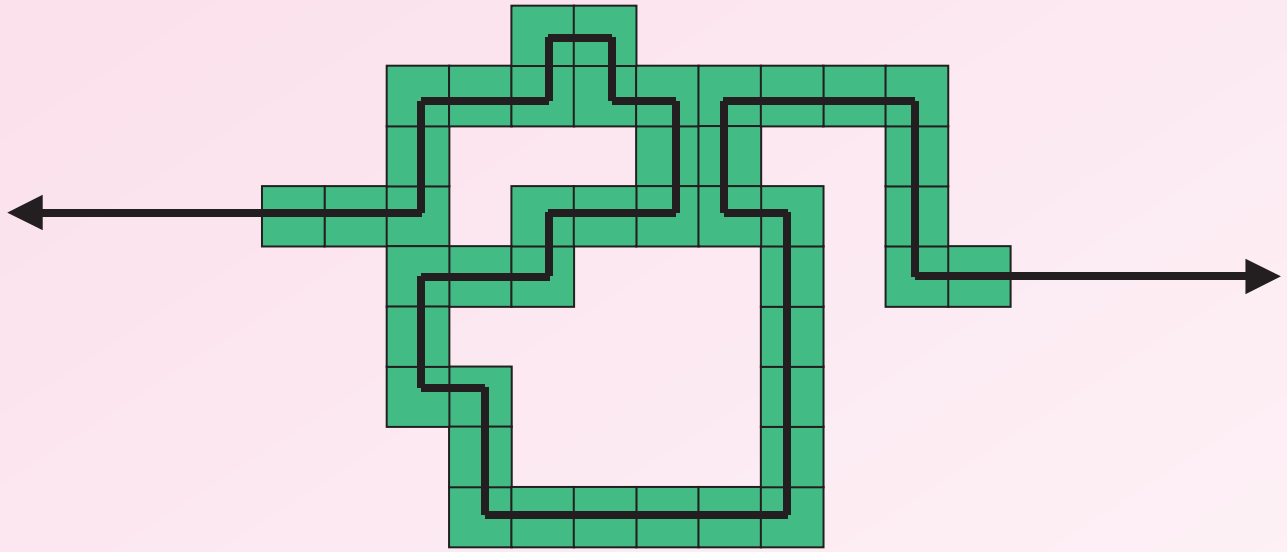
A tile set admits **unbounded growth** if an infinite sequence of tile additions without ever reaching a terminal assembly is possible.

$\Longrightarrow$ Decision problem...

The termination problem: Does a given tile set admit unbounded growth (under strong/weak sticking rules) ?

Easy to see:

- an unbounded assembly is possible if and only if the tiles admit a tiling of a (bi-infinite) snake.

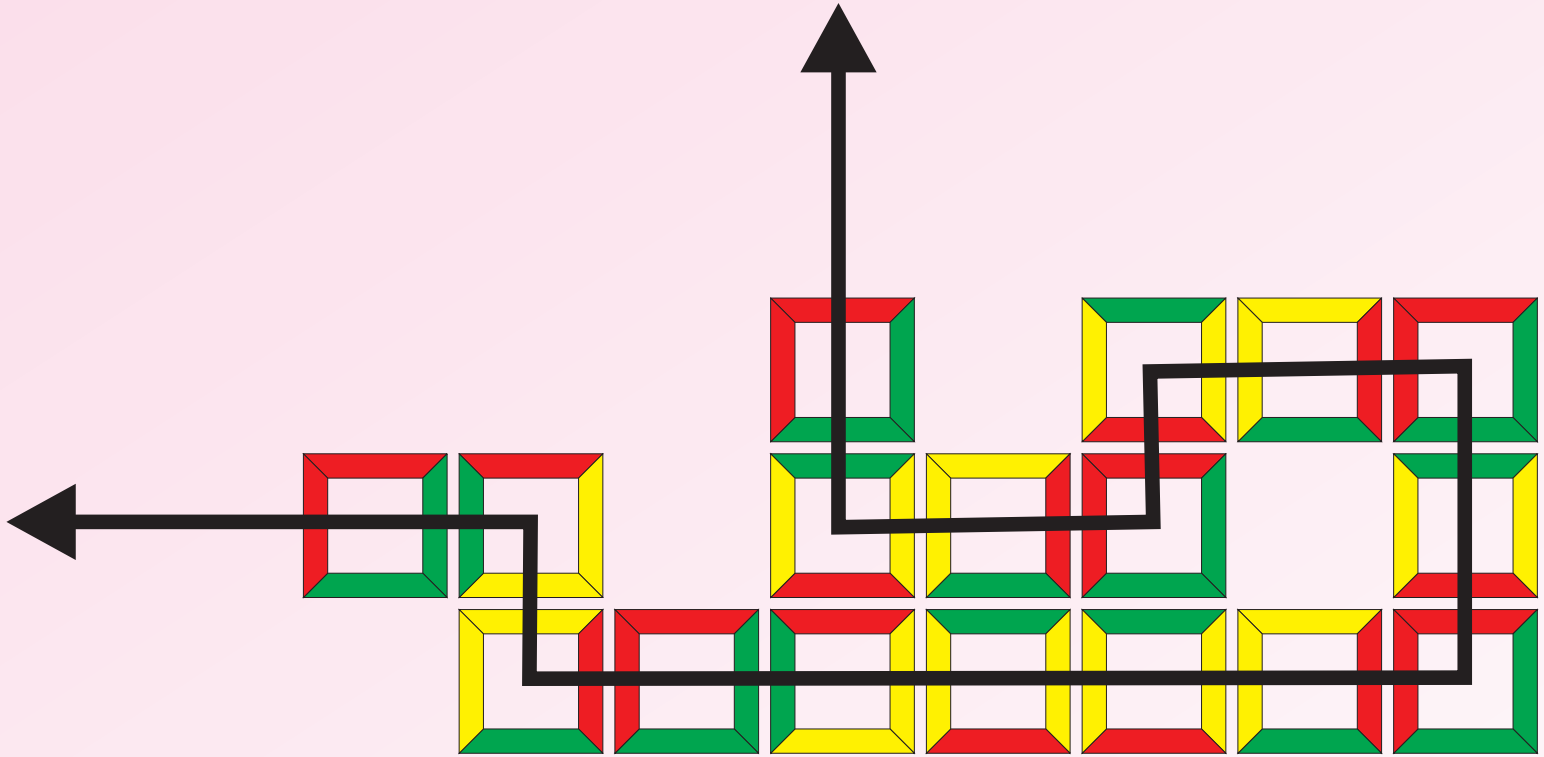


Snake: an injective function

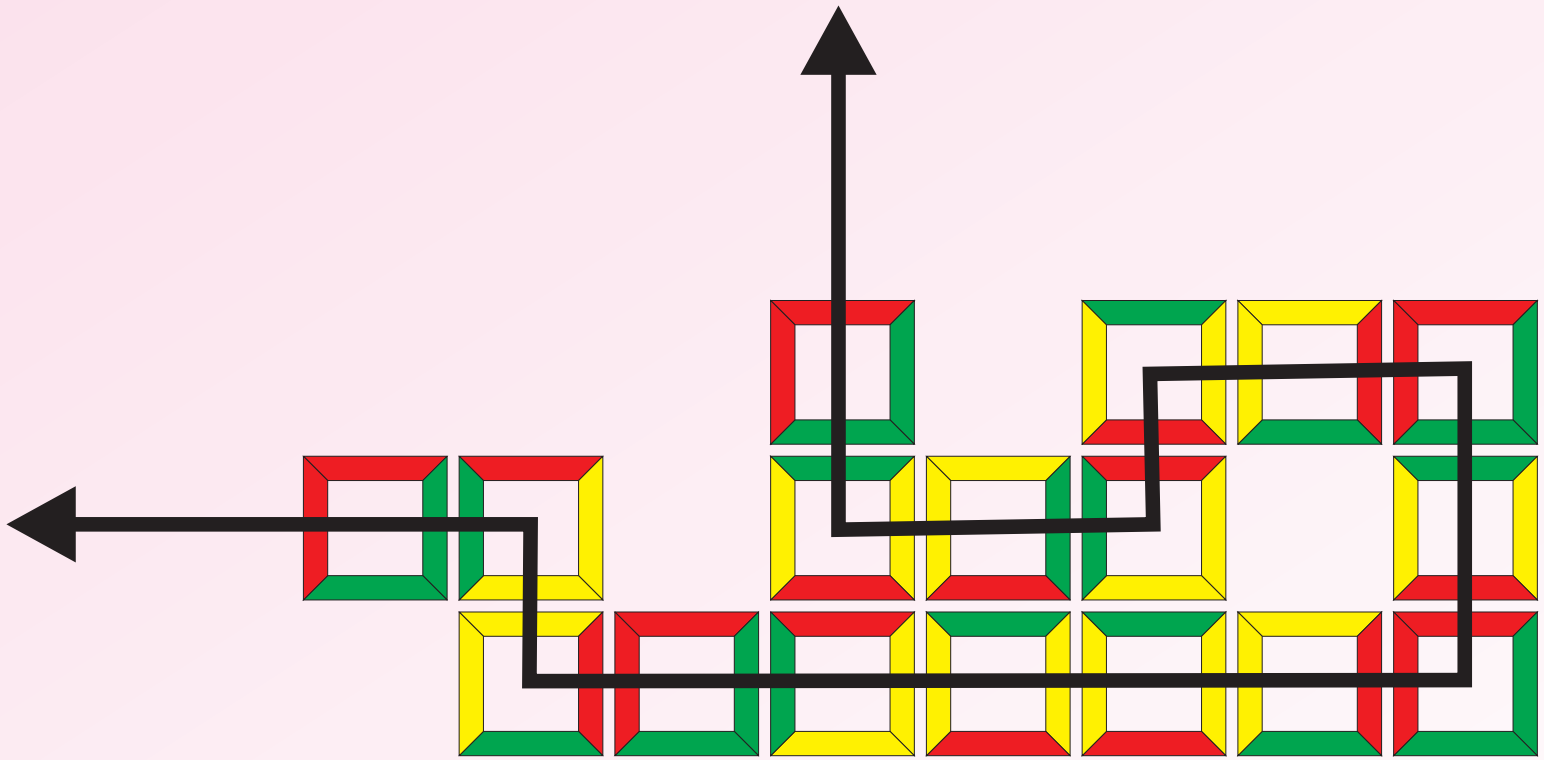
$$s : \mathbb{Z} \longrightarrow \mathbb{Z}^2$$

such that $s(i)$ and $s(i+1)$ are neighbors for all i.

- If the assembly uses strong sticking rules then the snake must be tiled so that **all neighboring tiles** of the snake must match. We call this a **strong snake**.



- If the assembly uses strong sticking rules then the snake must be tiled so that **all neighboring tiles** of the snake must match. We call this a **strong snake**.
- If the assembly uses weak sticking rules then it is enough that the **consecutive tiles** along the snake stick. This is a **weak snake**.



$\Longrightarrow$ Decision problems...

Snake tiling problems: Does a given finite set of Wang tiles admit a strong/weak snake?

These are equivalent to the termination problems in self-assembly (under the strong/weak sticking rules, respectively).

$\Longrightarrow$ Decision problems...

Snake tiling problems: Does a given finite set of Wang tiles admit a strong/weak snake?

These are equivalent to the termination problems in self-assembly (under the strong/weak sticking rules, respectively).

The Snake tiling problem was actually asked by Y. Etzion-Petruschka, D.Harel and D.Myers already in 1994, long before the self-assembly motivation of the problem.

Our answer: The questions are **undecidable**. We can reduce the domino problem into them.

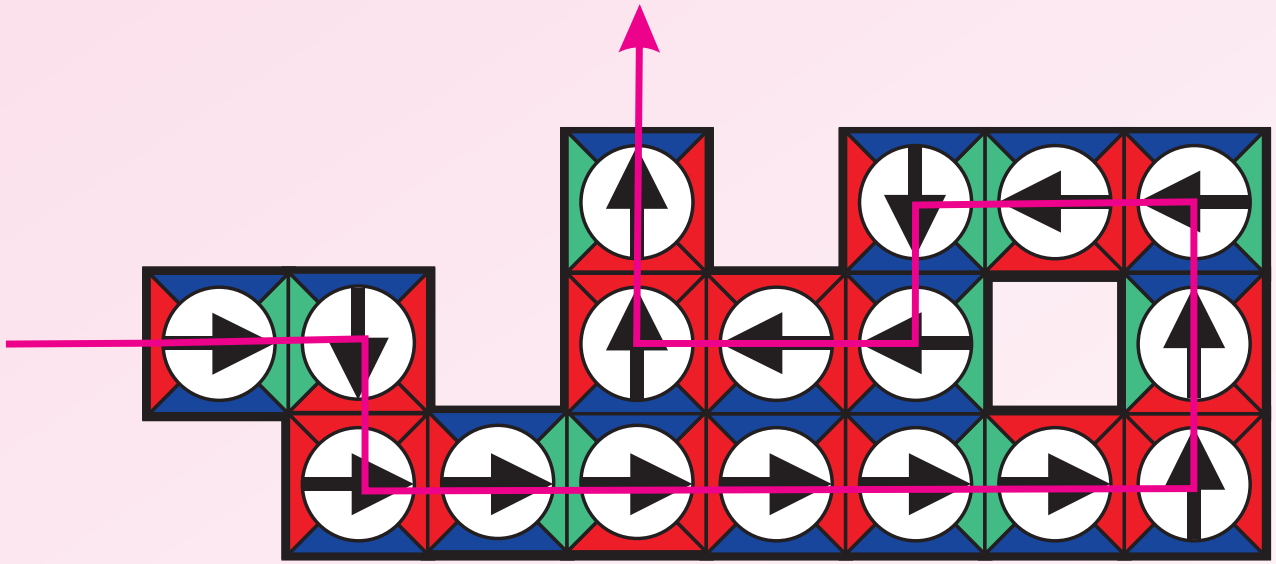
Tile set Snakes

In the reduction of the **Domino problem** into the **Snake tiling problem**, a specific tile set **Snakes** is used.

Tiles of **Snakes** have an arrow printed on them. The arrow is horizontal or vertical and it points to one of the four neighbors of the tile:

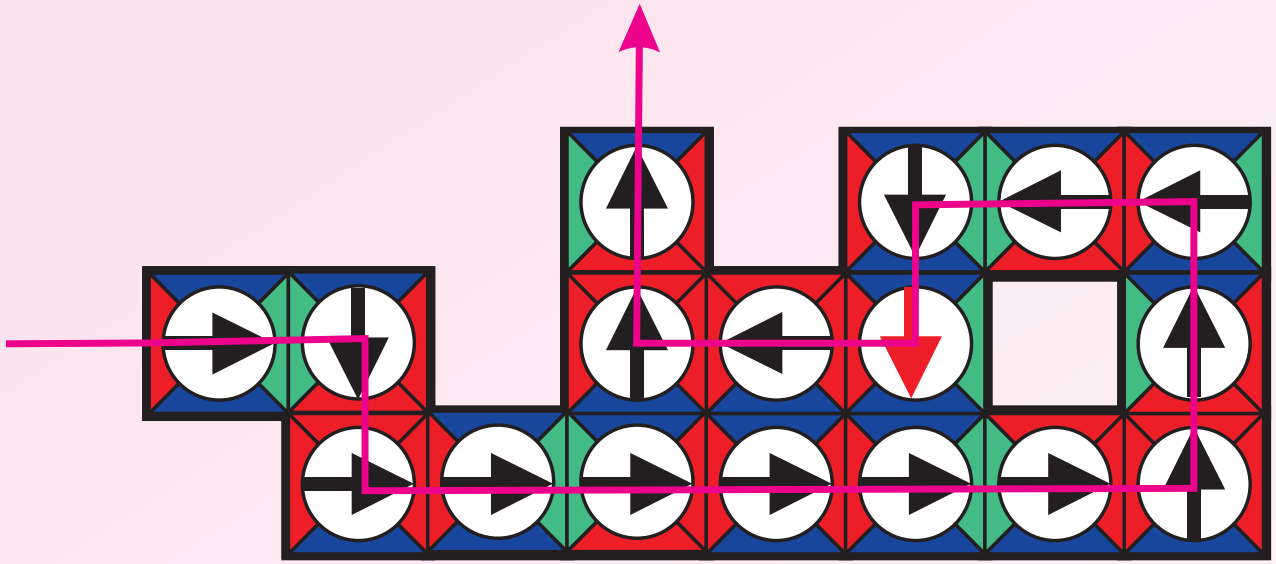
Such tiles with arrows are called **directed tiles**.

A **directed snake** is a snake, tiled with directed tiles, in which the direction of each tile points to the next tile along the snake.



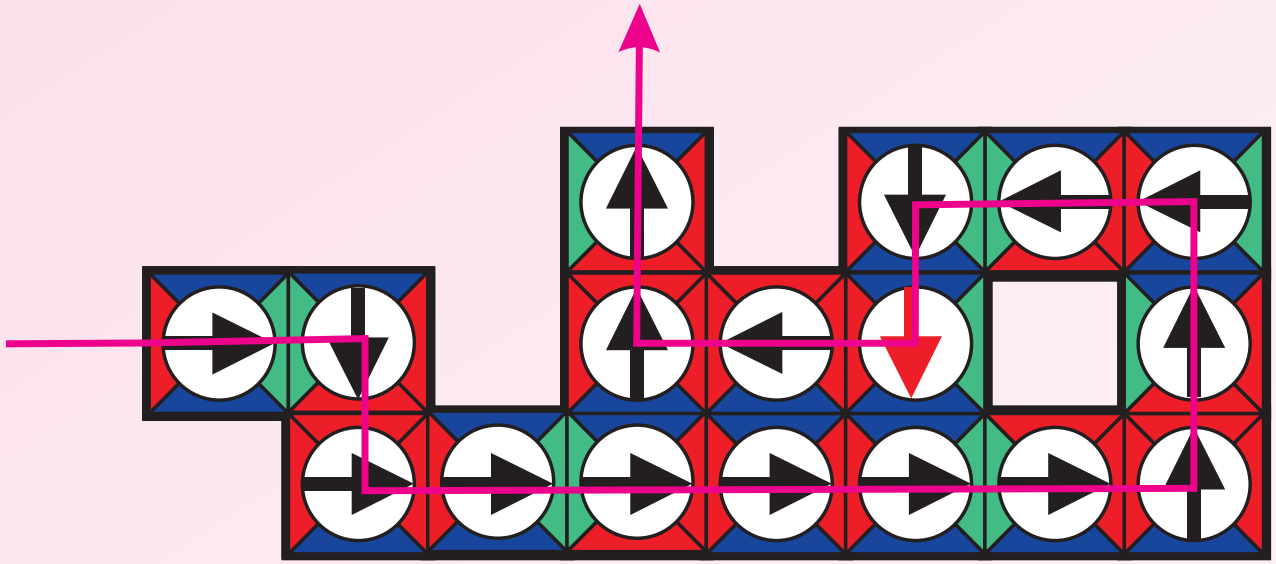
Ok

A **directed snake** is a snake, tiled with directed tiles, in which the direction of each tile points to the next tile along the snake.



Not Ok

A **directed snake** is a snake, tiled with directed tiles, in which the direction of each tile points to the next tile along the snake.



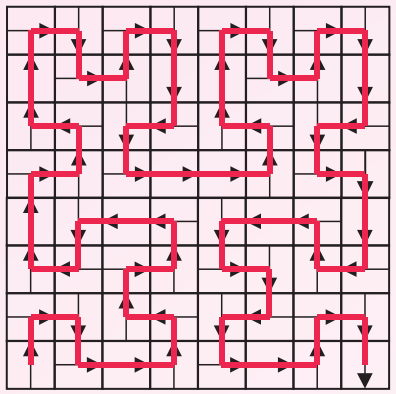
Not Ok

Two variants: Strong and weak directed snakes – but we only consider the strong variant.

The directed tile set **Snakes** has the following **plane-filling property**:

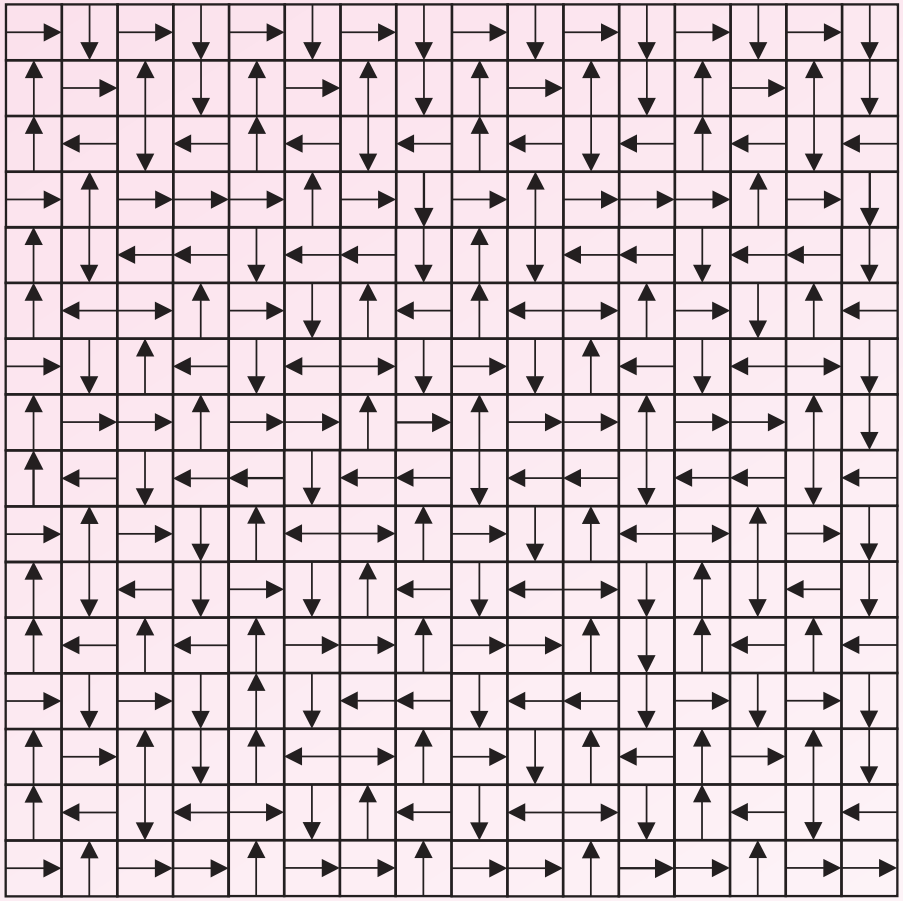
(PFP) A strong directed snake formed by Snakes is necessarily a plane-filling path.

[More precisely: for every positive integer n there exists an $n \times n$ square all of whose positions are visited by the path.]

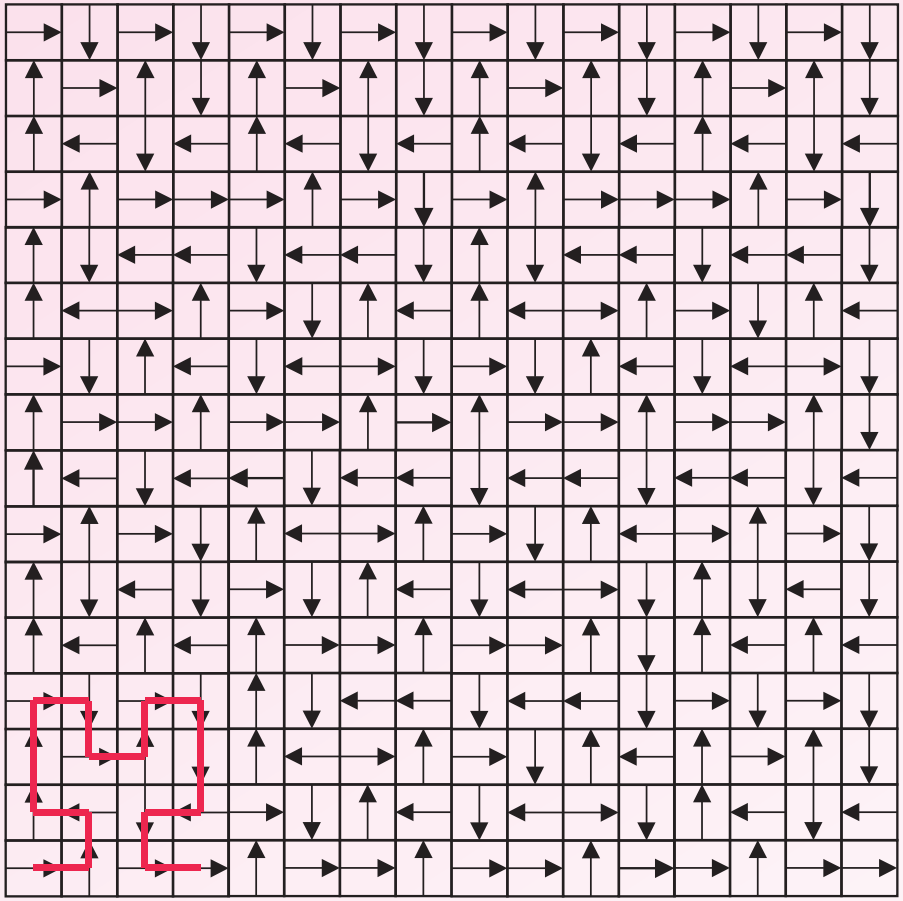


Snakes also has the property that it admits a valid tiling.

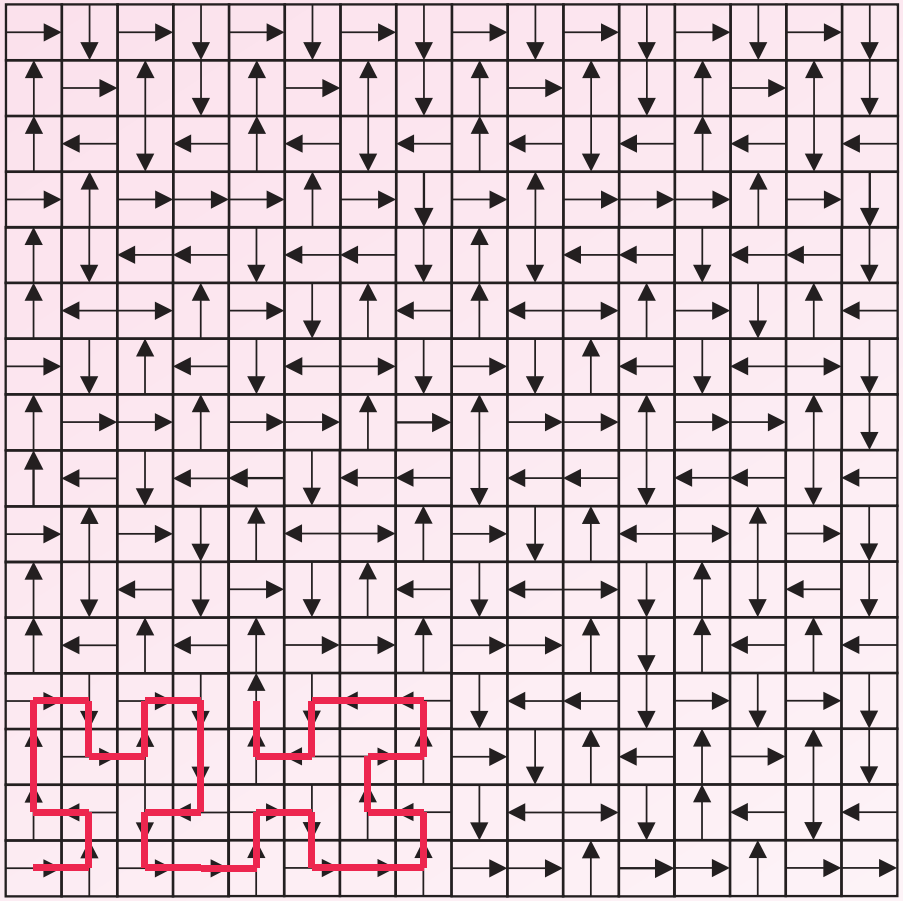
The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



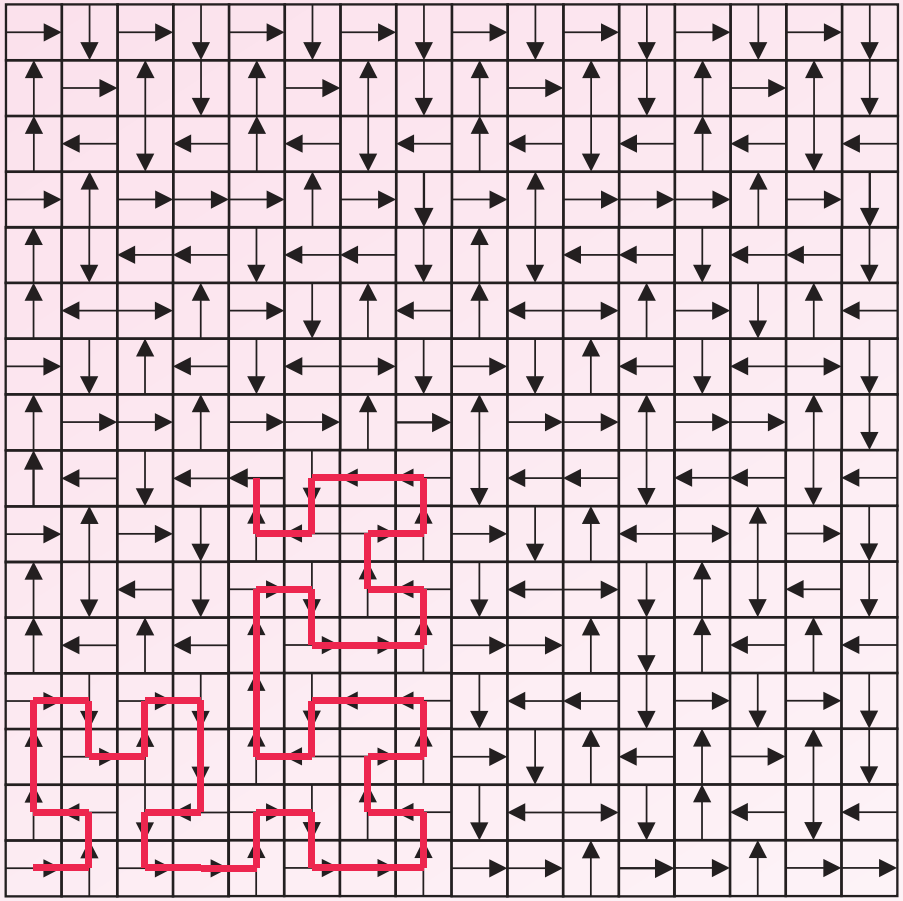
The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



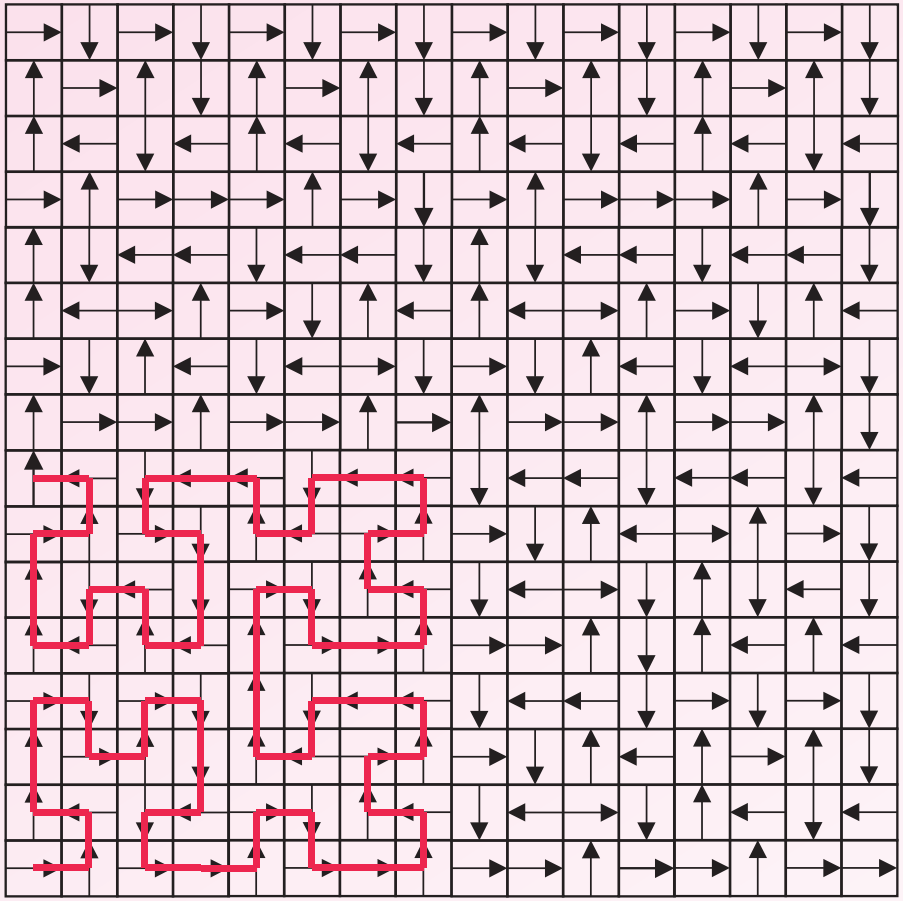
The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



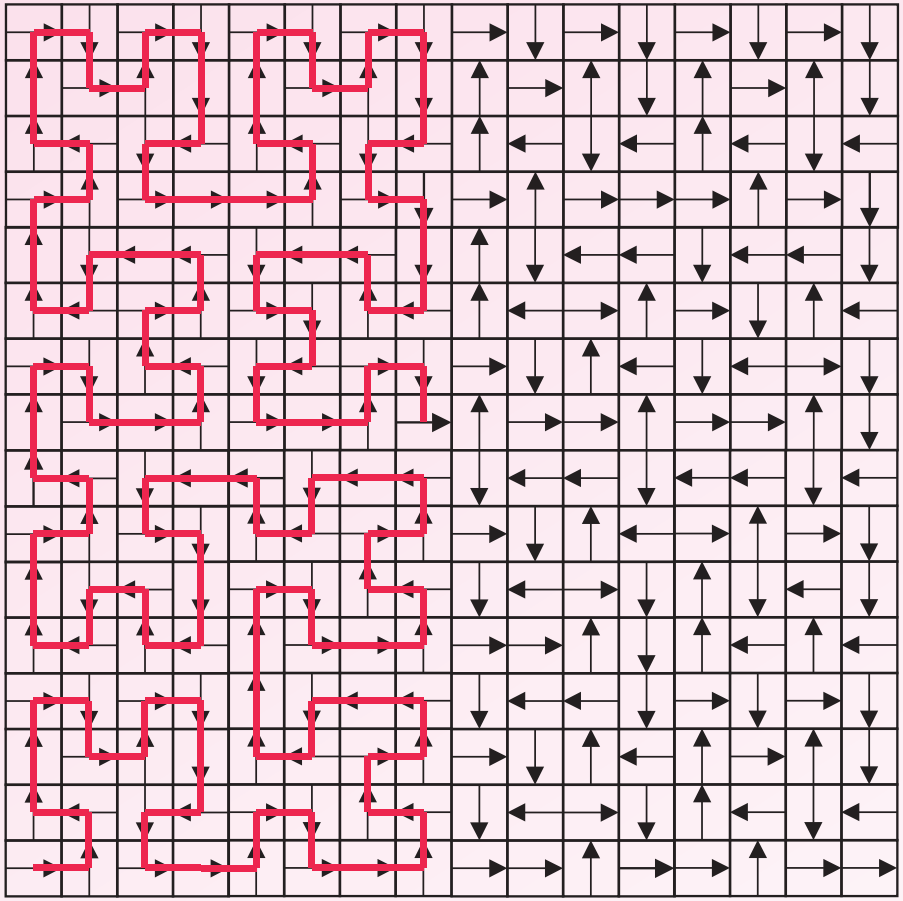
The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



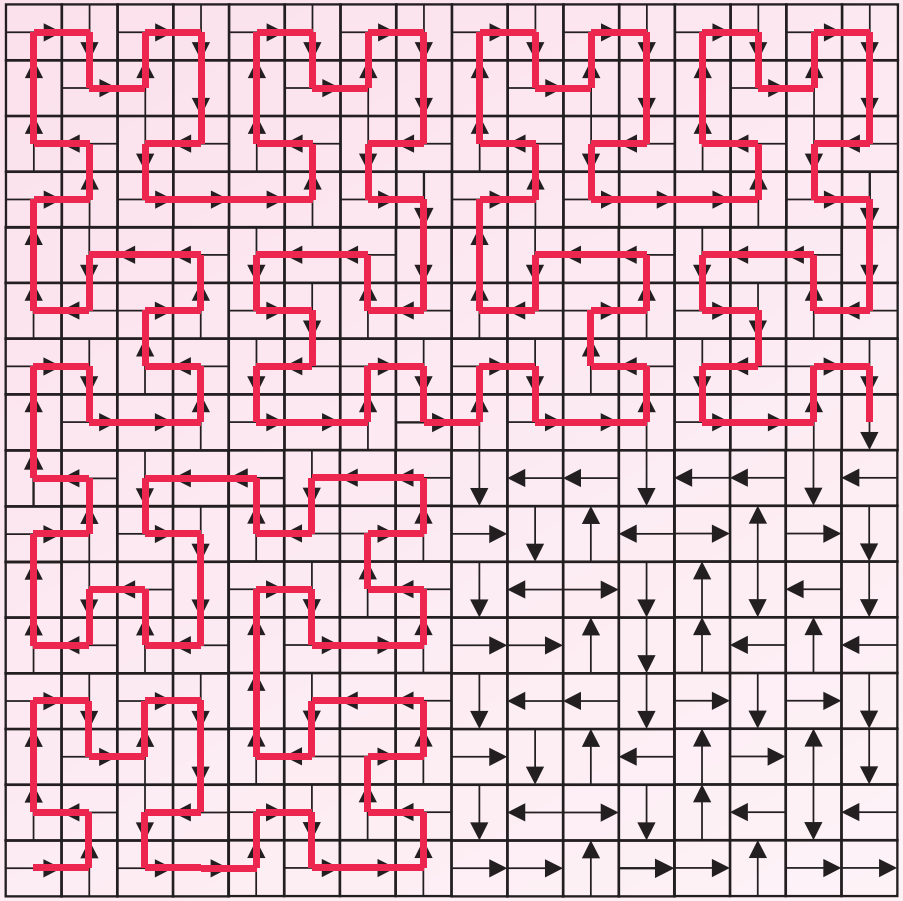
The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



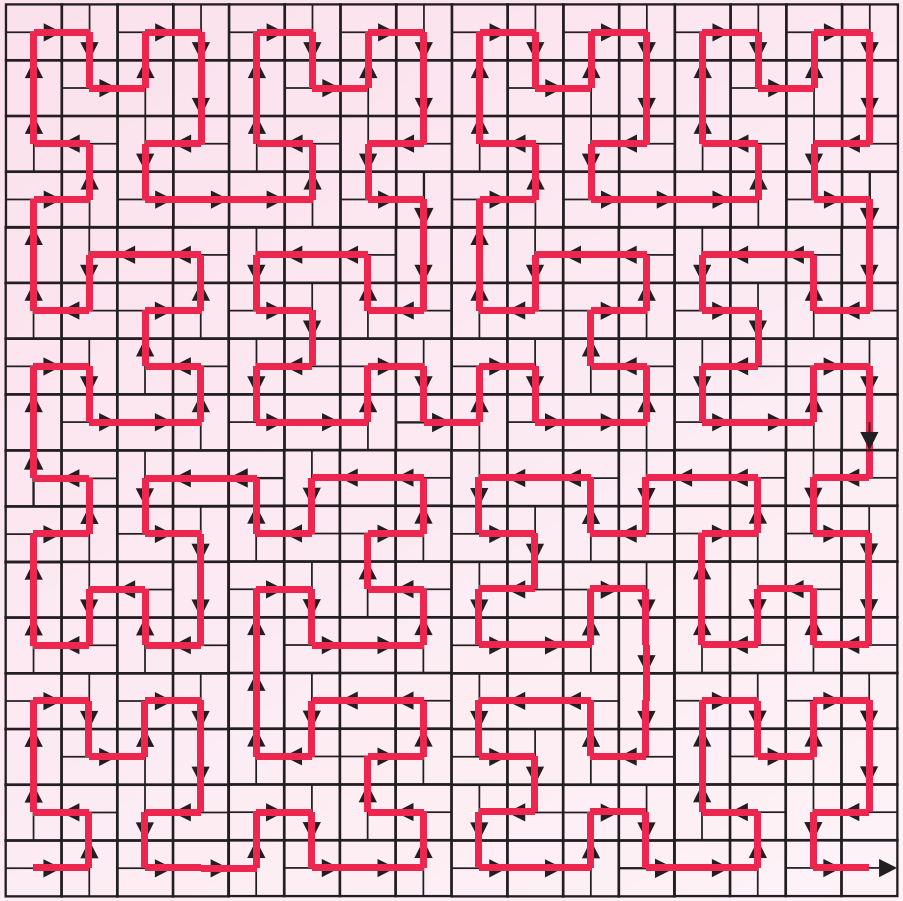
The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



The strong directed snakes by **Snakes** have the shape of the well known plane-filling Hilbert-curve



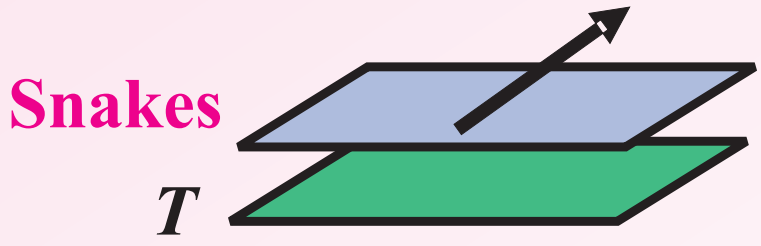
Directed snake tiling problem: Does a given set of directed tiles admit a **strong directed snake** ?

Directed snake tiling problem: Does a given set of directed tiles admit a **strong directed snake** ?

Proof of undecidability: We reduce the Domino problem, using the tile set **Snakes**.

For any given Wang tile set T, construct the directed tile set

$$D = T \times \textbf{Snakes}.$$



- Matching condition is that both layers stick.
- **Snakes** -component gives the direction.

Claim: The sandwich tiles in D admit a strong directed snake if and only if T admits a tiling of the plane

Claim: The sandwich tiles in D admit a strong directed snake if and only if T admits a tiling of the plane

($\Longleftarrow$) If T tiles the plane then D tiles the plane. Any path that follows the arrows is a strong directed snake.

Claim: The sandwich tiles in D admit a strong directed snake if and only if T admits a tiling of the plane

($\Longrightarrow$) Suppose D admits a strong directed snake s.

- Plane-filling property of **Snakes** $\Longrightarrow$ s covers arbitrarily large squares.
- No tiling errors in the T component along s $\Longrightarrow$ T can tile arbitrarily large squares $\Longrightarrow$ T admits a tiling of the plane.

Claim: The sandwich tiles in D admit a strong directed snake if and only if T admits a tiling of the plane

$\Longrightarrow$ **Directed snake tiling problem is undecidable.** $\square$

(Undirected) snake tiling problems

Next we reduce the **Directed snake tiling problem** to the (undirected, strong and weak) **Snake tiling problems**.

(Undirected) snake tiling problems

Next we reduce the **Directed snake tiling problem** to the (undirected, strong and weak) **Snake tiling problems**.

Let D be any given set of directed tiles, called **macro-tiles**.

We effectively construct a set T of Wang tiles, called **mini-tiles**, such that

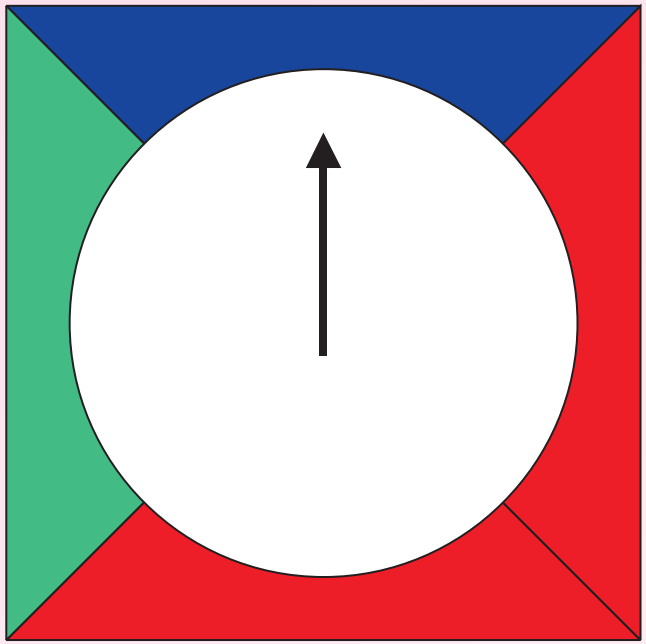
- If D admits a directed strong snake then T admits (undirected) strong and weak snakes, and
- If D does not admit a directed strong snake then T does not admit (undirected) strong or weak snakes.

(Undirected) snake tiling problems

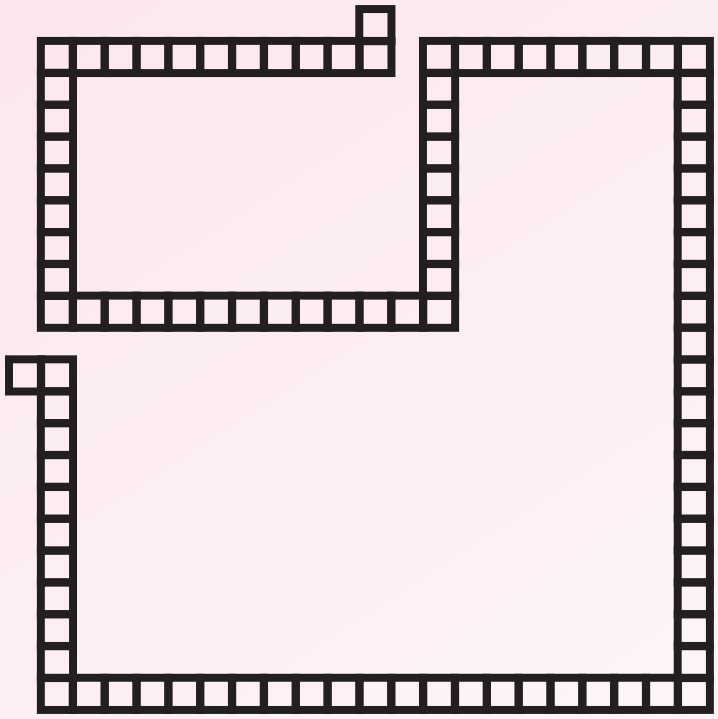
A **motif construction** is used to convert

- Directed $\longrightarrow$ Undirected
- Strong sticking $\longrightarrow$ weak sticking

For each macro-tile $d \in D$ we build a sequence of undirected mini-tiles. The mini-tiles are colored with unique colors that force them to form a motif: a finite snake that goes around the the four edges of the macro-tile d:

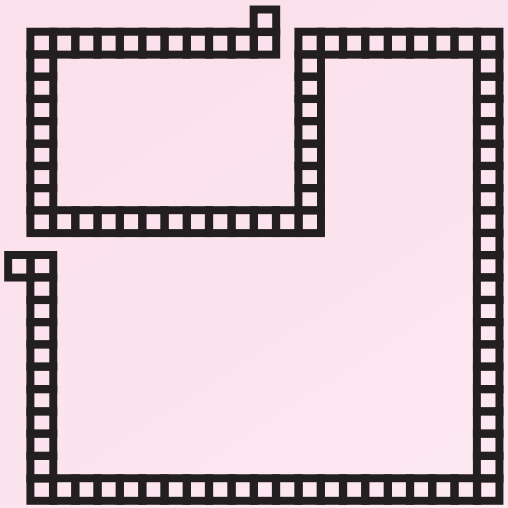


macro-tile d

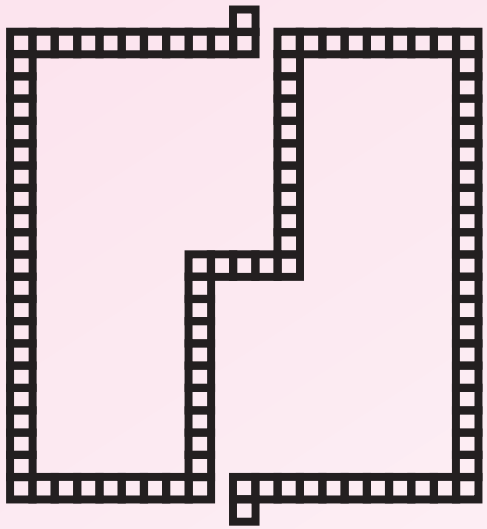


Corresponding motif of mini-tiles

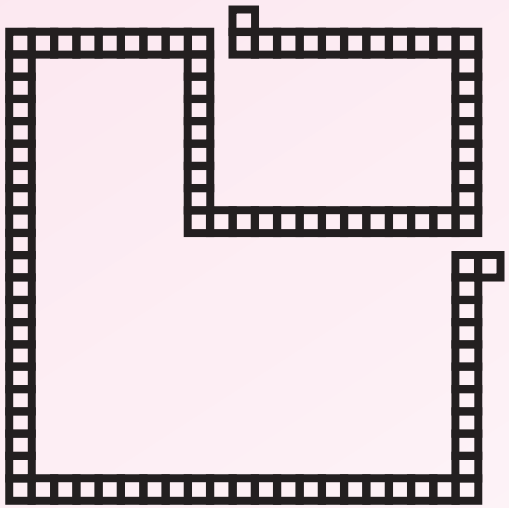
Since the entry direction into a tile is not specified by the arrow, we need three motifs for each $d \in D$, one for every possible entry direction:



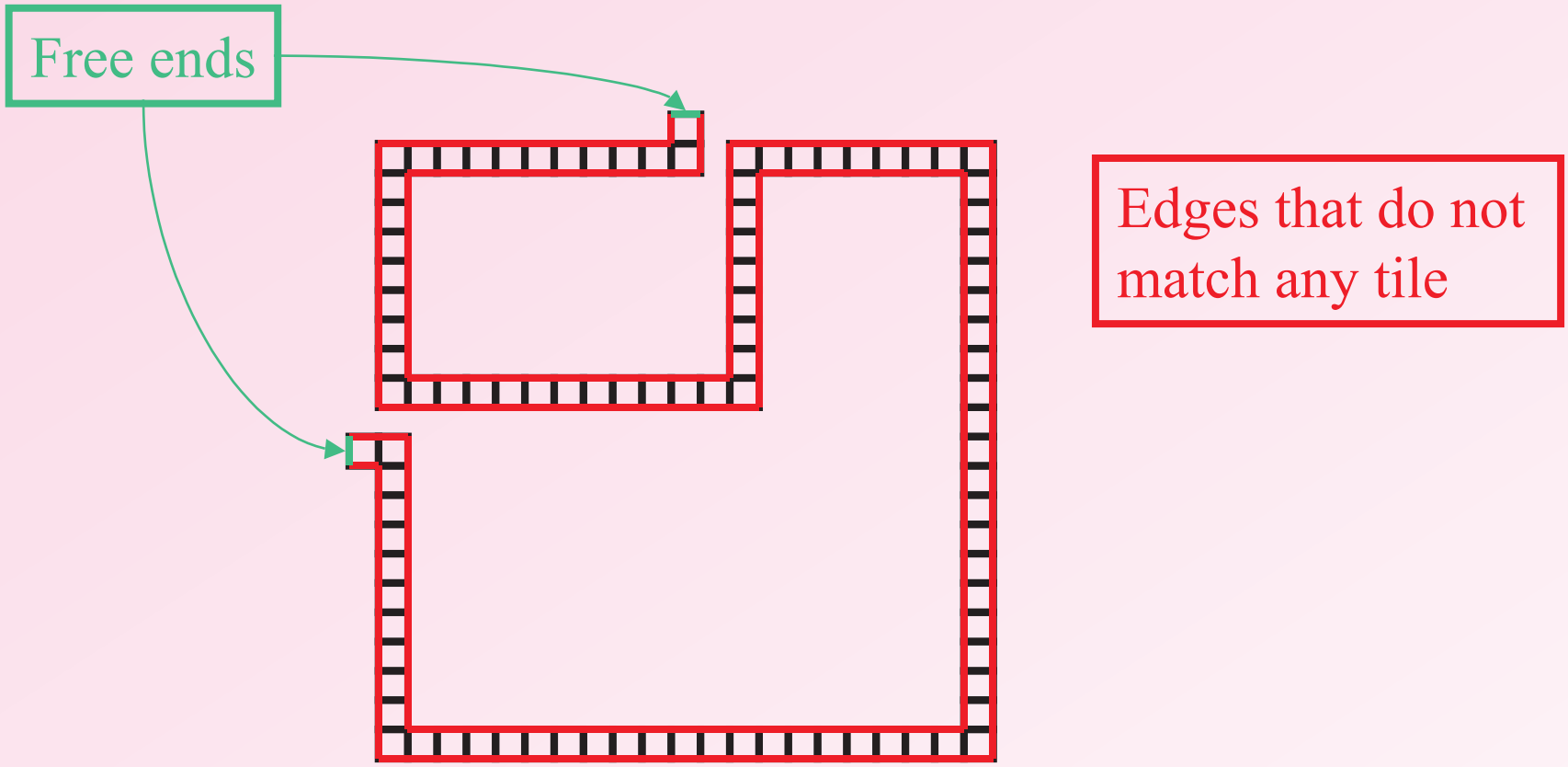
Left entry



Bottom entry

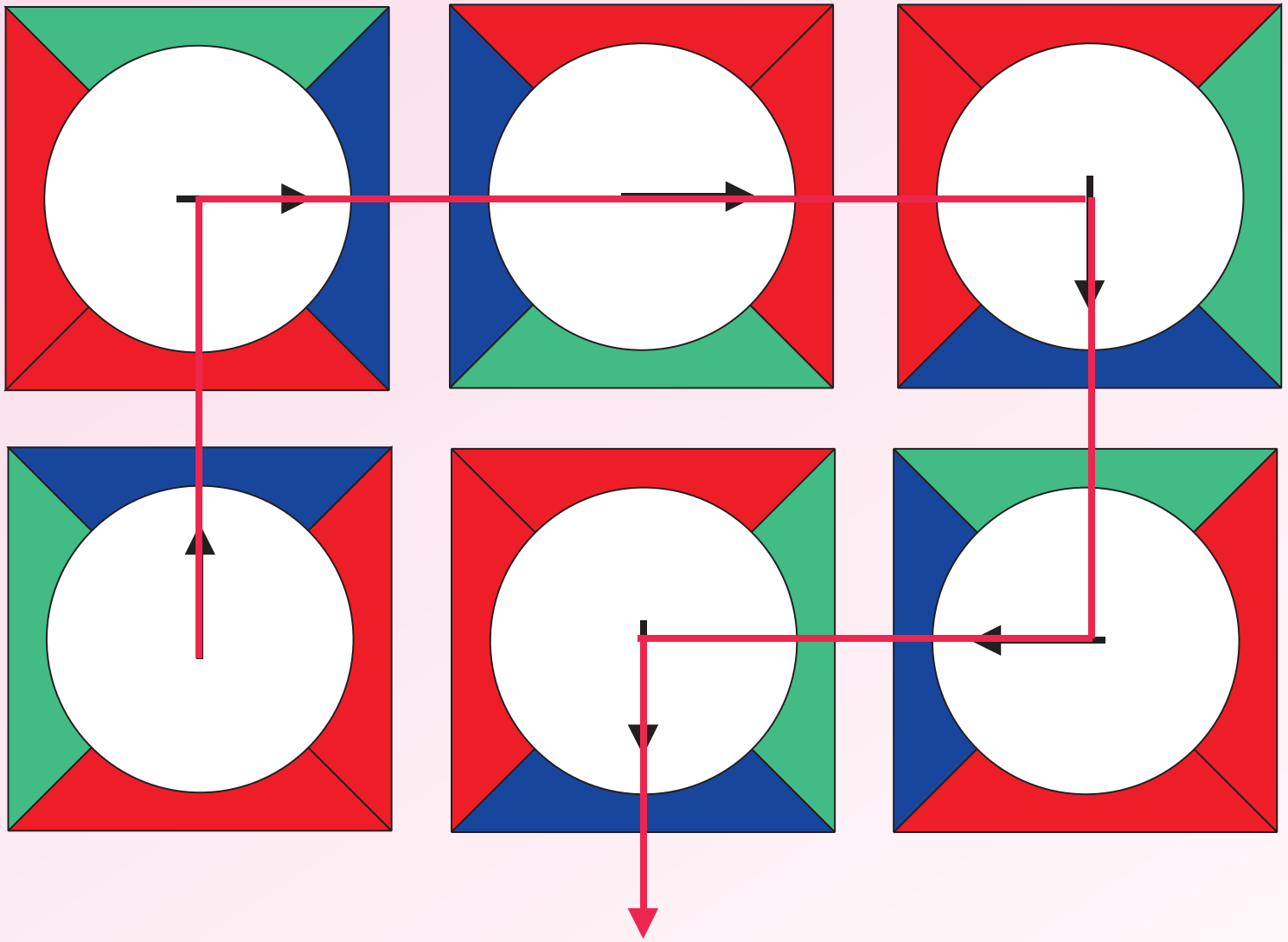


Right entry

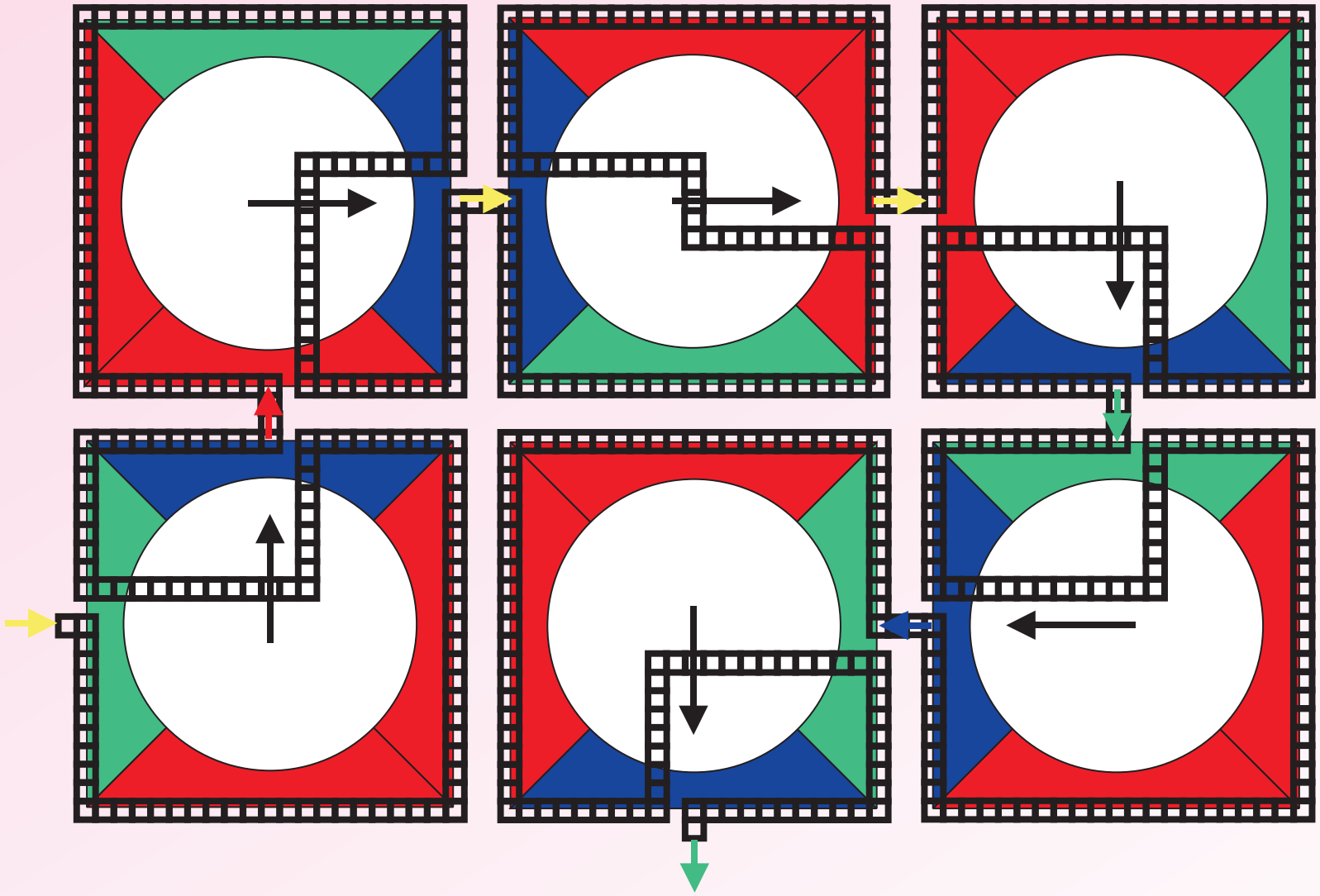


Each motif matches other motifs only at the two ”free” ends.

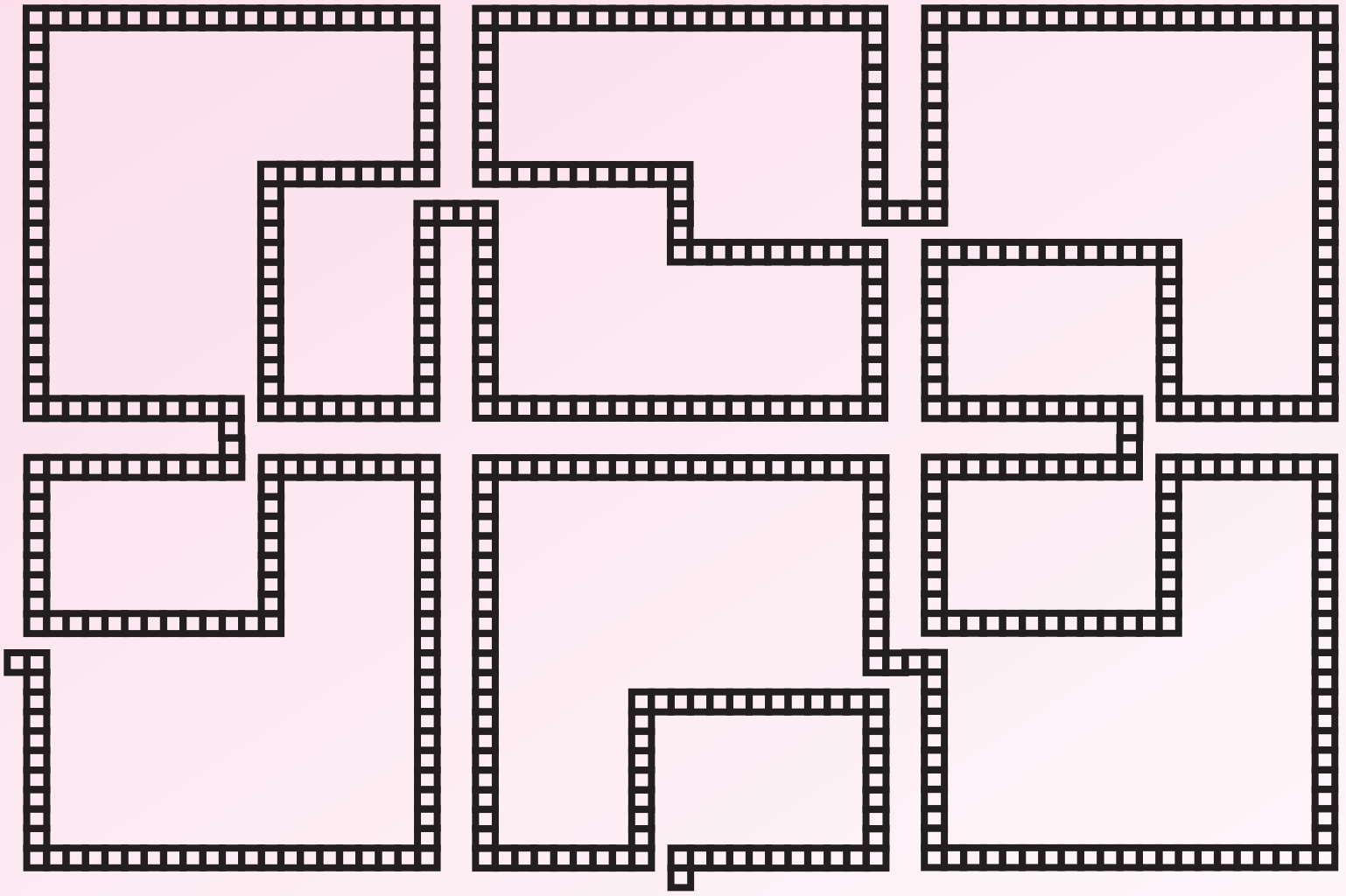
Gluing motifs from their free ends forms snakes that exactly correspond to paths of macro-tiles specified by arrows:



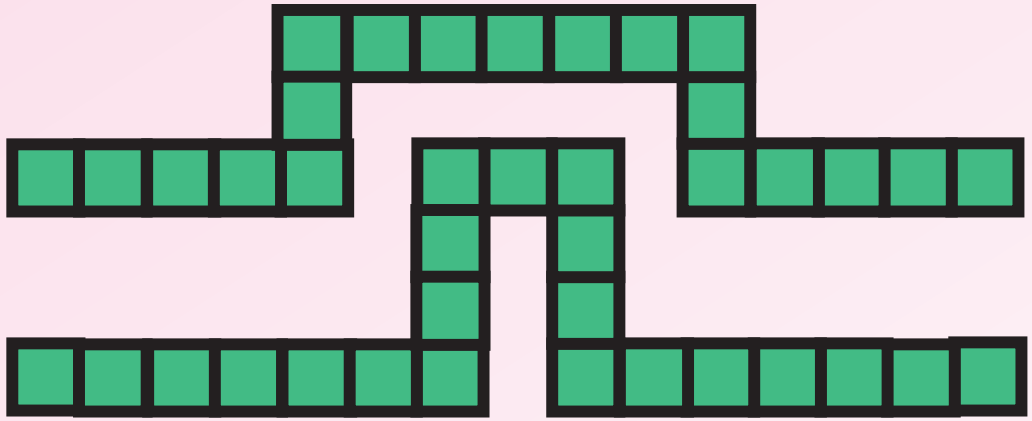
Gluing motifs from their free ends forms snakes that exactly correspond to paths of macro-tiles specified by arrows:



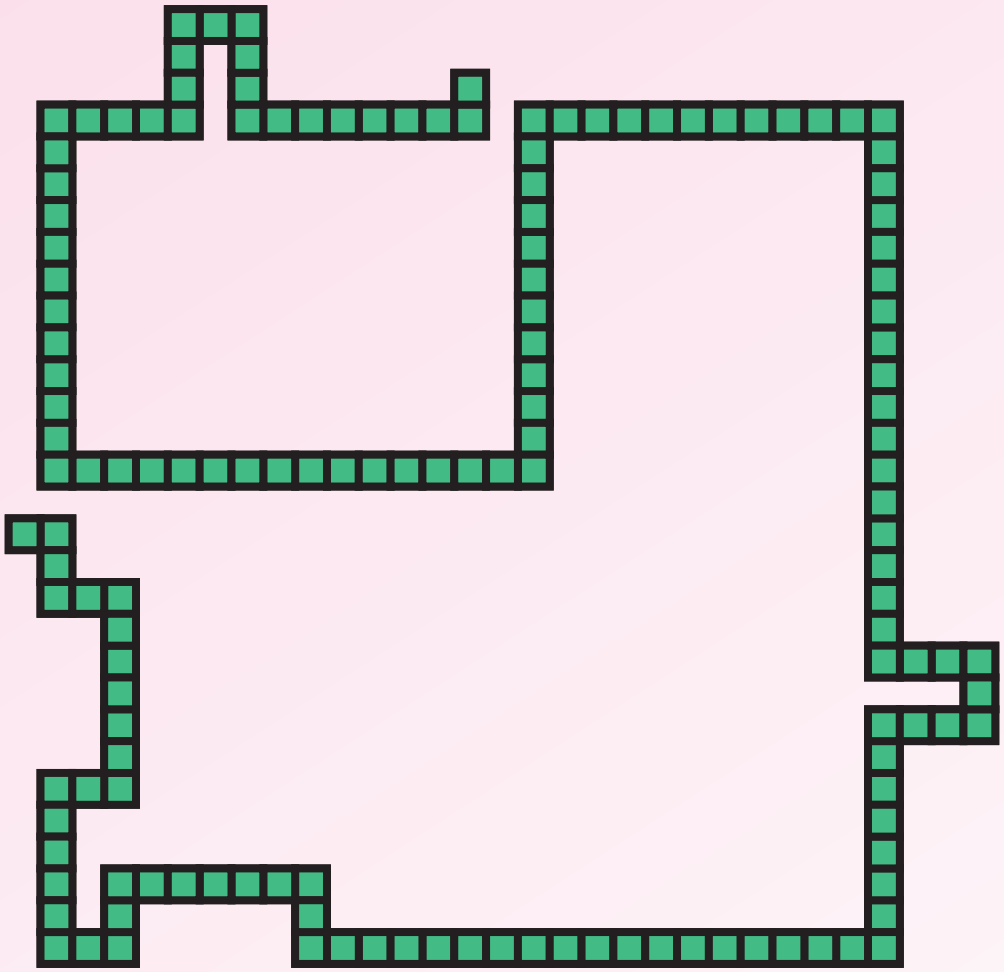
Gluing motifs from their free ends forms snakes that exactly correspond to paths of macro-tiles specified by arrows:



So far the motifs have no constraints corresponding to edge coloring of the macro-tiles. To simulate the colors, the motifs are bent to form one **bump** or **dent** on each side of the motif.



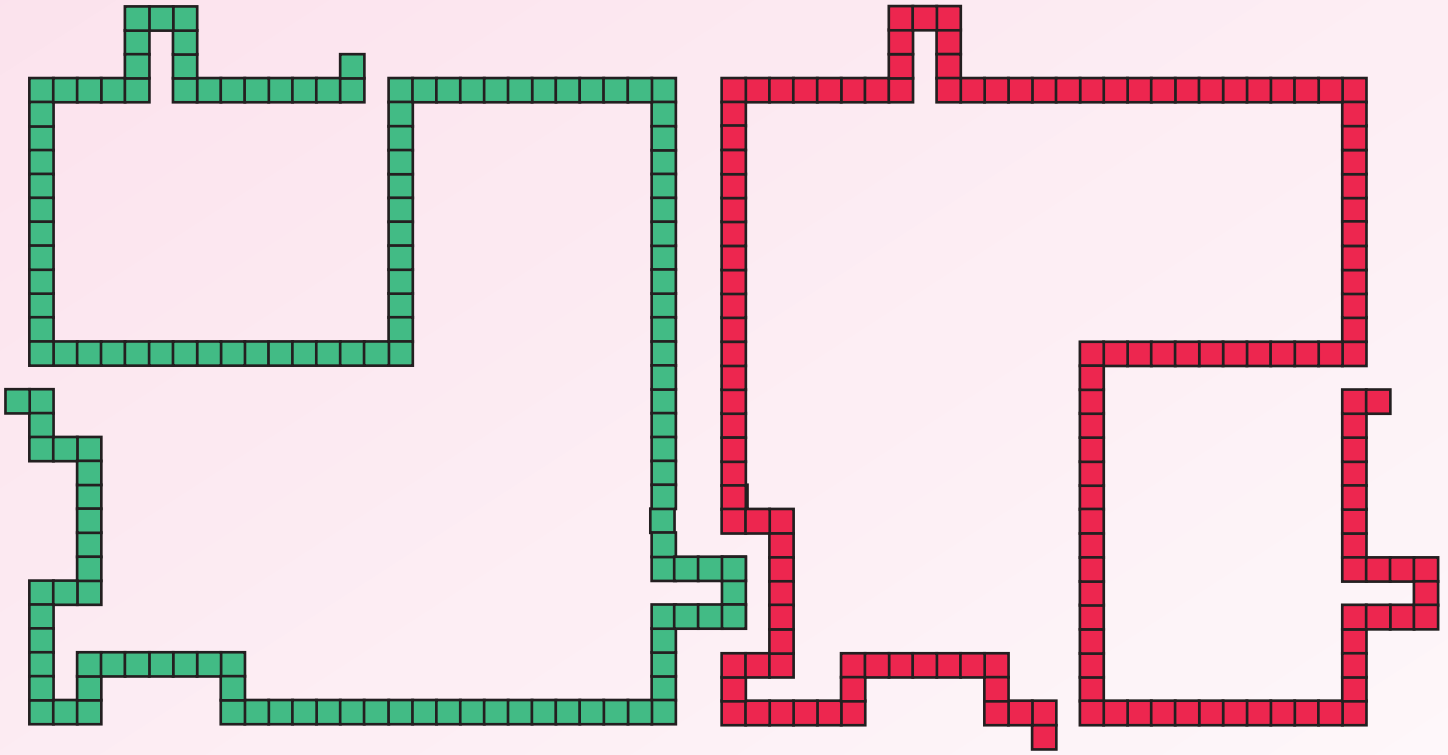
The N and E side of each motif contains one bump, and the S and W side contains one dent:



The exact position of the bump or the dent encodes the color of the edge in the corresponding macro-tile. Each color corresponds to a unique position.

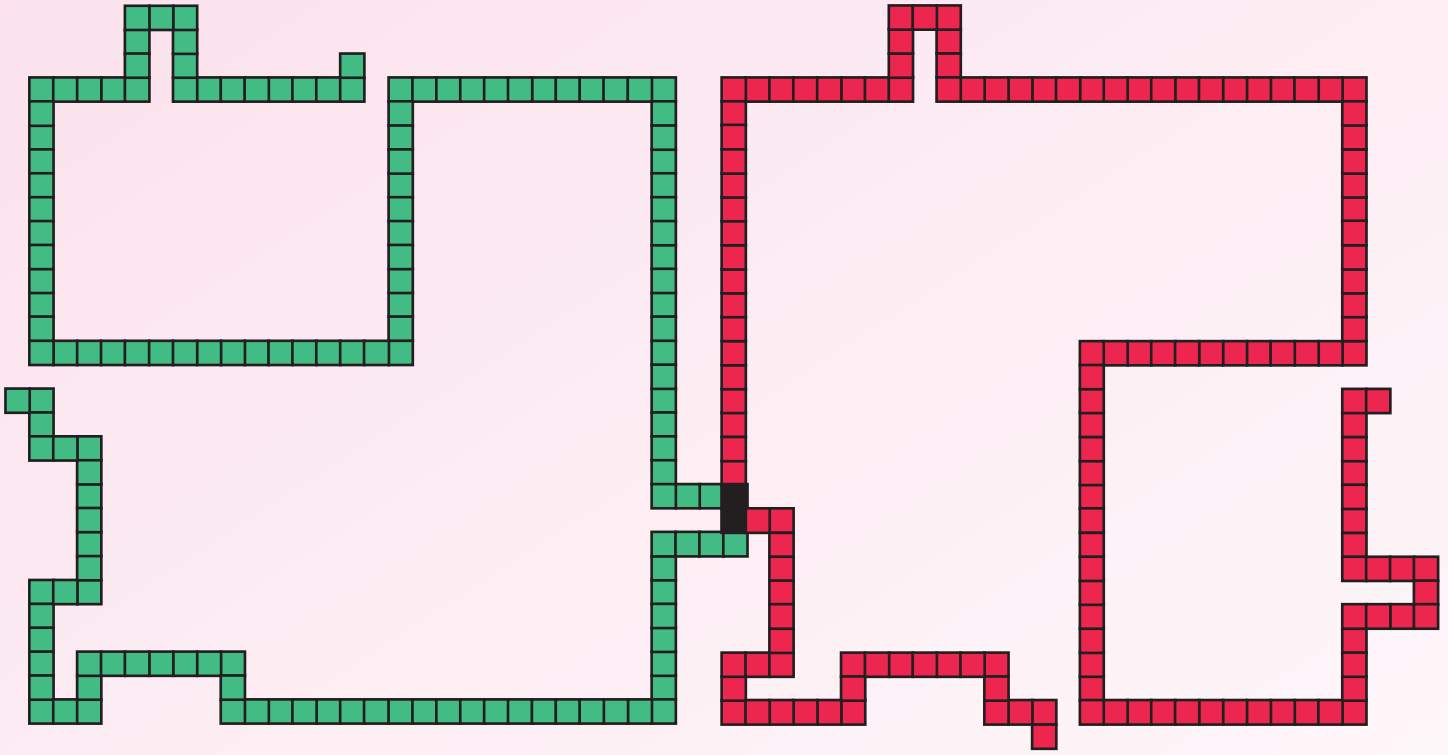
The exact position of the bump or the dent encodes the color of the edge in the corresponding macro-tile. Each color corresponds to a unique position.

If the colors of two adjacent macro-tiles match then the bump exactly fits inside the dent:



The exact position of the bump or the dent encodes the color of the edge in the corresponding macro-tile. Each color corresponds to a unique position.

But if the colors do not match then the two motifs would overlap, which is not possible:



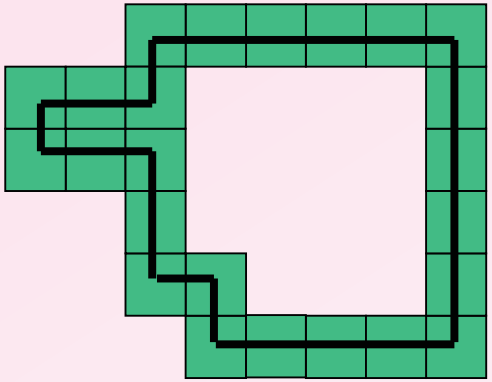
Claim: Macro-tiles in D admit a strong directed snake if and only if the mini-tiles in T admit a (strong/weak) infinite snake.

$\Longrightarrow$ **(Strong and weak) snake tiling problems are undecidable, as are the termination problems of self-assembly.**

□

Loops

With a similar argument we can prove the undecidability of the **Loop tiling problems**.



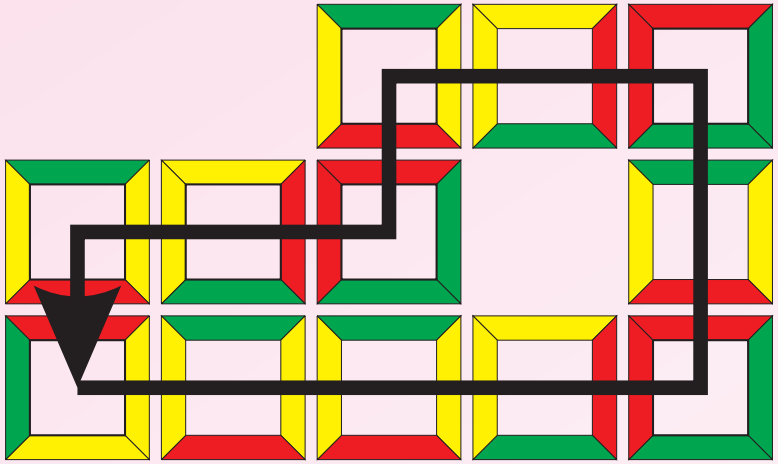
Loop: a finite snake whose ends meet. A loop of length n is a one-to-one function

$$s : \{1, 2, \ldots, n\} \longrightarrow \mathbb{Z}^2$$

such that $s(i)$ and $s(i + 1 \bmod n)$ are neighbors for all $i = 1, 2, \ldots, n$.

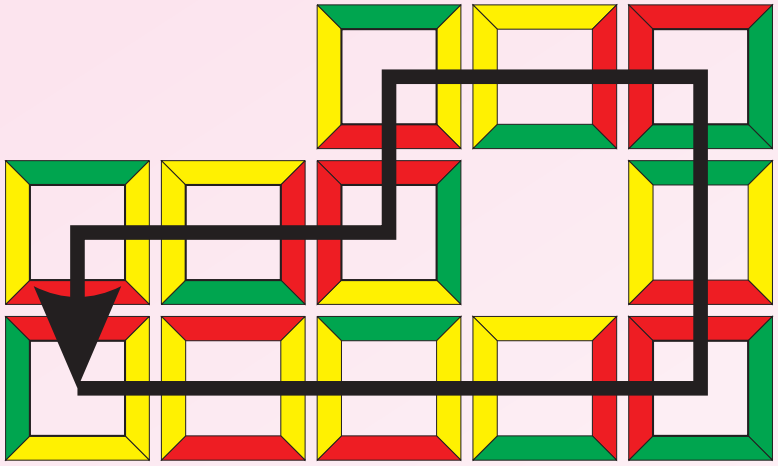
We tile loops with Wang tiles:

- **Strong loop**: all neighboring tiles on the loop stick.



We tile loops with Wang tiles:

- **Strong loop**: all neighboring tiles on the loop stick.
- **Weak loop**: only consecutive tiles are required to stick.

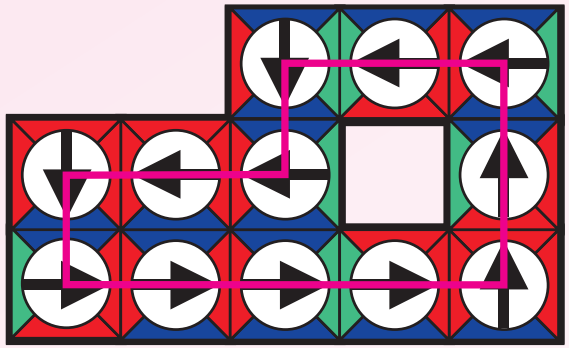


We tile loops with Wang tiles:

- **Strong loop**: all neighboring tiles on the loop stick.
- **Weak loop**: only consecutive tiles are required to stick.

Or we tile them with directed tiles:

- **Strong directed loop**: all neighboring tiles on the loop stick, and the loop follows the arrows on the tiles.

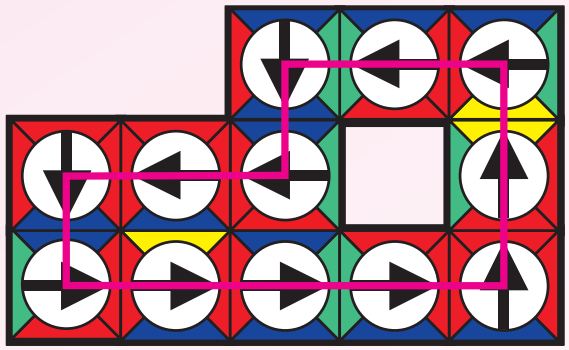


We tile loops with Wang tiles:

- **Strong loop**: all neighboring tiles on the loop stick.
- **Weak loop**: only consecutive tiles are required to stick.

Or we tile them with directed tiles:

- **Strong directed loop**: all neighboring tiles on the loop stick, and the loop follows the arrows on the tiles.
- **Weak directed loop**...



$\Longrightarrow$ Decision problems...

Loop tiling problems: Does a given set of (directed) Wang tiles admit a strong/weak (directed) loop ?

All variants are undecidable.

Remark:

- The **negative** instances of the **snake tiling problems**
- The **positive** instances of the **loop tiling problems**

are semi-decidable.

Tile set Loops

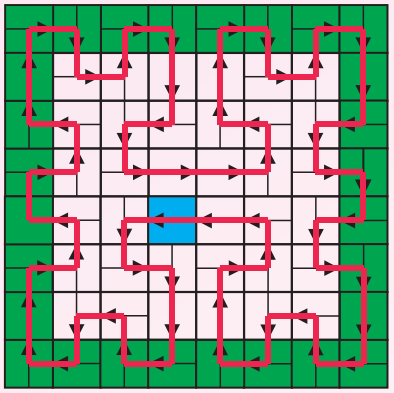
To prove the **Loop tiling problems** undecidable, we reduce the **Finite domino problem** using a specific set **Loops** of directed tiles.

The tiles in **Loops** are partitioned into three parts:

(A) Normal tiles (B) Blue tiles (C) Green tiles

Loops has a **rectangle-covering property**:

(RCP) Any strong directed loop formed by Loops completely covers some rectangle whose boundary tiles are green and there is a single blue tile in the interior.

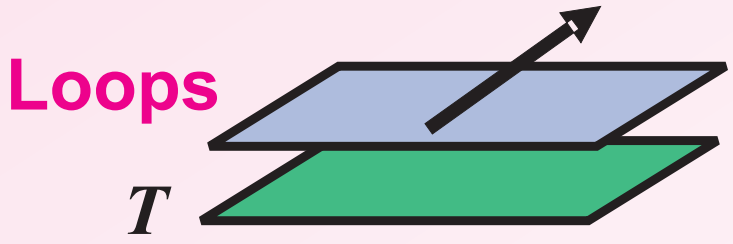


Loops also has the property that it admits valid tilings containing arbitrarily long loops.

We reduce the **finite domino problem** to the **directed loop tiling problem**, using the tile set **Loops**:

For any given Wang tile set T, construct the directed tile set

$$D \subseteq T \times \textbf{Loops}.$$



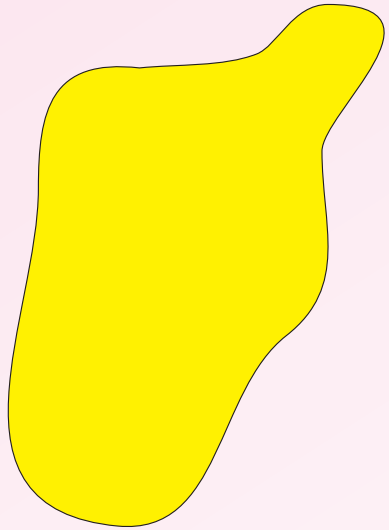
with the constraints that in $(t, l) \in D$

- if l is green then t is blank, and
- if l is blue then t is not blank.

Claim: The sandwich tiles in D admit a strong directed loop if and only if T admits a tiling of the plane such that all but a finite, non-zero number of tiles are blank.

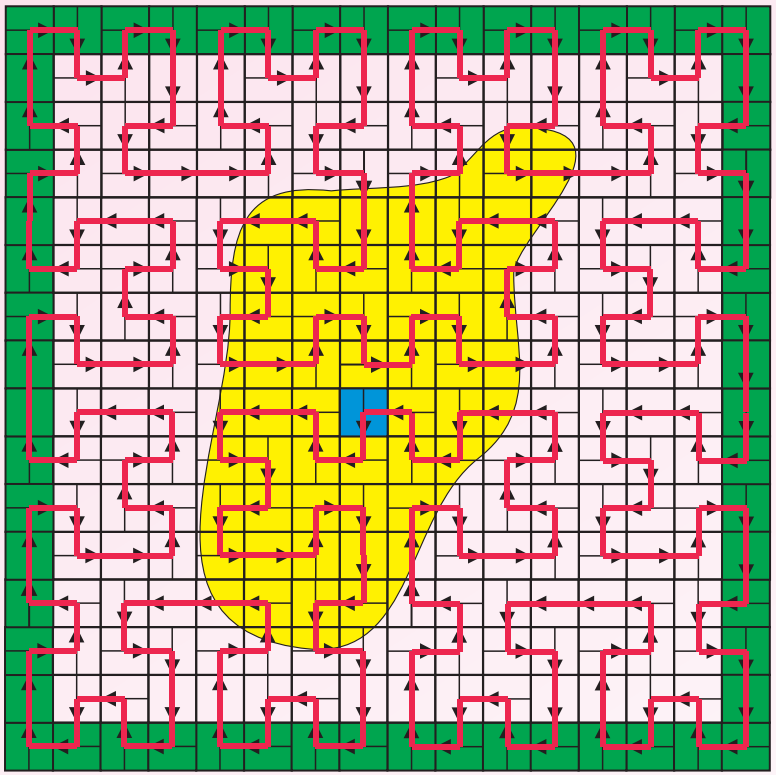
Claim: The sandwich tiles in D admit a strong directed loop if and only if T admits a tiling of the plane such that all but a finite, non-zero number of tiles are blank.

($\Longleftarrow$)



Claim: The sandwich tiles in D admit a strong directed loop if and only if T admits a tiling of the plane such that all but a finite, non-zero number of tiles are blank.

($\Longleftarrow$)

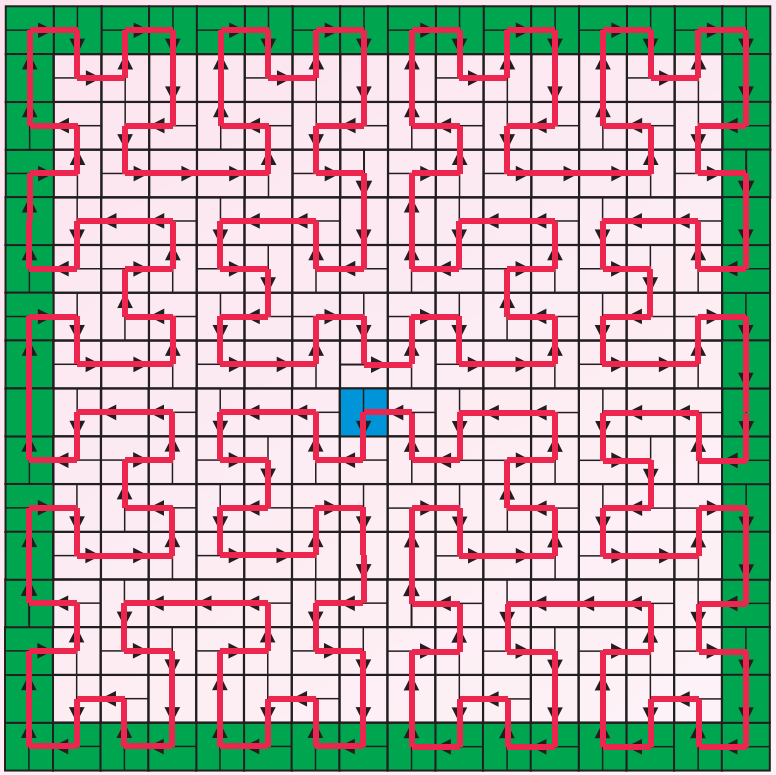


Claim: The sandwich tiles in D admit a strong directed loop if and only if T admits a tiling of the plane such that all but a finite, non-zero number of tiles are blank.

($\Longrightarrow$)

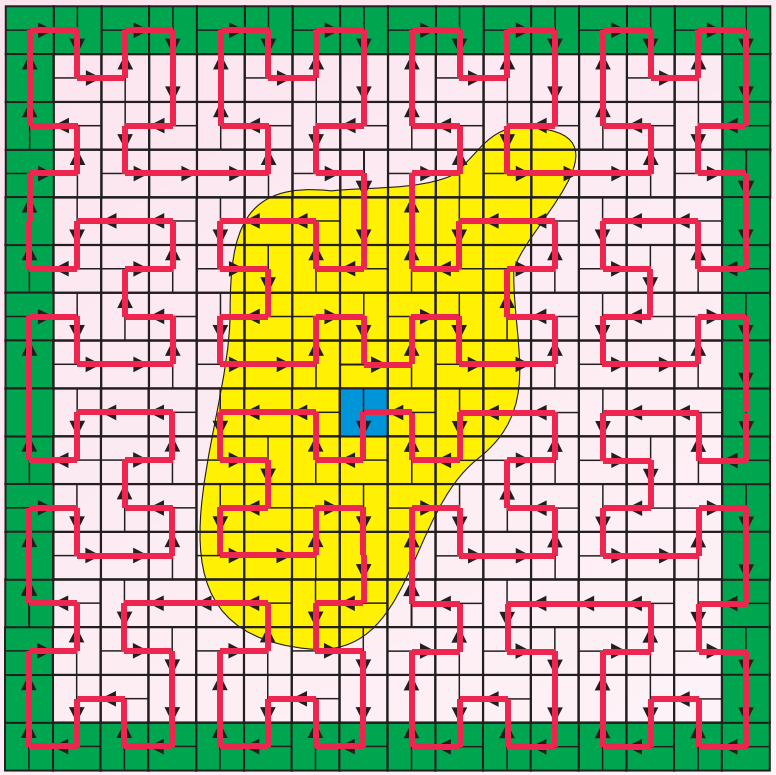
Claim: The sandwich tiles in D admit a strong directed loop if and only if T admits a tiling of the plane such that all but a finite, non-zero number of tiles are blank.

($\Longrightarrow$)



Claim: The sandwich tiles in D admit a strong directed loop if and only if T admits a tiling of the plane such that all but a finite, non-zero number of tiles are blank.

($\Longrightarrow$)



Claim: The sandwich tiles in D admit a strong directed loop if and only if T admits a tiling of the plane such that all but a finite, non-zero number of tiles are blank.

Any algorithm to solve the **directed loop tiling problem** yields an algorithm to solve the **finite domino problem**

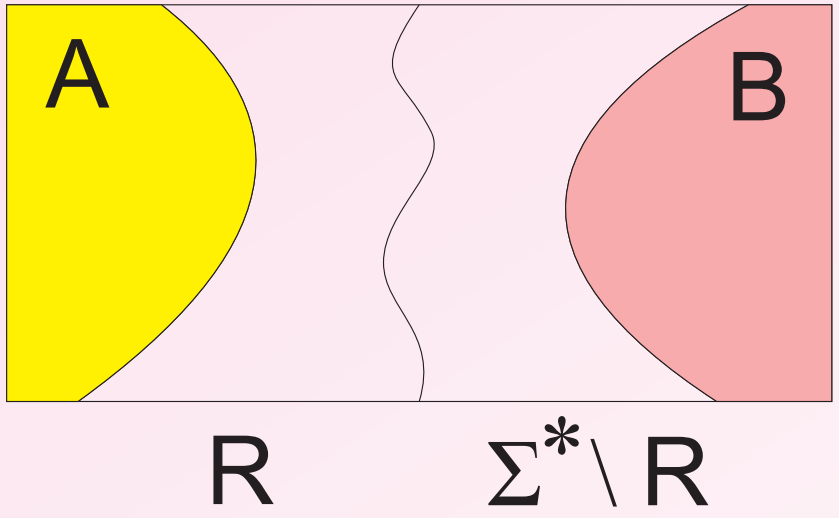
To change

- directed $\longrightarrow$ undirected,
- strong $\longrightarrow$ weak,

we use the same **motif construction**, obtaining the undecidability of all variants of the loop tiling problem.

Recursive inseparability

Disjoint sets $A, B \subseteq \Sigma^*$ are **recursively inseparable** if there is no recursive R such that $A \subseteq R$ and $B \cap R = \emptyset$.

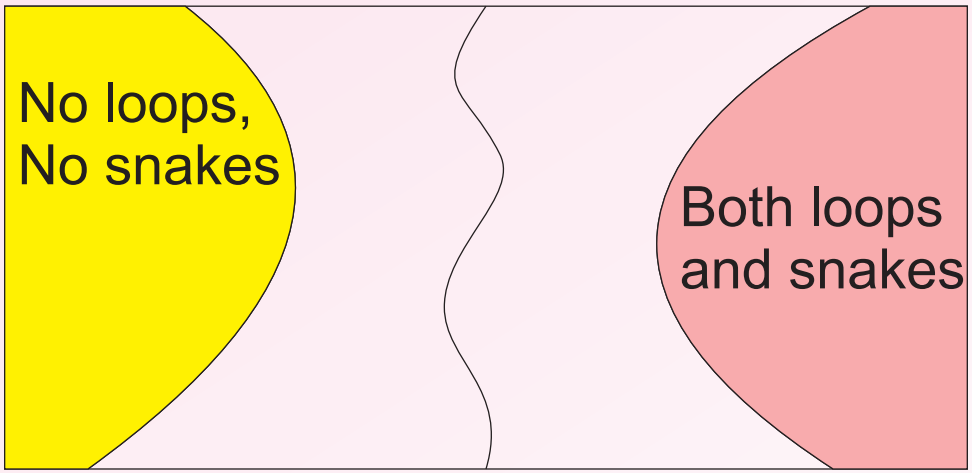


Recursively inseparable RE sets are known to exist.

The undecidability proofs for snakes and loops can be combined to show the rec. inseparability of the following two classes of tile sets:

A: Tile sets T that do not admit any infinite snakes or any loops (not even weak ones)

B: Tile sets T that admit arbitrarily long (even strong) loops ($\Longrightarrow$ admit also infinite snakes).



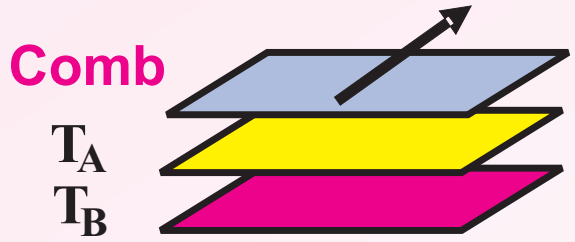
To prove rec. inseparability:

- Merge **Snakes** and **Loops** into a single tile set **Comb** with both PFP and RCP.
- Fix two rec. inseparable RE sets $A, B \subseteq \Sigma^*$.
- For any given $x \in \Sigma^*$ effectively construct tile sets T_A and T_B such that

  $x \in A \iff T_A$ admits a finite tiling,

  $x \in B \iff T_B$ admits no infinite tiling.

- Construct three layer sandwiches



Outline of the talk

1. Wang tiles and the Domino problem ✓
2. Two applications
   (A) Self-assembly ✓
   - Termination problem
   - Snake tiling problem
   - Loop tiling problem
   - Recursive inseparability

   (B) Cellular automata
   - Reversibility and surjectivity
   - Inseparability

Cellular automata (CA)

Cellular automata are among the oldest models of natural computing. They are investigated

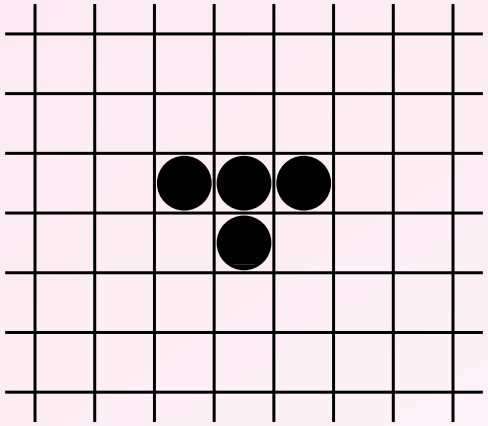
- **in physics** as discrete models of physical systems,
- **in computer science** as models of massively parallel computation under the realistic constraints of locality and uniformity,
- **in mathematics** as endomorphisms of the full shift in the context of symbolic dynamics.

Example: the **Game-of-Life** by John Conway.

- Infinite square grid whose squares (=cells) are colored black (=**alive**) or white (=**dead**).
- At each discrete time step each cell counts the number of living cells surrounding it, and based on this number determines its new state.
- All cells change their state simultaneously.

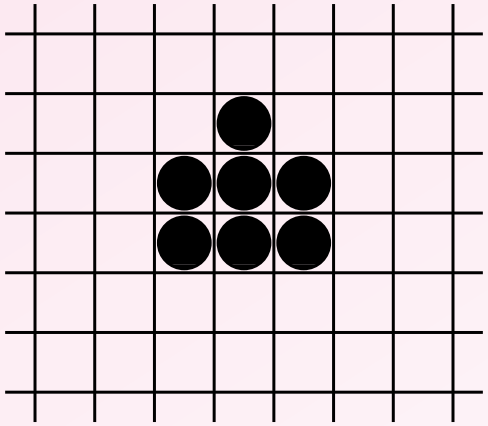
The local update rule asks each cell to check the present states of the eight surrounding cells.

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.



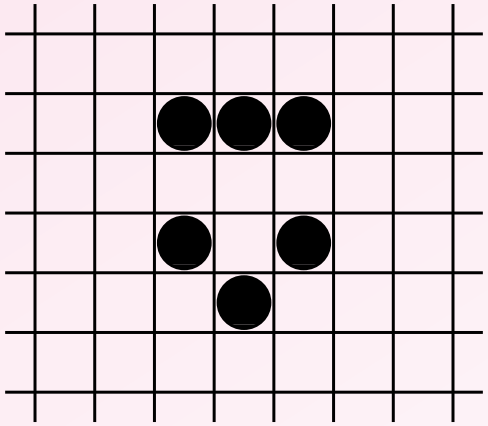
The local update rule asks each cell to check the present states of the eight surrounding cells.

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.



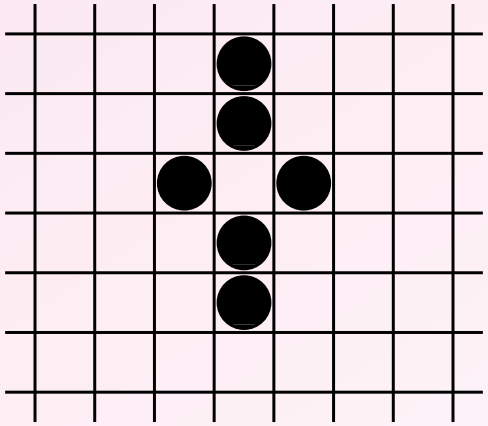
The local update rule asks each cell to check the present states of the eight surrounding cells.

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.



The local update rule asks each cell to check the present states of the eight surrounding cells.

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.



More generally: A **two-dimensional cellular automaton** (2D CA) over a finite **state set** S is determined by a **local update rule** that gives the new state of each cell from the old states of its neighbors.

The update rule is applied **synchronously** at all cells.

The resulting function

$$G : S^{\mathbb{Z}^2} \longrightarrow S^{\mathbb{Z}^2}$$

is the CA function.

Reversible CA

A CA is called

- **injective** if G is one-to-one,
- **surjective** if G is onto,
- **bijective** if G is both one-to-one and onto.

Reversible CA

A CA is called

- **injective** if G is one-to-one,
- **surjective** if G is onto,
- **bijective** if G is both one-to-one and onto.

A CA G is a **reversible** (RCA) if there is another CA function F that is its inverse, i.e.

$$G \circ F = F \circ G = \text{identity function}.$$

RCA G and F are called the **inverse automata** of each other.

A reversible CA must be bijective but also the converse is true:

Theorem (Curtis-Hedlund-Lyndon): A cellular automaton G is reversible if and only if it is bijective.

A reversible CA must be bijective but also the converse is true:

Theorem (Curtis-Hedlund-Lyndon): A cellular automaton G is reversible if and only if it is bijective.

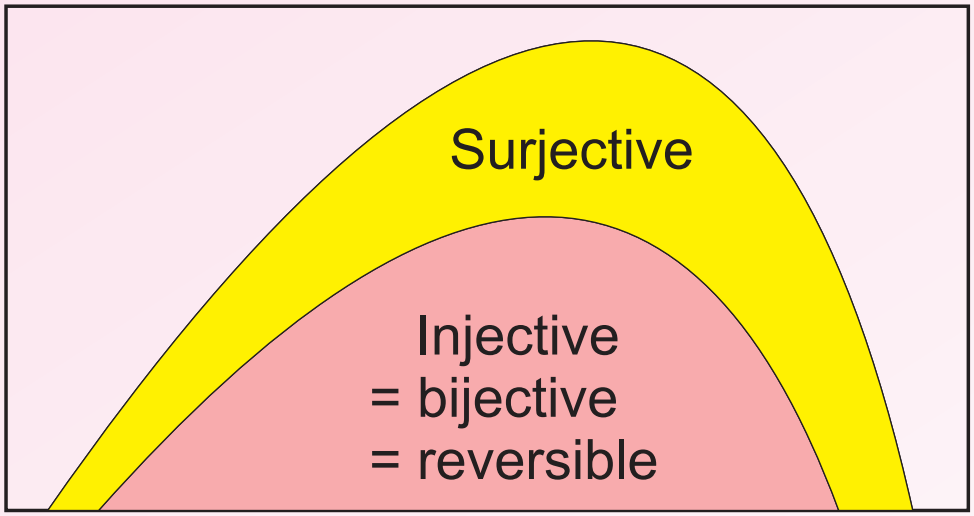
In bijective CA, each cell can determine its previous state by looking at the current states in some bounded neighborhood around it.

Easy to see:

$$G \text{ injective} \Longrightarrow G \text{ surjective}$$

so

$$\mathbf{\mathit{G}\ injective} \Longleftrightarrow \mathbf{\mathit{G}\ bijective} \Longleftrightarrow \mathbf{\mathit{G}\ reversible}$$



Decision problems...

The reversibility problem: Is a given CA reversible ?

The surjectivity problem: Is a given CA surjective ?

Both problems are undecidable for 2D CA.

Decision problems...

The reversibility problem: Is a given CA reversible ?

The surjectivity problem: Is a given CA surjective ?

Both problems are undecidable for 2D CA.

Remark:

- The **positive** instances of the **reversibility problem**
- The **negative** instances of the **surjectivity problem**

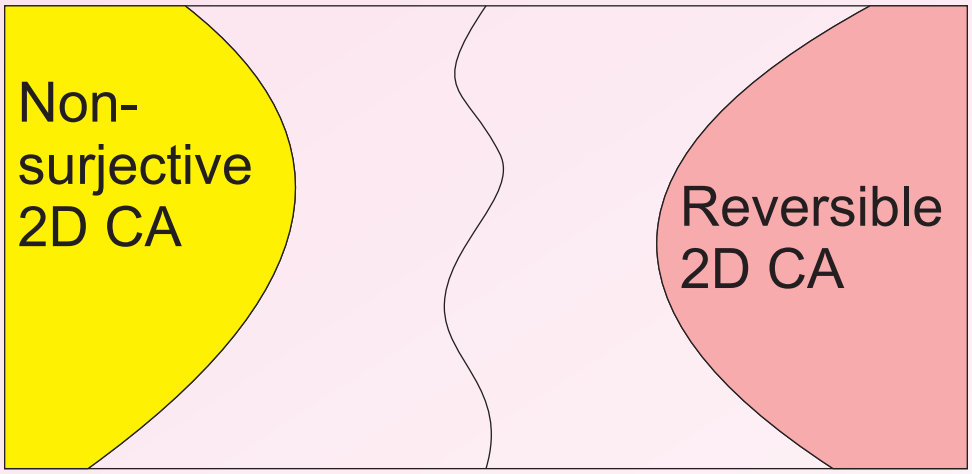
are semi-decidable.

Even better: The classes of

A: Reversible 2D CA

B: Non-surjective 2D CA

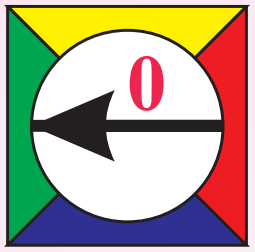
are recursively inseparable



Proof: direct reduction from the **snake** and **loop tiling problems**.

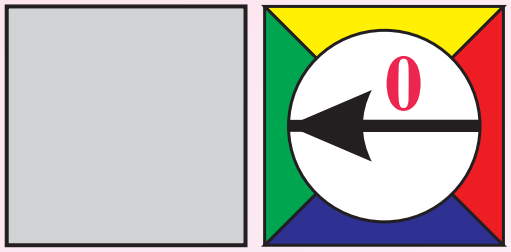
The reductions: For a given Wang tile set T, construct a 2D CA with the state set

$$T \times \{\leftarrow, \uparrow, \rightarrow, \downarrow\} \times \{0, 1\}.$$



The reductions: For a given Wang tile set T, construct a 2D CA with the state set

$$T \times \{\leftarrow, \uparrow, \rightarrow, \downarrow\} \times \{0, 1\}.$$

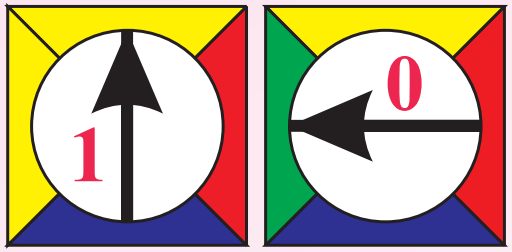


The neighbor in the direction of the arrow is the **follower**.

A cell is **active** iff the tile matches in color with the follower.

The reductions: For a given Wang tile set T, construct a 2D CA with the state set

$$T \times \{\leftarrow, \uparrow, \rightarrow, \downarrow\} \times \{0, 1\}.$$

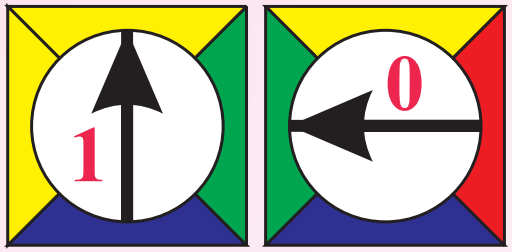


Local rule:

- Cell **inactive** (=no match) $\Longrightarrow$ **no change** in the cell.

The reductions: For a given Wang tile set T, construct a 2D CA with the state set

$$T \times \{\leftarrow, \uparrow, \rightarrow, \downarrow\} \times \{0, 1\}.$$

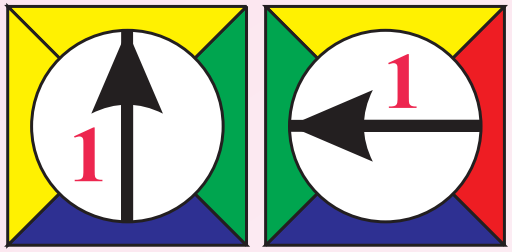


Local rule:

- Cell **inactive** (=no match) $\Longrightarrow$ **no change** in the cell.
- Cell **active** (=colors match) $\Longrightarrow$ the bit component is **XOR**ed with the bit of follower.

The reductions: For a given Wang tile set T, construct a 2D CA with the state set

$$T \times \{\leftarrow, \uparrow, \rightarrow, \downarrow\} \times \{0, 1\}.$$



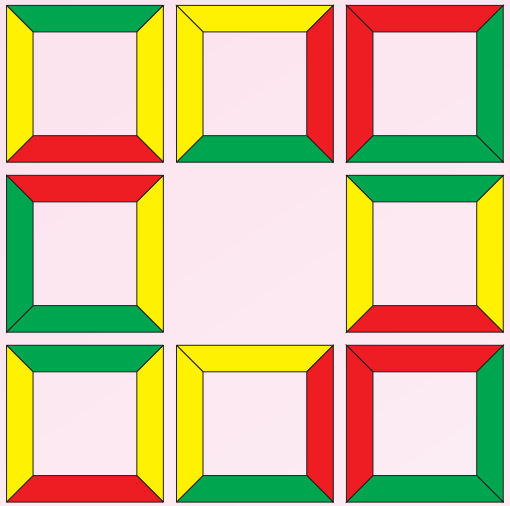
Local rule:

- Cell **inactive** (=no match) $\Longrightarrow$ **no change** in the cell.
- Cell **active** (=colors match) $\Longrightarrow$ the bit component is **XORed** with the bit of follower.

Claim 1: If T admits a weak loop, the CA is not surjective.

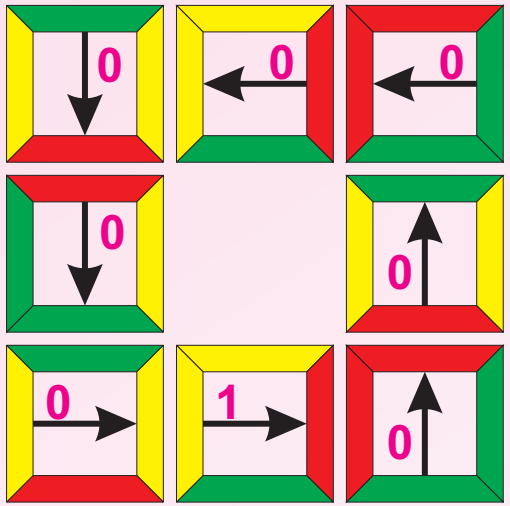
Claim 1: If T admits a weak loop, the CA is not surjective.

Proof: Assume a weak loop.



Claim 1: If T admits a weak loop, the CA is not surjective.

Proof. Assume a weak loop:



Pattern with no pre-image (single cell of the loop carries bit 1).

Claim 2: If the CA is not reversible then T admits a loop or a snake

Claim 2: If the CA is not reversible then T admits a loop or a snake

Proof. Assume two different configurations have the same image:

They differ in the bit components only.

Claim 2: If the CA is not reversible then T admits a loop or a snake

Proof. Assume two different configurations have the same image:

The cell must be active, and the bits in the follower must be different.

Claim 2: If the CA is not reversible then T admits a loop or a snake

Proof. Assume two different configurations have the same image:

The reasoning can be repeated on the follower. It must be active, and the bits in its follower are different.

Claim 2: If the CA is not reversible then T admits a loop or a snake

Proof. Assume two different configurations have the same image:

Repeat the reasoning over-and-over $\Longrightarrow$ A weak snake or a loop.

So we have

- T admits loops and snakes $\Longrightarrow$ CA is not surjective.
- T has no loops or snakes $\Longrightarrow$ CA is reversible.

Theorem: Reversibility and non-surjectivity are recursively inseparable among 2D CA.

Conclusions

Any property of 2D CA that is between reversibility and surjectivity is undecidable. Some examples of such properties:

- Openness,
- Closingness,
- Density of periodic orbits.

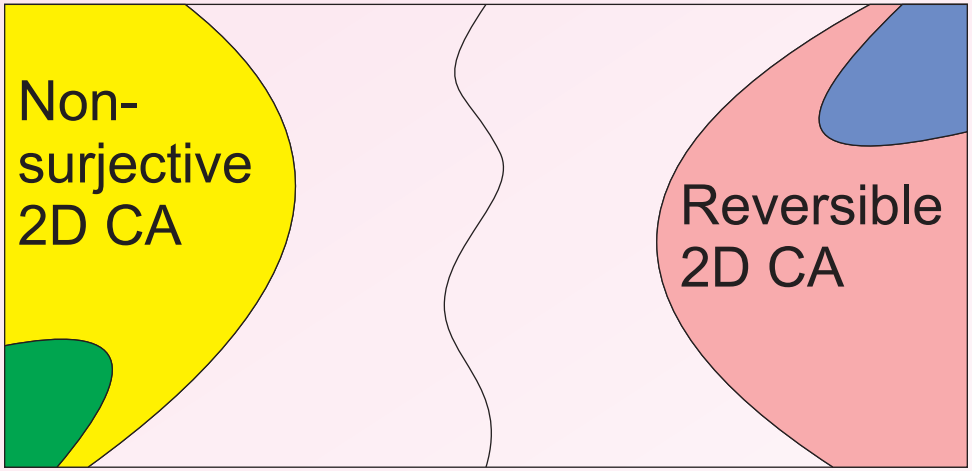
Density of periodic orbits is conjectured to be equivalent to surjectivity. We have no idea on the solution of this long standing open problem but, in any case, we know now that it is undecidable if a given 2D CA has dense periodic orbits.

Conclusions

It would be interesting to refine the inseparability result by shrinking the classes. For example, are

- periodic CA
- eventually periodic, but not periodic CA

recursively inseparable ?



Thank You