Cellular Automata and Tilings

Jarkko Kari

Department of Mathematics, University of Turku, Finland
TUCS (Turku Centre for Computer Science)
Topics of the four lectures

(1) Tutorial on Cellular automata
(2) Tilings and undecidability
(3) Snakes and tiles
(4) ? Surprise topic
Topics of the four lectures

(1) Tutorial on Cellular automata
   • Introduction and examples
   • General definitions
   • Topolgy & Curtis-Hedlund-Lyndom -theorem
   • Reversible CA
   • Surjective CA: balance, Garden-of-Eden -theorem
   • Finite and periodic configurations

(2) Tilings and undecidability

(3) Snakes and tiles

(4) ? Surprise topic
Cellular automata are among the oldest models of natural computing. They are versatile objects of study, 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.
Cellular automata possess several fundamental properties of the physical world: they are

- massively parallel,
- homogeneous in time and space,
- all interactions are local,
- time reversibility and conservation laws can be obtained by choosing the local update rule properly.
Example: the **Game-of-Life** by John Conway.

- Infinite checker-board 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.
A typical snapshot of a time evolution in Game-of-Life:

Initial uniformly random configuration.
A typical snapshot of a time evolution in Game-of-Life:

The next generation after all cells applied the update rule.
A typical snapshot of a time evolution in Game-of-Life:

Generation 10
A typical snapshot of a time evolution in Game-of-Life:

Generation 100
A typical snapshot of a time evolution in Game-of-Life:

GOL is a computationally universal two-dimensional CA.
Another famous universal CA: **rule 110** by S. Wolfram.

A one-dimensional CA with binary state set \{0, 1\}, i.e. a two-way infinite sequence of 0’s and 1’s.

Each cell is updated based on its old state and the states of its left and right neighbors as follows:

<table>
<thead>
<tr>
<th>Old State</th>
<th>New State</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
</tr>
</tbody>
</table>
Another famous universal CA: **rule 110** by S.Wolfram.

A one-dimensional CA with binary state set \(\{0, 1\}\), i.e. a two-way infinite sequence of 0’s and 1’s.

Each cell is updated based on its old state and the states of its left and right neighbors as follows:

\[
\begin{align*}
111 & \rightarrow 0 \\
110 & \rightarrow 1 \\
101 & \rightarrow 1 \\
100 & \rightarrow 0 \\
011 & \rightarrow 1 \\
010 & \rightarrow 1 \\
001 & \rightarrow 1 \\
000 & \rightarrow 0 
\end{align*}
\]

110 is the **Wolfram number** of this CA rule.
Space-time diagram is a pictorial representation of a time evolution in one-dimensional CA, where space and time are represented by the horizontal and vertical direction:
General definition of $d$-dimensional CA

- **Finite state set** $S$.
- **Configurations** are elements of $S^{\mathbb{Z}^d}$, i.e., functions $\mathbb{Z}^d \rightarrow S$ assigning states to cells,
- **A neighborhood vector**
  \[ N = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n) \]
  is a vector of $n$ distinct elements of $\mathbb{Z}^d$ that provide the relative offsets to neighbors.
- **The neighbors** of a cell at location $\vec{x} \in \mathbb{Z}^d$ are the $n$ cells at locations
  \[ \vec{x} + \vec{x}_i, \text{ for } i = 1, 2, \ldots, n. \]
Typical two-dimensional neighborhoods:

Von Neumann neighborhood
\{(0, 0), (\pm 1, 0), (0, \pm 1)\}

Moore neighborhood
\{-1, 0, 1\} \times \{-1, 0, 1\}
The **local rule** is a function

\[ f : S^n \rightarrow S \]

where \( n \) is the size of the neighborhood.

State \( f(a_1, a_2, \ldots, a_n) \) is the new state of a cell whose \( n \)
neighbors were at states \( a_1, a_2, \ldots, a_n \) one time step before.
The local update rule determines the global dynamics of the CA: Configuration $c$ becomes in one time step the configuration $e$ where, for all $\vec{x} \in \mathbb{Z}^d$,

$$e(\vec{x}) = f(c(\vec{x} + \vec{x}_1), c(\vec{x} + \vec{x}_2), \ldots, c(\vec{x} + \vec{x}_n)).$$

The transformation

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

that maps $c \mapsto e$ is the **global transition function** of the CA.

Function $G$ is our main object of study and we simply call it a **CA function**. In algorithmic questions we use its finite presentation (the local rule).
It is convenient to endow $S^\mathbb{Z}^d$ with the product topology. The topology is compact and induced by a metric.

The topology is generated by the cylinder sets.
A **cylinder** is a subset of $S^\mathbb{Z}^d$ determined by a **finite pattern**: Let $D \subset \mathbb{Z}^d$ be a finite domain, and $p : D \rightarrow S$ an assignment of states to the domain. The cylinder determined by $D, p$ is the set

$$\{ c \in S^{\mathbb{Z}^d} | \forall i \in D : c_i = p_i \}$$

of all configurations that agree with $p$ in domain $D$.

Cylinders are both open and closed: They form a clopen basis of the topology.
Under this topology, a sequence $c_1, c_2, \ldots$ of configurations converges to $c \in S^{\mathbb{Z}^d}$ if and only if for all cells $\vec{x} \in \mathbb{Z}^d$ and for all sufficiently large $i$ holds

$$c_i(\vec{x}) = c(\vec{x}).$$

Compactness of the topology means that all infinite sequences $c_1, c_2, \ldots$ of configurations have converging subsequences.
All cellular automata are continuous transformations

\[ S^\mathbb{Z}^d \longrightarrow S^\mathbb{Z}^d \]

under the topology. Indeed, locality of the update rule means that if

\[ c_1, c_2, \ldots \]

is a converging sequence of configurations then

\[ G(c_1), G(c_2), \ldots \]

converges as well, and

\[ \lim_{i \to \infty} G(c_i) = G(\lim_{i \to \infty} c_i). \]
The translation \( \tau \) determined by vector \( \vec{r} \in \mathbb{Z}^d \) is the transformation

\[
S_{\mathbb{Z}^d} \longrightarrow S_{\mathbb{Z}^d}
\]

that maps \( c \mapsto e \) where

\[
e(\vec{x}) = c(\vec{x} - \vec{r}) \text{ for all } \vec{x} \in \mathbb{Z}^d.
\]

(It is the CA whose local rule is the identity function and whose neighborhood consists of \(-\vec{r}\) alone.)

Translations determined by unit coordinate vectors \((0, \ldots, 0, 1, 0 \ldots, 0)\) are called \textit{shifts}.\]
Since all cells of a CA use the same local rule, the CA commutes with all translations:

\[ G \circ \tau = \tau \circ G. \]
We have seen that all CA are continuous, translation commuting maps $S^\mathbb{Z}_d \rightarrow S^\mathbb{Z}_d$.

The **Curtis-Hedlund- Lyndon theorem** from 1969 states that also the converse is true:

**Theorem:** A function $G : S^\mathbb{Z}_d \rightarrow S^\mathbb{Z}_d$ is a CA function if and only if

(i) $G$ is continuous, and

(ii) $G$ commutes with translations.
• The set $S^{\mathbb{Z}^d}$, together with the shift maps, is the $d$-dimensional full shift.

• Topologically closed, shift invariant subsets of $S^{\mathbb{Z}^d}$ are called subshifts.

• Cellular automata are the endomorphisms of the full shift.
Finite and periodic configurations

It is obviously not possible to simulate CA functions on arbitrary infinite configurations, but one has to limit the attention to some subset of $S^d$. We often consider the action on finite configurations or on periodic configurations.
**Finite configurations:** One state \( q \in S \) is often identified as the **quiescent** state, and it is expected to be stable:

\[
f(q, q, \ldots, q) = q.
\]

A configuration \( c \in S^{\mathbb{Z}^d} \) is called **finite** if the set

\[
\{ \vec{n} \in \mathbb{Z}^d \mid c(\vec{n}) \neq q \}
\]

is finite.

Due to stability of \( q \), CA \( G \) maps finite configurations to finite configurations.
Periodic configurations: Configuration $c \in S^{\mathbb{Z}^d}$ has period $\vec{r} \in \mathbb{Z}^d$ if it is invariant under the translation $\tau$ by $\vec{r}$:

$$\tau(c) = c.$$ 

CA functions commute with translations, so we also have

$$\tau(G(c)) = G(\tau(c)) = G(c).$$

Period $\vec{r}$ of $c$ is also a period of $G(c)$. 
Configuration $c \in S^{\mathbb{Z}^d}$ is (fully) periodic if it has $d$ linearly independent periods.
Configuration $c \in S^{\mathbb{Z}^d}$ is (fully) \textbf{periodic} if it has $d$ linearly independent periods.
Configuration $c \in S^{\mathbb{Z}^d}$ is (fully) **periodic** if it has $d$ linearly independent periods.
Finite configurations and periodic configurations are **dense** in $S^{\mathbb{Z}^d}$: each cylinder contains finite and periodic configurations.

We denote

- $G_F$ for the restriction of $G$ on the set of finite configurations,
- $G_P$ for the restriction of $G$ on the set of periodic configurations.
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 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.
Game-of-Life and Rule 110 are irreversible: Configurations may have several pre-images.
Two-dimensional **Q2R** Ising model by G.Vichniac (1984) is an example of a reversible cellular automaton.

Each cell has a spin that is directed either up or down. The direction of a spin is swapped if and only if among the four immediate neighbors there are exactly two cells with spin up and two cells with spin down:

```
  ↓  ↓  ↓  ↓
↑   ↓   ↓   ↓
  ↑  ↑  ↑  ↑
```
The twist that makes the Q2R rule reversible: Color the space as a checker-board. On even time steps only update the spins of the white cells and on odd time steps update the spins of the black cells.
The twist that makes the Q2R rule reversible: Color the space as a checker-board. On even time steps only update the spins of the white cells and on odd time steps update the spins of the black cells.
The twist that makes the Q2R rule reversible: Color the space as a checker-board. On even time steps only update the spins of the white cells and on odd time steps update the spins of the black cells.
The twist that makes the Q2R rule reversible: Color the space as a checker-board. On even time steps only update the spins of the white cells and on odd time steps update the spins of the black cells.
Q2R is **reversible**: The same rule (applied again on squares of the same color) reconstructs the previous generation.

Q2R rule also exhibits a local **conservation law**: The number of neighbors with opposite spins remains constant over time.
Evolution of Q2R from an uneven random distribution of spins:

Initial random configuration with 8% spins up.
Evolution of Q2R from an uneven random distribution of spins:

One million steps. The length of the B/W boundary is invariant.
From the Curtis-Hedlund-Lyndom theorem we get

**Corollary:** A cellular automaton $G$ is reversible if and only if it is bijective.

**Proof:** If $G$ is a reversible CA function then $G$ is by definition bijective.

Conversely, suppose that $G$ is a bijective CA function. Then $G$ has an inverse function $G^{-1}$ that clearly commutes with the shifts. The inverse function $G^{-1}$ is also continuous because the space $S^{\mathbb{Z}^d}$ is compact. It now follows from the Curtis-Hedlund-Lyndon theorem that $G^{-1}$ is a cellular automaton. \qed
The point of the corollary is that in bijective CA each cell can determine its previous state by looking at the current states in some bounded neighborhood around them.
Configurations that do not have a pre-image are called Garden-Of-Eden -configurations. Only non-surjective CA have GOE configurations.

A finite pattern consists of a finite domain $D \subseteq \mathbb{Z}^d$ and an assignment

$$p : D \rightarrow S$$

of states.

Finite pattern is called an orphan for CA $G$ if every configuration containing the pattern is a GOE.
From the compactness of $S^\mathbb{Z}_d$ we directly get:

**Proposition.** Every GOE configuration contains an orphan pattern.

Non-surjectivity is hence equivalent to the existence of orphans.
Balance in surjective CA

All surjective CA have balanced local rules: for every $a \in S$

$$|f^{-1}(a)| = |S|^{n-1}.$$
Balance in surjective CA

All surjective CA have balanced local rules: for every \( a \in S \)

\[ |f^{-1}(a)| = |S|^{n-1}. \]

Indeed, consider a non-balanced local rule such as rule 110 where five contexts give new state 1 while only three contexts give state 0:

\[
\begin{align*}
111 & \rightarrow 0 \\
110 & \rightarrow 1 \\
101 & \rightarrow 1 \\
100 & \rightarrow 0 \\
011 & \rightarrow 1 \\
010 & \rightarrow 1 \\
001 & \rightarrow 1 \\
000 & \rightarrow 0
\end{align*}
\]
Consider finite patterns where state 0 appears in every third position. There are $2^{2(k-1)} = 4^{k-1}$ such patterns where $k$ is the number of 0’s.
Consider finite patterns where state 0 appears in every third position. There are \(2^{2(k-1)} = 4^{k-1}\) such patterns where \(k\) is the number of 0’s.

A pre-image of such a pattern must consist of \(k\) segments of length three, each of which is mapped to 0 by the local rule. There are \(3^k\) choices.

As for large values of \(k\) we have \(3^k < 4^{k-1}\), there are fewer choices for the red cells than for the blue ones. Hence some pattern has no pre-image and it must be an orphan. \(\square\)
One can also verify directly that pattern

```
01010
```

is an orphan of rule 110. It is the shortest orphan.
Balance of the local rule is not sufficient for surjectivity. For example, the majority CA (Wolfram number 232) is a counter example. The local rule

\[ f(a, b, c) = 1 \text{ if and only if } a + b + c \geq 2 \]

is clearly balanced, but 01001 is an orphan.
The balance property of surjective CA generalizes to finite patterns of arbitrary shape:

**Theorem:** Let $G$ be surjective. Let $M, D \subseteq \mathbb{Z}^d$ be finite domains such that $D$ contains the neighborhood of $M$. Then every finite pattern with domain $M$ has the same number

$$n|D| - |M|$$

of pre-images in domain $D$, where $n$ is the number of states. □
The balance property of surjective CA generalizes to finite patterns of arbitrary shape:

**Theorem:** Let $G$ be surjective. Let $M, D \subseteq \mathbb{Z}^d$ be finite domains such that $D$ contains the neighborhood of $M$. Then every finite pattern with domain $M$ has the same number

$$n|D| - |M|$$

of pre-images in domain $D$, where $n$ is the number of states. $\square$

The balance property means that the uniform probability measure is **invariant** for surjective CA. (Uniform randomness is preserved by surjective CA.)
Garden-Of-Eden -theorem

Let us call configurations $c_1$ and $c_2$ asymptotic if the set

$$\text{diff}(c_1, c_2) = \{ \vec{n} \in \mathbb{Z}^d \mid c_1(\vec{n}) \neq c_2(\vec{n}) \}$$

of positions where $c_1$ and $c_2$ differ is finite.

A CA is called pre-injective if any asymptotic $c_1 \neq c_2$ satisfy $G(c_1) \neq G(c_2)$. 
The **Garden-Of-Eden -theorem** by Moore (1962) and Myhill (1963) connects surjectivity with pre-injectivity.

**Theorem:** CA $G$ is surjective if and only if it is pre-injective.
The Garden-Of-Eden -theorem by Moore (1962) and Myhill (1963) connects surjectivity with pre-injectivity.

**Theorem:** CA $G$ is surjective if and only if it is pre-injective.

The proof idea can be easily explained using rule 110 as a running example.
1) $G$ not surjective $\implies G$ not pre-injective:

Since rule 110 is not surjective it has an orphan 01010 of length five. Consider a segment of length $5k - 2$, for some $k$, and configurations $c$ that are in state 0 outside this segment. There are $2^{5k-2} = 32^k/4$ such configurations.
1) $G$ not surjective $\implies G$ not pre-injective:

The non-0 part of $G(c)$ is within a segment of length $5k$. Partition this segment into $k$ parts of length 5. Pattern 01010 cannot appear in any part, so only $2^5 - 1 = 31$ different patterns show up in the subsegments. There are at most $31^k$ possible configurations $G(c)$. 

![Diagram](image-url)
1) $G$ not surjective $\implies G$ not pre-injective:

The non-0 part of $G(c)$ is within a segment of length $5k$. Partition this segment into $k$ parts of length 5. Pattern 01010 cannot appear in any part, so only $2^5 - 1 = 31$ different patterns show up in the subsegments. There are at most $31^k$ possible configurations $G(c)$.

As $32^k/4 > 31^k$ for large $k$, there are more choices for red than blue segments. So there must exist two different red configurations with the same image. □
2) \( G \) not pre-injective \( \implies G \) not surjective:

In rule 110

\[
\begin{array}{c}
\text{p} \\
0011101000 \\
\downarrow \\
1111110
\end{array}
\quad \begin{array}{c}
\text{q} \\
00101100 \\
\downarrow \\
1111110
\end{array}
\]

so patterns \( p \) and \( q \) of length 8 can be exchanged to each other in any configuration without affecting its image. There exist just

\[2^8 - 1 = 255\]

essentially different blocks of length 8.
2) $G$ not pre-injective $\implies G$ not surjective:

Consider a segment of $8k$ cells, consisting of $k$ parts of length 8. Patterns $p$ and $q$ are exchangeable, so the segment has at most $255^k$ different images.
2) \( G \) not pre-injective \( \implies G \) not surjective:

Consider a segment of \( 8k \) cells, consisting of \( k \) parts of length 8. Patterns \( p \) and \( q \) are exchangeable, so the segment has at most \( 255^k \) different images.

There are, however, \( 2^{8k-2} = 256^k/4 \) different patterns of size \( 8k - 2 \). Because \( 255^k < 256^k/4 \) for large \( k \), there are blue patterns without any pre-image.
Garden-Of-Eden theorem: CA $G$ is surjective if and only if it is pre-injective.
Garden-Of-Eden -theorem: CA $G$ is surjective if and only if it is pre-injective.

Corollary: Every injective CA is also surjective. Injectivity, bijectivity and reversibility are equivalent concepts.

Proof: If $G$ is injective then it is pre-injective. The claim follows from the Garden-Of-Eden -theorem.
Examples:

The majority rule is not surjective: finite configurations

\[ \ldots 0000000 \ldots \quad \text{and} \quad \ldots 0001000 \ldots \]

have the same image, so $G$ is not pre-injective. Pattern

\[ 01001 \]

is an orphan.
Examples:

In Game-Of-Life a lonely living cell dies immediately, so $G$ is not pre-injective. GOL is hence not surjective.
Interestingly, no small orphans are known for Game-Of-Life. Currently, the smallest known orphan consists of 92 cells (56 life, 36 dead):

M. Heule, C. Hartman, K. Kwekkeboom, A. Noels (2011)
Examples:

The **Traffic CA** is the elementary CA number 226.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>→ 1</td>
</tr>
<tr>
<td>110</td>
<td>→ 1</td>
</tr>
<tr>
<td>101</td>
<td>→ 1</td>
</tr>
<tr>
<td>100</td>
<td>→ 0</td>
</tr>
<tr>
<td>011</td>
<td>→ 0</td>
</tr>
<tr>
<td>010</td>
<td>→ 0</td>
</tr>
<tr>
<td>001</td>
<td>→ 1</td>
</tr>
<tr>
<td>000</td>
<td>→ 0</td>
</tr>
</tbody>
</table>

The local rule replaces pattern 01 by pattern 10.
<table>
<thead>
<tr>
<th>Binary</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
</tr>
</tbody>
</table>
111 → 1
110 → 1
101 → 1
100 → 0
011 → 0
010 → 0
001 → 1
000 → 0
<table>
<thead>
<tr>
<th>Binary</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
</tr>
</tbody>
</table>

The table shows the binary numbers and their corresponding results. Each row represents a binary number, and the right column indicates the result of applying a specific operation or function to that binary number.
<table>
<thead>
<tr>
<th>Binary</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>→ 1</td>
</tr>
<tr>
<td>110</td>
<td>→ 1</td>
</tr>
<tr>
<td>101</td>
<td>→ 1</td>
</tr>
<tr>
<td>100</td>
<td>→ 0</td>
</tr>
<tr>
<td>011</td>
<td>→ 0</td>
</tr>
<tr>
<td>010</td>
<td>→ 0</td>
</tr>
<tr>
<td>001</td>
<td>→ 1</td>
</tr>
<tr>
<td>000</td>
<td>→ 0</td>
</tr>
</tbody>
</table>
The local rule is balanced. However, there are two finite configurations with the same successor:

and hence traffic CA is not surjective.
There is an orphan of size four:
Implications

\[
\begin{align*}
\mathbf{G} \text{ injective} & \implies \mathbf{G}_F \text{ injective} \\
\mathbf{G} \text{ injective} & \implies \mathbf{G}_P \text{ injective}
\end{align*}
\]

are trivial.
Implications

\[ G \text{ injective} \implies G_F \text{ injective} \]
\[ G \text{ injective} \implies G_P \text{ injective} \]

are trivial.

\[ G_F \text{ surjective} \implies G \text{ surjective} \]
\[ G_P \text{ surjective} \implies G \text{ surjective} \]

follow from the denseness of the finite and periodic configurations in \( S^{\mathbb{Z}^d} \).
For $G_F$ we have

\[ G \text{ reversible } \implies G_F \text{ surjective} \]
For $G_F$ we have

\[ G \text{ reversible} \implies G_F \text{ surjective} \]

Indeed, the quiescent state $q$ is also stable for $G^{-1}$, so the pre-image $G^{-1}(c)$ of a finite configuration $c$ is also finite.
Also we have

\[ G \text{ pre-injective} \iff G_F \text{ injective} \]
Also we have

\[ G \text{ pre-injective } \iff G_F \text{ injective} \]

- finite configurations are asymptotic with each other.

- for any asymptotic \( c_1, c_2 \) such that \( G(c_1) = G(c_2) \) we obtain corresponding finite \( e_1, e_2 \) such that \( G(e_1) = G(e_2) \) by changing into \( q \) all cells not interacting with

\[
\text{diff}(c_1, c_2) = \{ \vec{n} \in \mathbb{Z}^d | c_1(\vec{n}) \neq c_2(\vec{n}) \} 
\]
$G$ injective $\leftrightarrow$ $G$ bijective $\leftrightarrow$ $G$ reversible

$G_F$ surjective $\leftrightarrow$ $G_F$ bijective

$G$ surjective $\leftrightarrow$ $G_F$ injective $\leftrightarrow$ $G$ pre-injective
The xor-CA is the binary state CA with neighborhood \((0, 1)\) and local rule

\[
f(a, b) = a + b \pmod{2}.
\]

In the xor-CA every configuration has exactly two pre-images, so \(G\) is surjective but not injective:

One can freely choose one value in the pre-image, after which all remaining states are uniquely determined by the \textit{left-permutatativity} and the \textit{right-permutatativity} of xor.
The xor-CA is the binary state CA with neighborhood \((0, 1)\) and local rule
\[ f(a, b) = a + b \pmod{2}. \]

In the xor-CA every configuration has exactly two pre-images, so \(G\) is surjective but not injective:

\[
\begin{array}{ccccccccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
\]

One can freely choose one value in the pre-image, after which all remaining states are uniquely determined by the \textit{left-permutativity} and the \textit{right-permutativity} of xor.
The xor-CA is the binary state CA with neighborhood \((0, 1)\) and local rule

\[ f(a, b) = a + b \pmod{2}. \]

In the xor-CA every configuration has exactly two pre-images, so \( G \) is surjective but not injective:

One can freely choose one value in the pre-image, after which all remaining states are uniquely determined by the **left-permutativity** and the **right-permutativity** of xor.
The xor-CA is the binary state CA with neighborhood \((0, 1)\) and local rule
\[
f(a, b) = a + b \pmod{2}.
\]

In the xor-CA every configuration has exactly two pre-images, so \(G\) is surjective but not injective:

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}
\]

One can freely choose one value in the pre-image, after which all remaining states are uniquely determined by the left-permutativity and the right-permutativity of xor.
The xor-CA is the binary state CA with neighborhood \((0, 1)\) and local rule

\[
f(a, b) = a + b \pmod{2}.
\]

In the xor-CA every configuration has exactly two pre-images, so \(G\) is surjective but not injective:

One can freely choose one value in the pre-image, after which all remaining states are uniquely determined by the left-permutativity and the right-permutativity of xor.
The xor-CA is the binary state CA with neighborhood \((0, 1)\) and local rule

\[ f(a, b) = a + b \pmod{2}. \]

In the xor-CA every configuration has exactly two pre-images, so \(G\) is surjective but not injective:

One can freely choose one value in the pre-image, after which all remaining states are uniquely determined by the left-permutativity and the right-permutativity of xor.
The two pre-images of the finite configuration
\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
are both infinite:
\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]
\[
\begin{array}{cccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
So \( G_F \) is not surjective.
The **controlled-xor** CA has two binary tracks. The lower track does not change, but it indicates which cells are active. Non-active cells do not change at all, while active cells apply the xor rule on the first track:

```
1 0 1 0 0 1 0 1 1 0 0 0 1 1 0
A A A A A A
```

Non-active cells are shown in blue and active cells in gray.
The **controlled-xor** CA has two binary tracks. The lower track does not change, but it indicates which cells are active. Non-active cells do not change at all, while active cells apply the xor rule on the first track:
The **controlled-xor** CA has two binary tracks. The lower track does not change, but it indicates which cells are active. Non-active cells do not change at all, while active cells apply the xor rule on the first track:

The quiescent state is 0
The **controlled-xor** CA has two binary tracks. The lower track does not change, but it indicates which cells are active. Non-active cells do not change at all, while active cells apply the xor rule on the first track:

The quiescent state is

The CA is not injective as all active 0’s and all active 1’s have the same image, but every finite configuration has a finite pre-image, so $G_F$ is surjective.
For $G_P$ we have

$G_P$ injective $\implies G_P$ surjective
For $G_P$ we have

$$G_P \text{ injective } \implies G_P \text{ surjective}$$

Indeed, fix any $d$ linearly independent periods, and let $A \subseteq S^{\mathbb{Z}^d}$ be the set of configurations with these periods. Then

- $A$ is finite,
- $G$ is injective on $A$,
- $G(A) \subseteq A$.

We conclude that $G(A) = A$, and every periodic configuration has a periodic pre-image.
G injective $\iff$ G bijective $\iff$ G reversible

G surjective $\iff$ G injective

G pre-injective $\iff$ G surjective $\iff$ G injective $\iff$ G pre-injective

G_p surjective

G_F surjective $\iff$ G_F bijective

controlled XOR

XOR

G_F injective $\iff$ G_F bijective

G_p injective
Here we get the first **dimension sensitive** property. The following equivalences hold among one-dimensional CA:

\[
\begin{align*}
G \text{ injective} & \iff G_P \text{ injective} \\
G \text{ surjective} & \iff G_P \text{ surjective}
\end{align*}
\]

• We see in the third lecture that the first equivalence is not true among two-dimensional CA: there are cellular automata that are injective on periodic configurations but merge some non-periodic ones.

• It is not known whether the second equivalence is true in general.
Only in 1D

G injective ⇔ G bijective ⇔ G reversible ⇔ $G_P$ injective

$G_F$ surjective ⇔ $G_F$ bijective

G surjective ⇔ $G_F$ injective ⇔ G pre-injective ⇔ $G_P$ surjective
In 2D

G injective ↔ G bijective ↔ G reversible

G surjective ↔ G injective

G pre-injective

G surjective ↔ G_F injective ↔ G_pre-injective

G_F surjective ↔ G_F bijective

G_P surjective

G_P injective

Snake-XOR

controlled XOR

XOR

XOR
In 2D

G injective $\iff$ G bijective $\iff$ G reversible

G surjective $\iff$ G injective $\iff$ G pre-injective

G_{F} surjective $\iff$ G_{F} bijective

G_{P} injective

Snake-XOR

controlled XOR

XOR

G_{P} surjective

?
We have two proofs that injective \( CA \) are surjective:

\[
\begin{align*}
G \text{ injective} & \implies G \text{ pre-injective} \implies G \text{ surjective} \\
G \text{ injective} & \implies G_P \text{ injective} \implies G_P \text{ surjective} \implies G \text{ surjective}
\end{align*}
\]
We have two proofs that injective CA are surjective:

\[ \mathbb{G} \text{ injective} \implies \mathbb{G} \text{ pre-injective} \implies \mathbb{G} \text{ surjective} \]
\[ \mathbb{G} \text{ injective} \implies \mathbb{G}_P \text{ injective} \implies \mathbb{G}_P \text{ surjective} \implies \mathbb{G} \text{ surjective} \]

It is good to have both implication chains available, if one wants to generalize results to cellular automata whose underlying grid is not \( \mathbb{Z}^d \) but some other group.

- The first chain generalizes to all \textit{amenable} groups.
- The second chain generalizes to \textit{residually finite} groups.

A group is called \textit{surjunctive} if every injective CA on the group is also surjective. It is not known if all groups are surjunctive.