

# Identification of partially observed deterministic and stochastic cellular automata

**Witold Bolt**

Systems Research Institute, Polish Academy of Sciences

**Warsaw, 20.06.2024**

This research was supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing (MPD). Project financed from The European Union within the Innovative Economy Operational Programme 2007-2013 and European Regional Development Fund.



prof. dr hab. **Danuta Makowiec**  
1957 - 2024

# Contents of the presented dissertation

## Identification of Deterministic Cellular Automata:

[B1] W. Bołt, J. M. Baetens, and B. De Baets, “*An evolutionary approach to the identification of cellular automata based on partial observations*,” in 2015 IEEE Congress on Evolutionary Computation (CEC), 2015.

[B5] W. Bołt, J. M. Baetens, and B. De Baets, “*Identification of cellular automata based on incomplete observations with bounded time gaps*,” IEEE Transactions on Cybernetics, vol. 50, no. 3, 2020.

## Identification of Stochastic Cellular Automata:

[B2] W. Bołt, B. Wolnik, J. M. Baetens, and B. De Baets, “*On the identification of  $\alpha$ -asynchronous cellular automata in the case of partial observations with spatially separated gaps*,” in Challenging Problems and Solutions in Intelligent Systems, G. De Trè, P. Grzegorzewski, J. Kacprzyk, J. W. Owsinski, W. Penczek, and S. Zadrozny, Eds. Springer International Publishing, 2016.

[B3] W. Bołt, A. Bołt, B. Wolnik, J. M. Baetens, and B. De Baets, “*A statistical approach to the identification of diploid cellular automata*,” in Theory and Practice of Natural Computing, C. Martín-Vide, R. Neruda, and M. A. Vega-Rodríguez, Eds., Springer International Publishing, 2017.

[B4] W. Bołt, A. Bołt, B. Wolnik, J. M. Baetens, and B. De Baets, “*A statistical approach to the identification of diploid cellular automata based on incomplete observations*,” Biosystems, vol. 186, 2019.

# Identification of Deterministic Cellular Automata

# What is a Cellular Automaton (CA)?

- **Space:** 1D space divided into **finite** number ( $N > 0$ ) of indivisible **cells**
- **Time:** discrete, unit-less ticks of a “clock”  $t = 0, 1, 2, 3, \dots$
- **States:** finite state set (in our case:  $\{0, 1\}$ )
- **Configuration:** assignment of states to cells, here a vector  $\mathbf{x} \in \{0, 1\}^N$
- **Dynamics:** deterministic evolution of configurations (*change of states*) according to a **rule**

# What is a Cellular Automaton (CA)?

- Let  $\mathbf{x} \in \{0,1\}^N$  be a **configuration**. Let  $F: \{0,1\}^N \rightarrow \{0,1\}^N$  be a function, and  $F(\mathbf{x}) = \mathbf{x}'$ .
- $F$  is a **global rule** of a 1D, binary CA if there exists  $r > 0$  and a function  $f: \{0,1\}^{2r+1} \rightarrow \{0,1\}$  such that:

$$x'_i = f(x_{i-r}, \dots, x_i, \dots, x_{i+r}),$$

where operations on indices are applied modulo  $N$  (**periodic boundary conditions**).

- Sequence  $(x_{i-r}, \dots, x_{i+r})$  is the **neighborhood configuration** of the  $i$ -th cell.
- Such  $f$  is the **local rule**, and the number  $r$  the **radius of the neighborhood**.
- Let  $T > 0$ . The sequence of **configurations**  $(\mathbf{x}, F(\mathbf{x}), F(F(\mathbf{x})), F^3(\mathbf{x}), \dots, F^{T-1}(\mathbf{x}))$  is called a **space-time diagram** starting from the **initial configuration**  $\mathbf{x}$ .

# Local rule and the LUT

- Local rule  $f$  **does not** depend on the choice of  $N$  and can be an **arbitrary** function.
- A local rule can be expressed as a lookup table (LUT) by listing all possible inputs and outputs.
- An example for an **Elementary CA (ECA)**, where  $r = 1$ , is given below:

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ \ell_7 & \ell_6 & \ell_5 & \ell_4 & \ell_3 & \ell_2 & \ell_1 & \ell_0 \end{bmatrix},$$

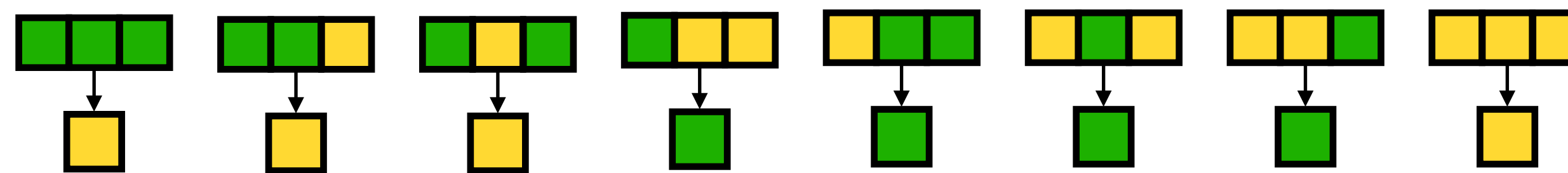
where essentially only the **second row** needs to be stored.

- Therefore a **local rule**  $f$  is equivalent to its LUT **vector**.



# Local rule and the LUT

- Local rule  $f$  **does not** depend on the choice of  $N$  and can be an **arbitrary** function.
- A local rule can be expressed as a lookup table (LUT) by listing all possible inputs and outputs.
- An example for an **Elementary CA (ECA)**, where  $r = 1$ , is given below:

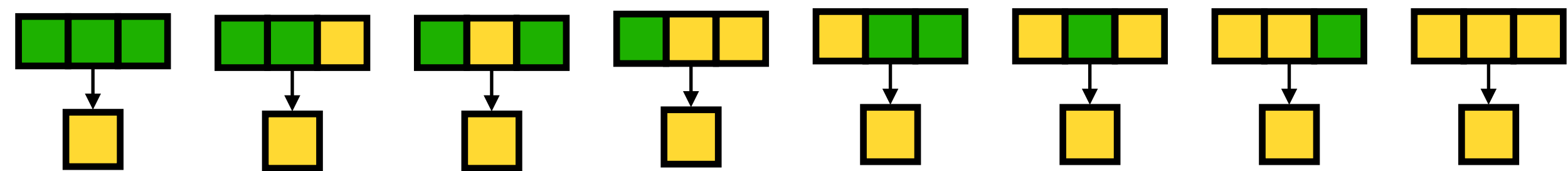


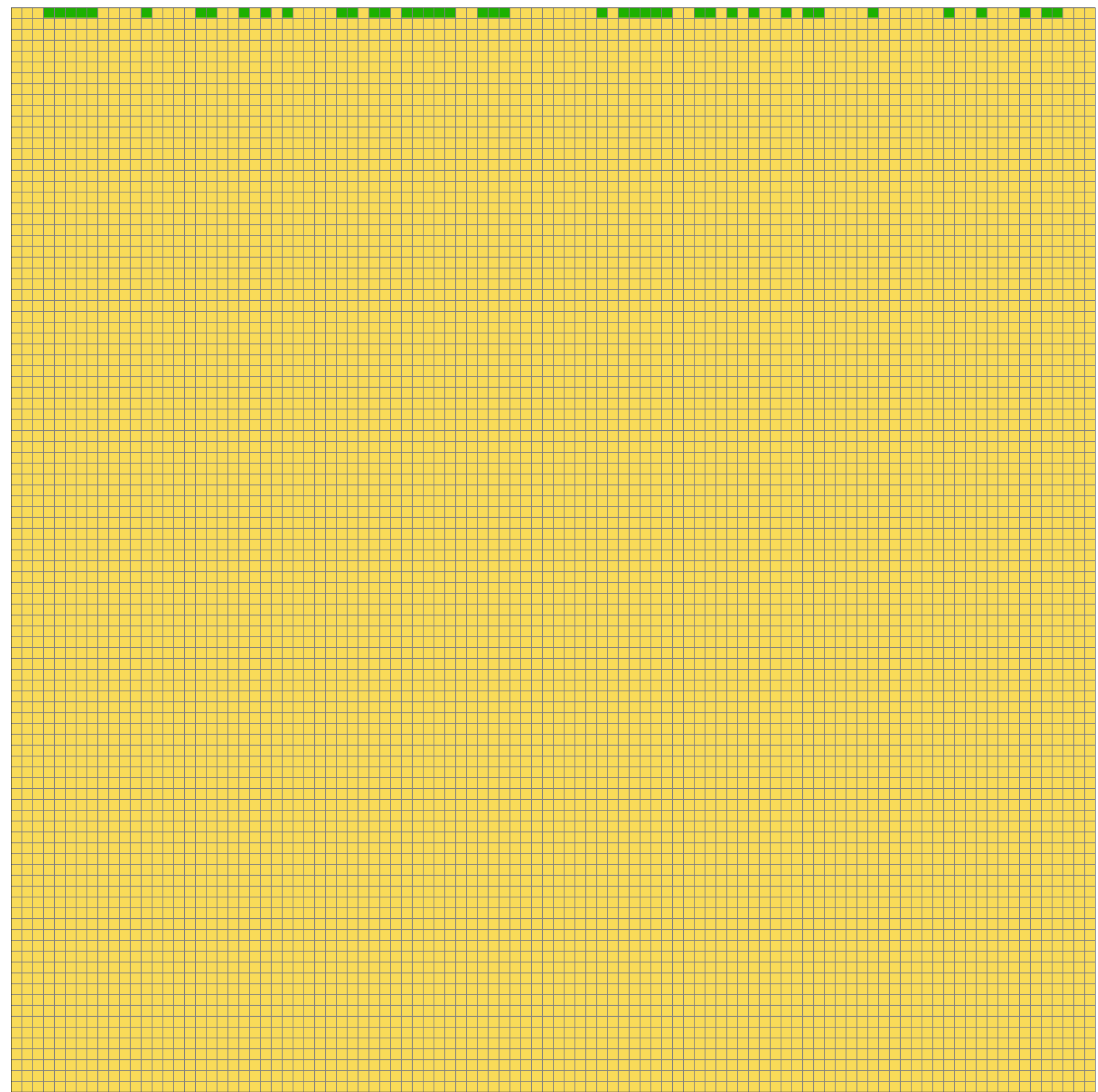
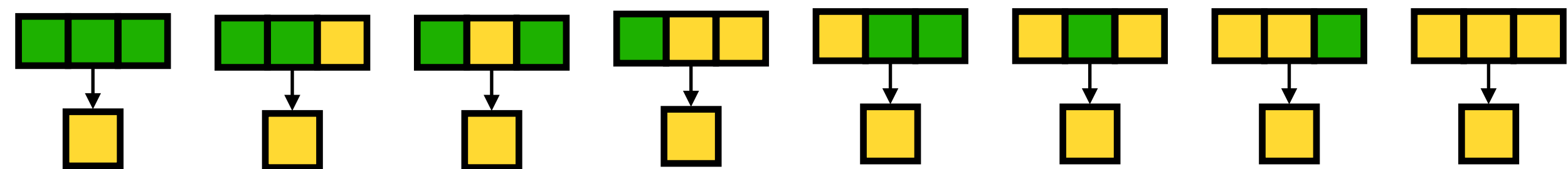
Yellow square = 0      Green square = 1

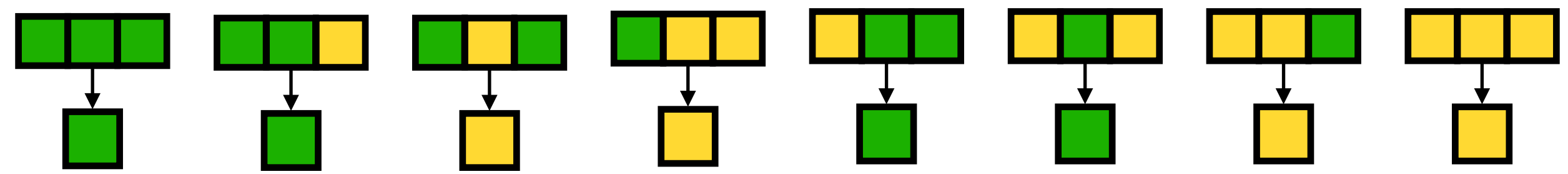
where essentially only the **second row** needs to be stored.

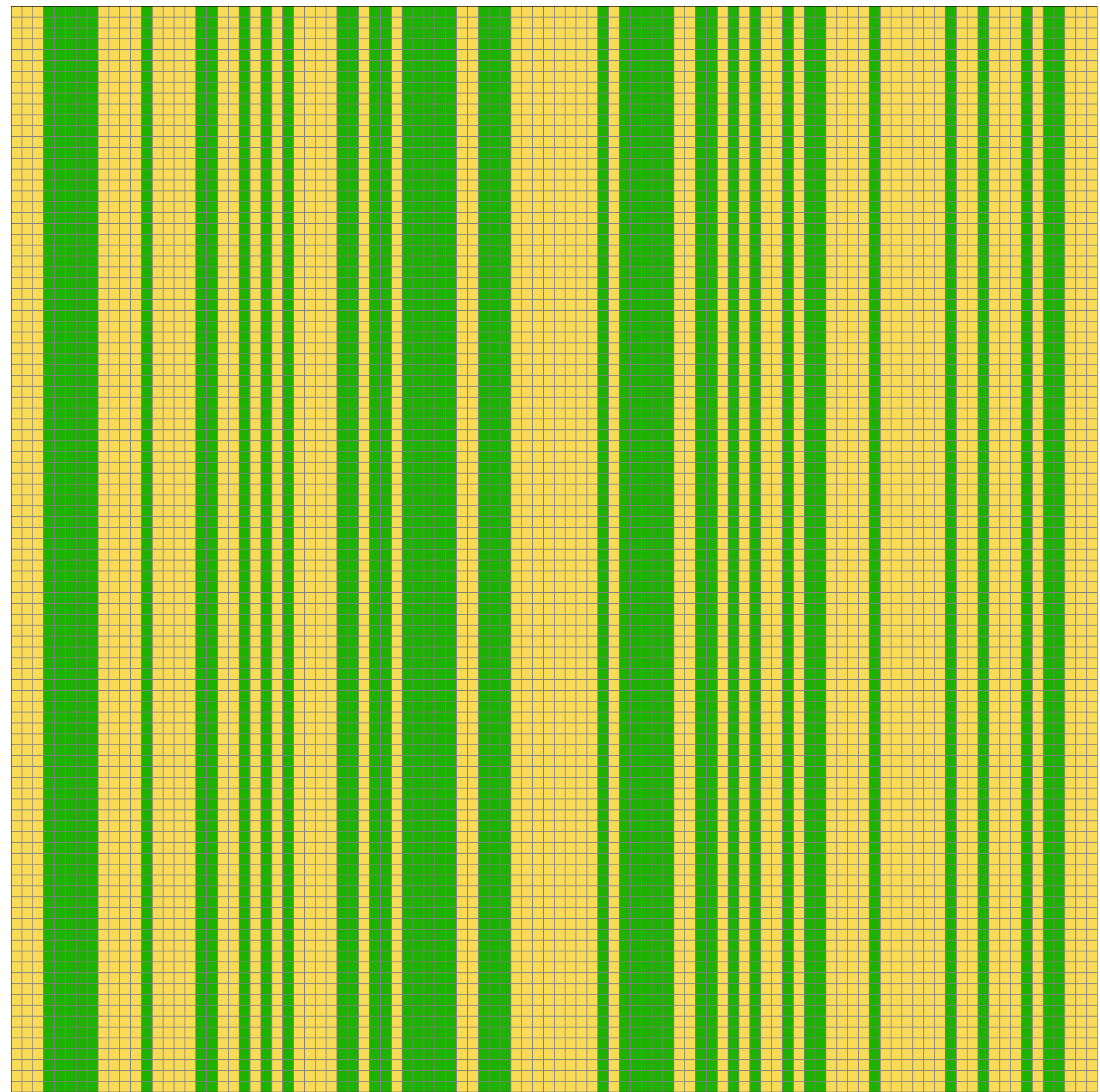
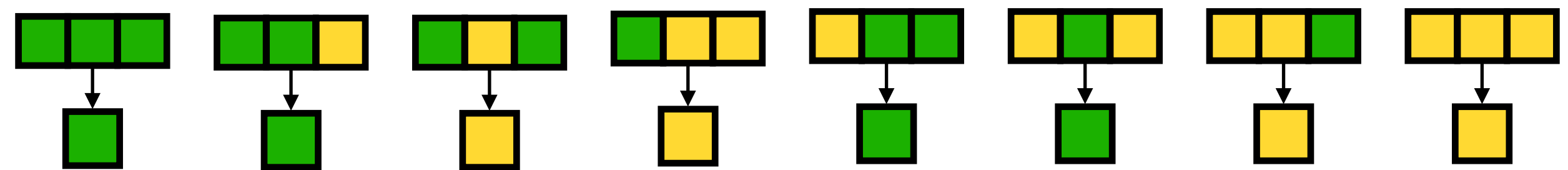
- Therefore a **local rule**  $f$  is equivalent to its LUT **vector**.

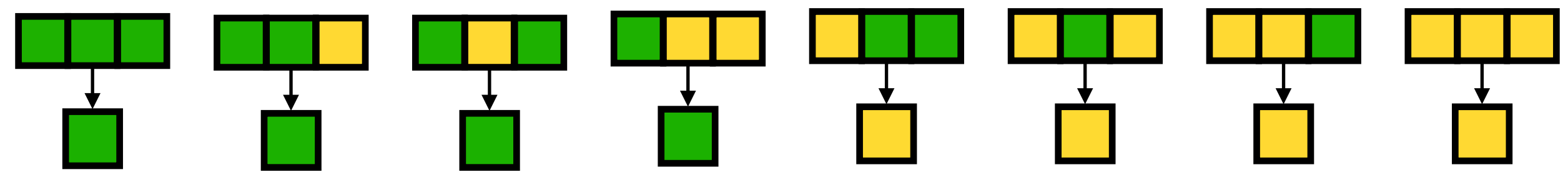


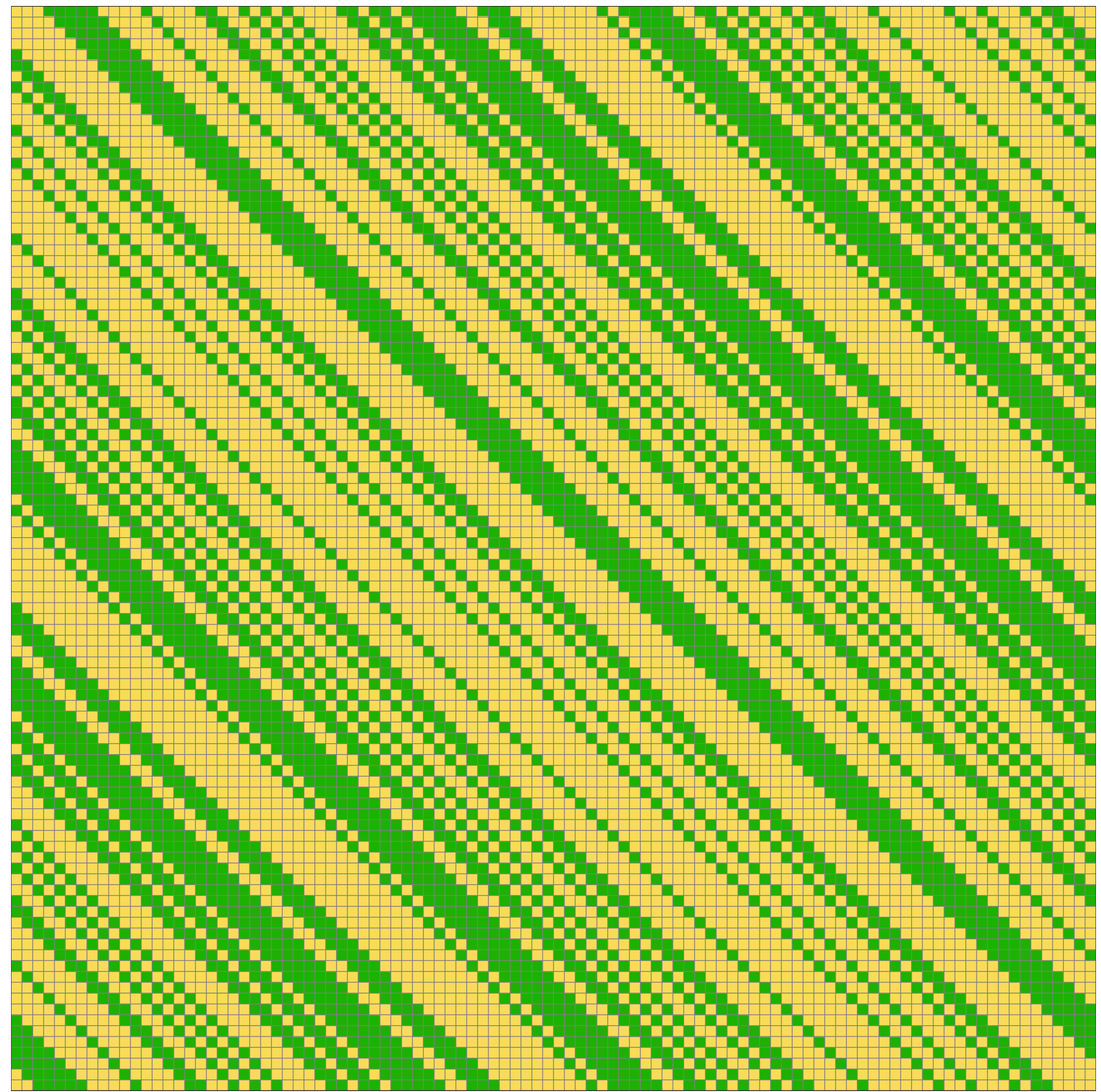
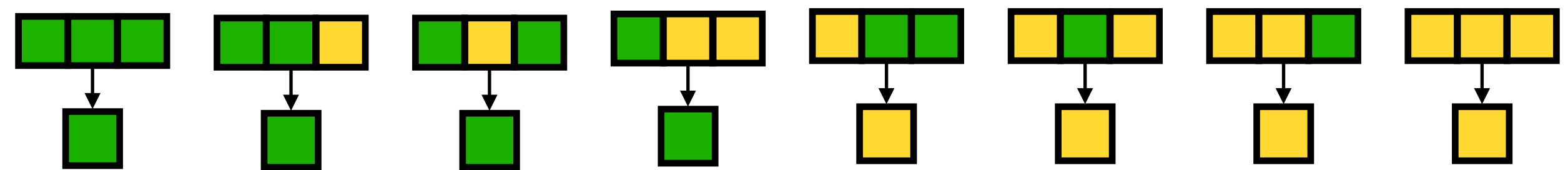


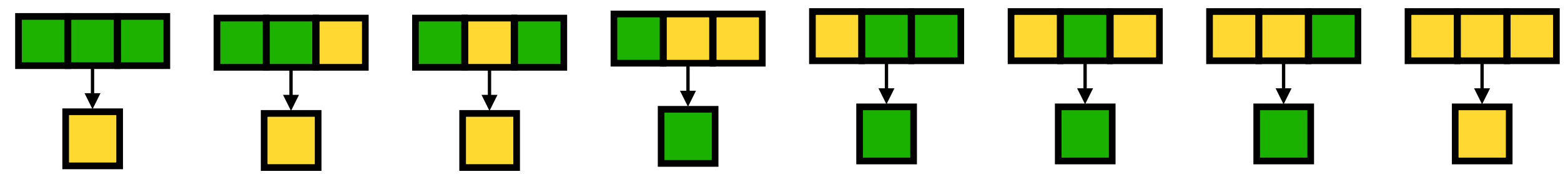




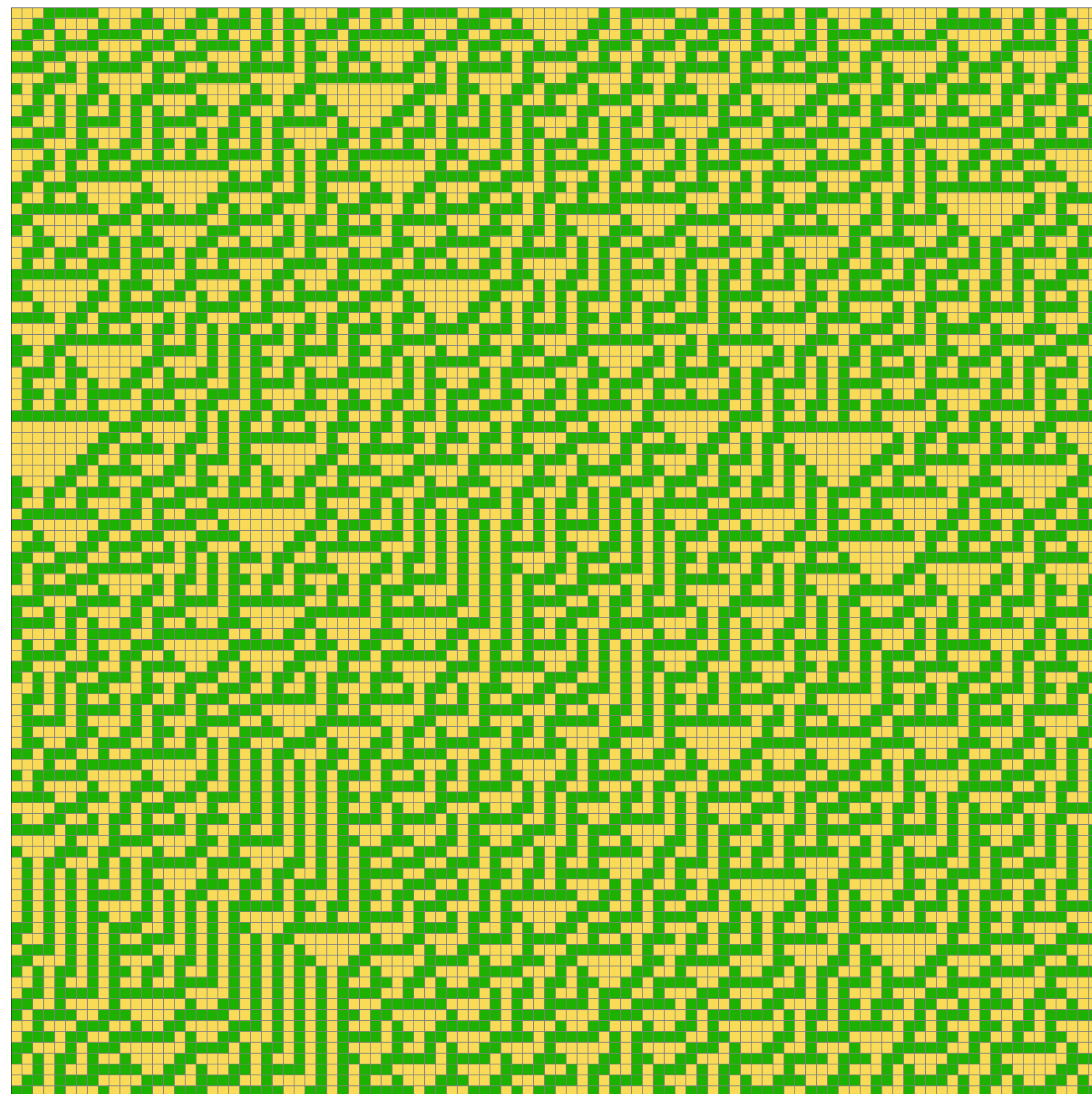
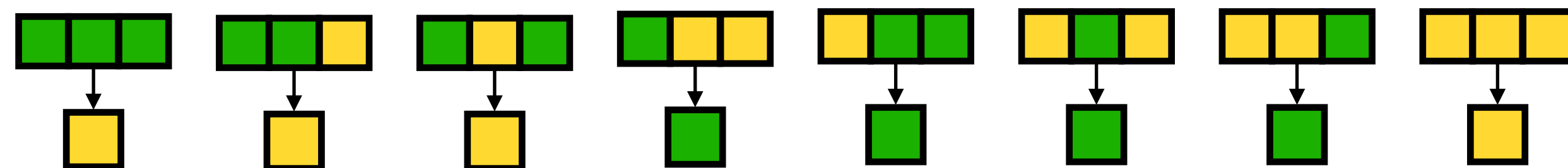


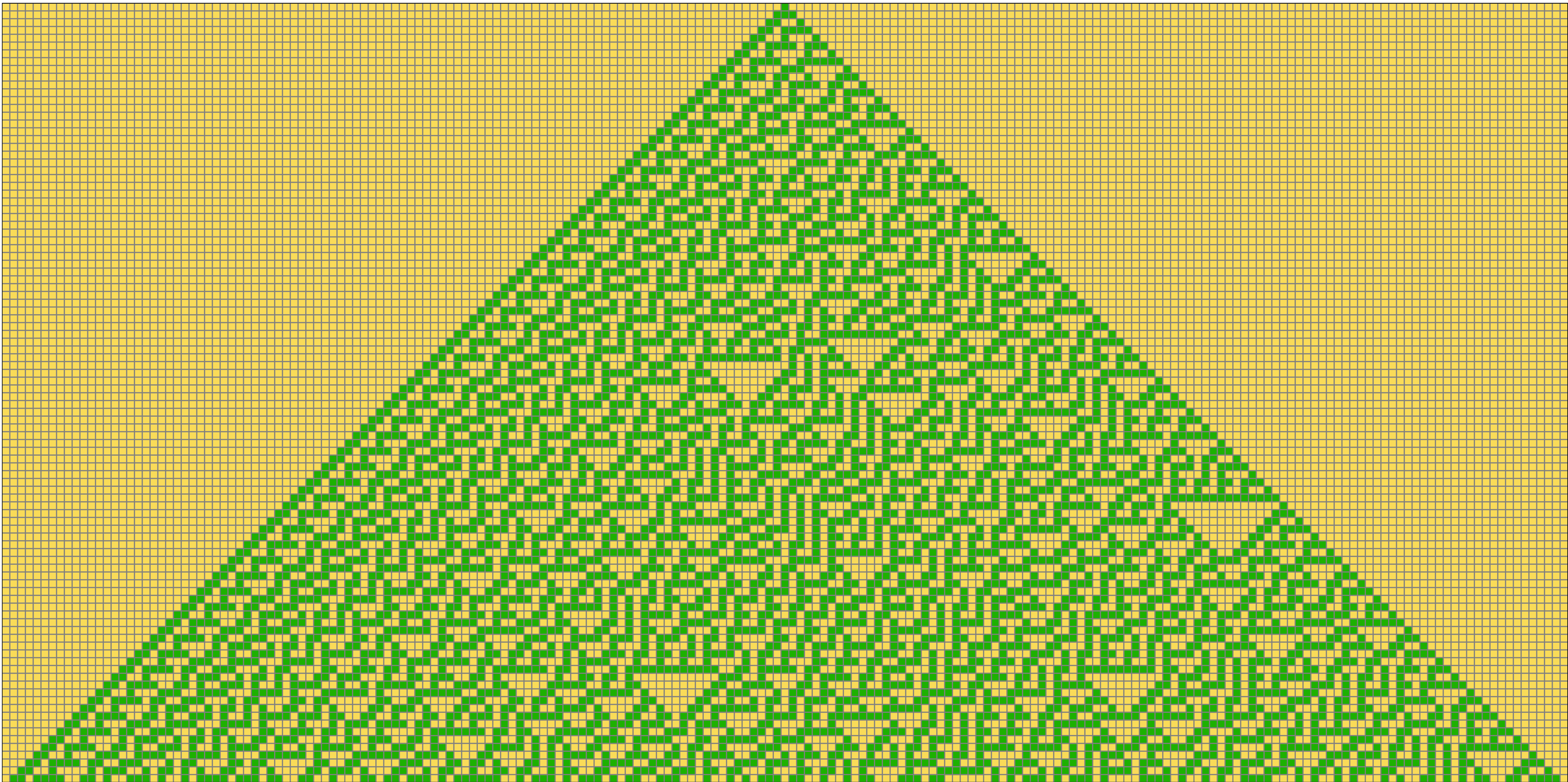




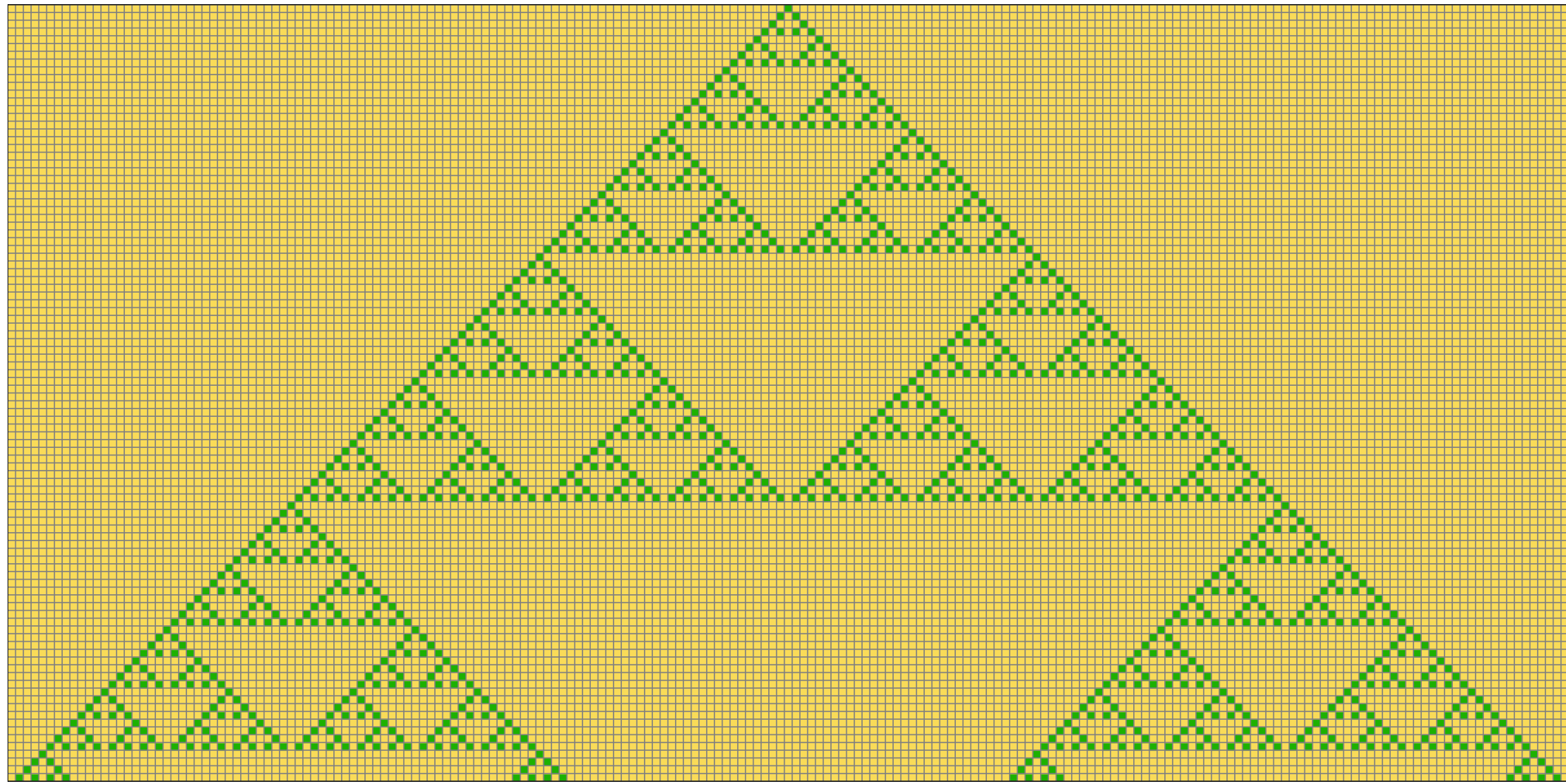




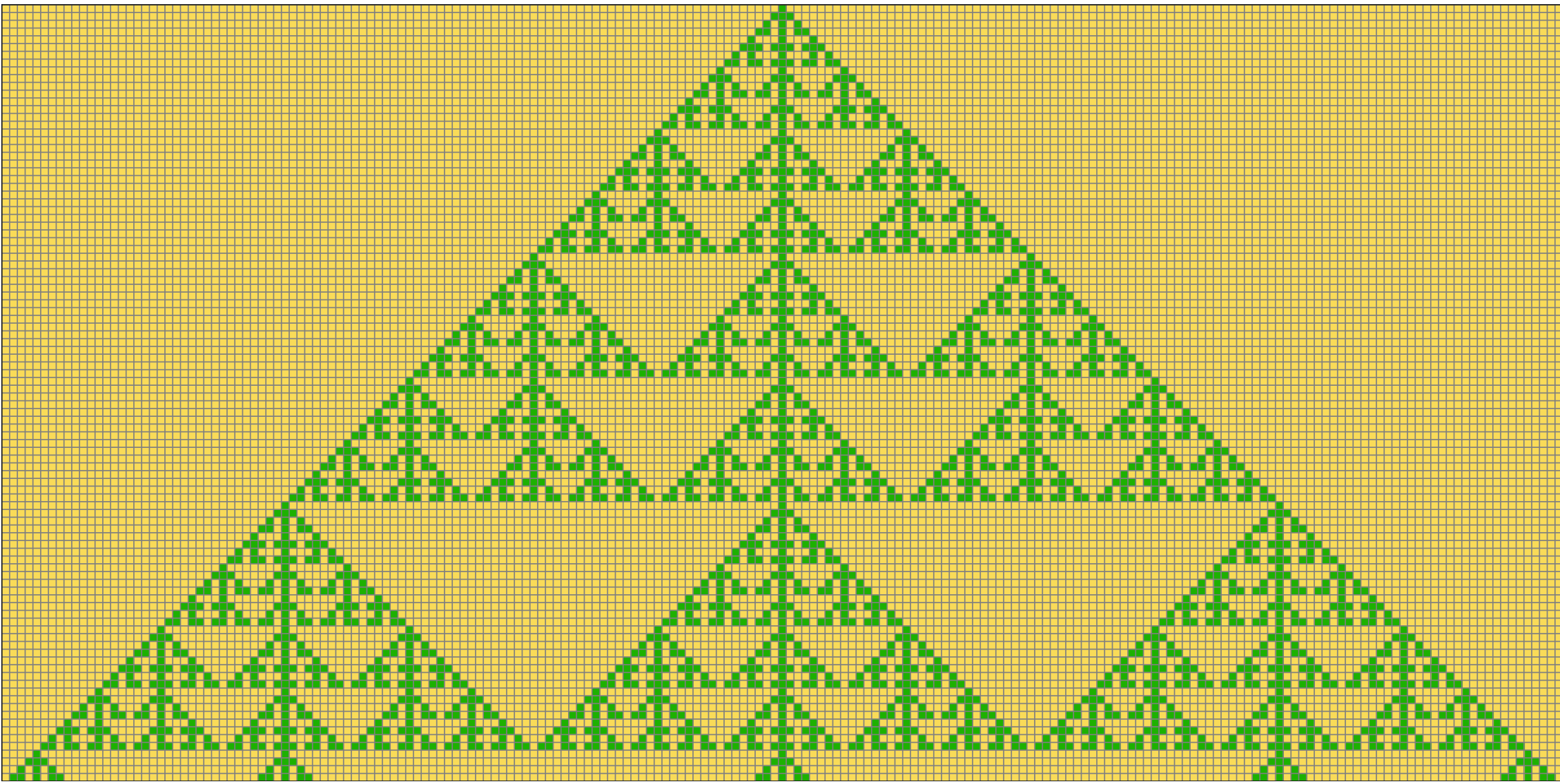




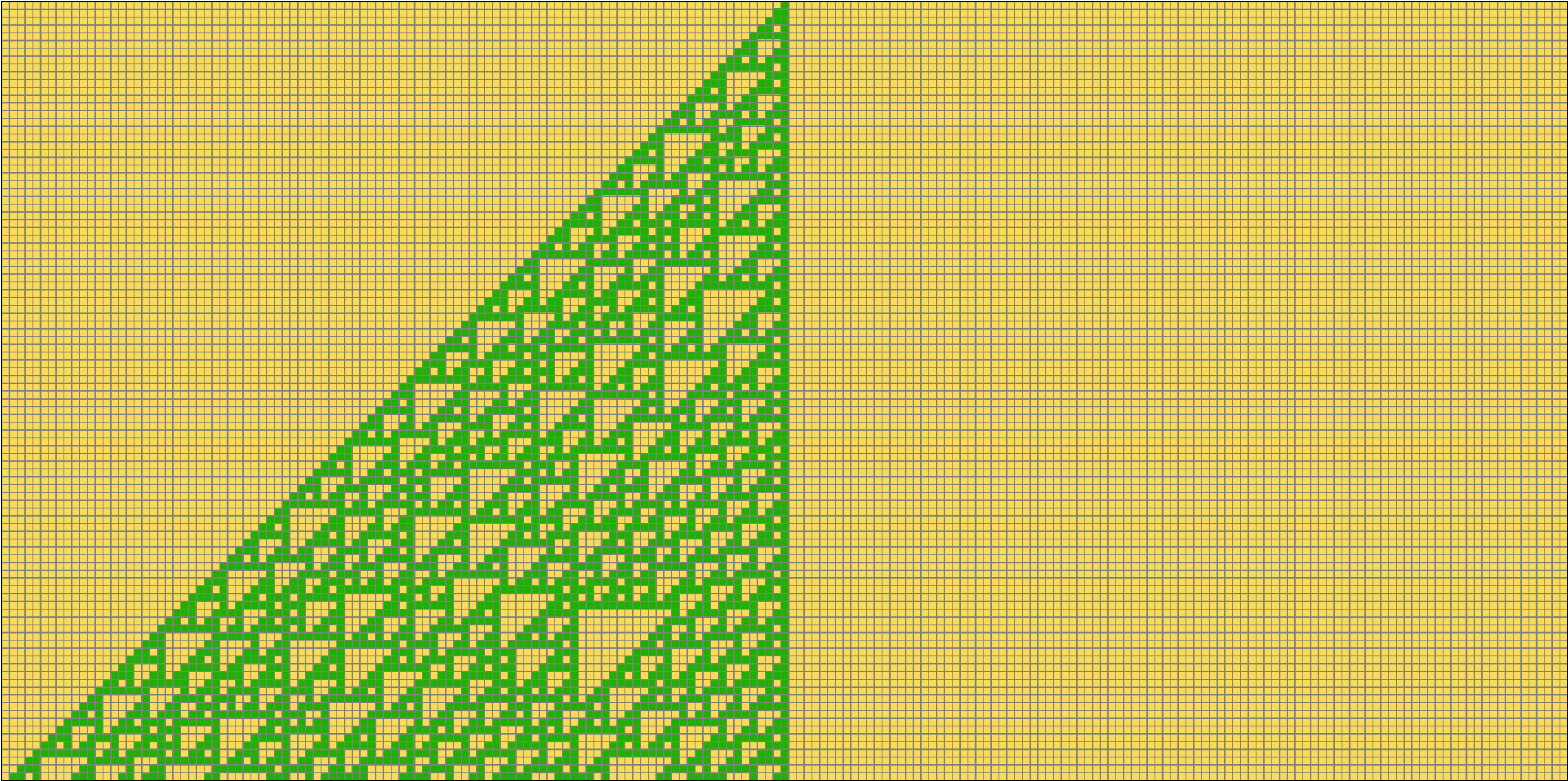
ECA 30



ECA 154



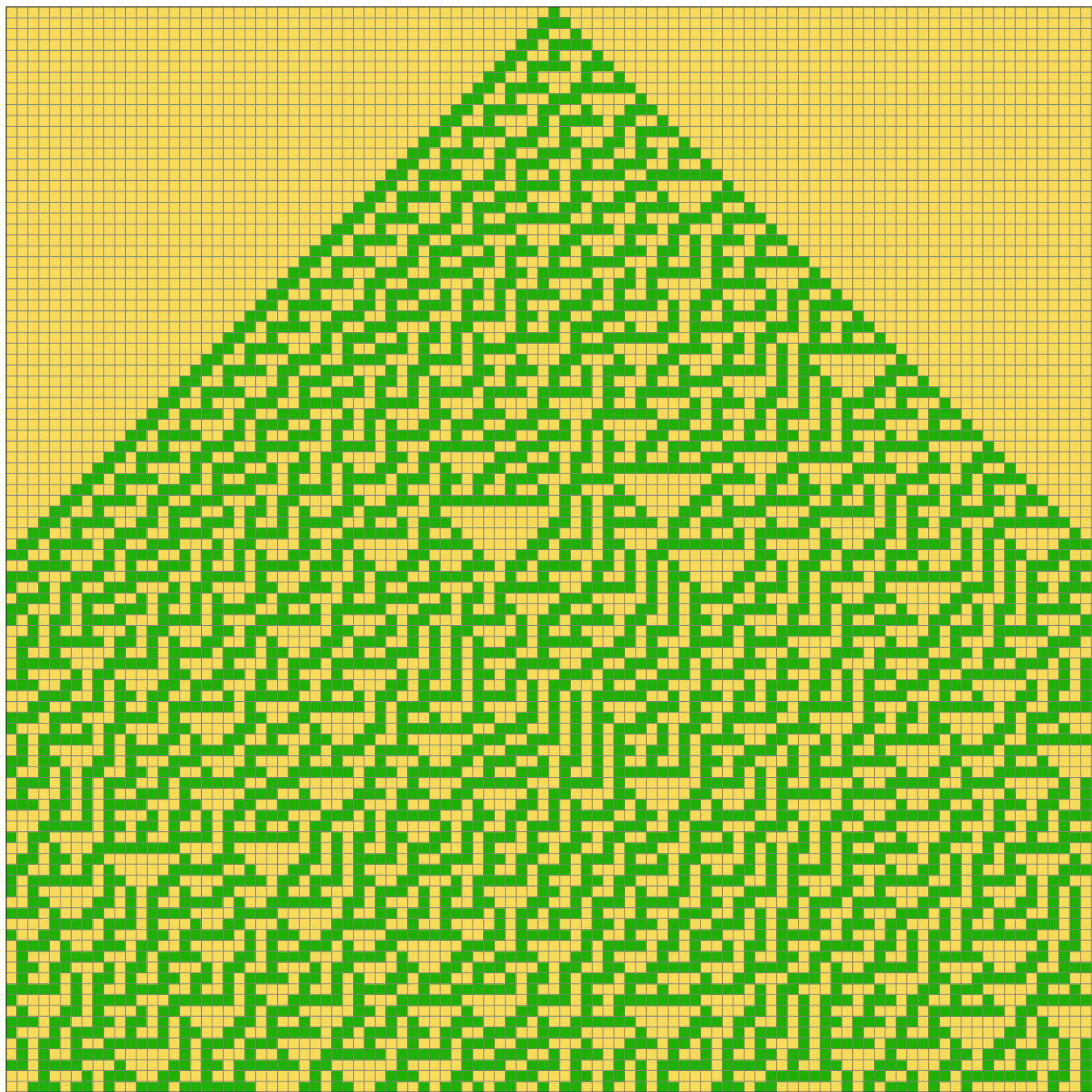
ECA 150



ECA 110

# Identification problem

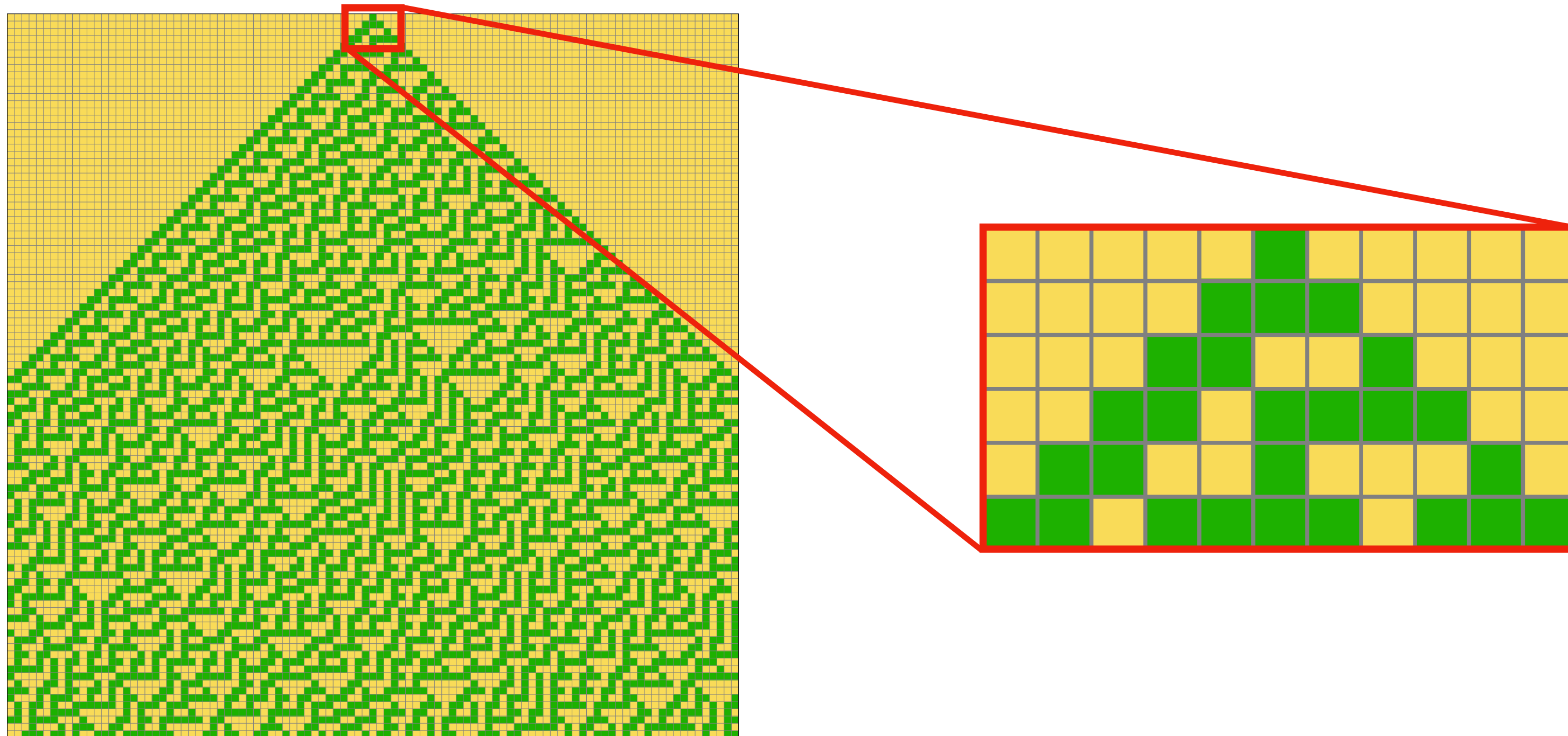
**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



**complete** space-time diagram

# Identification problem

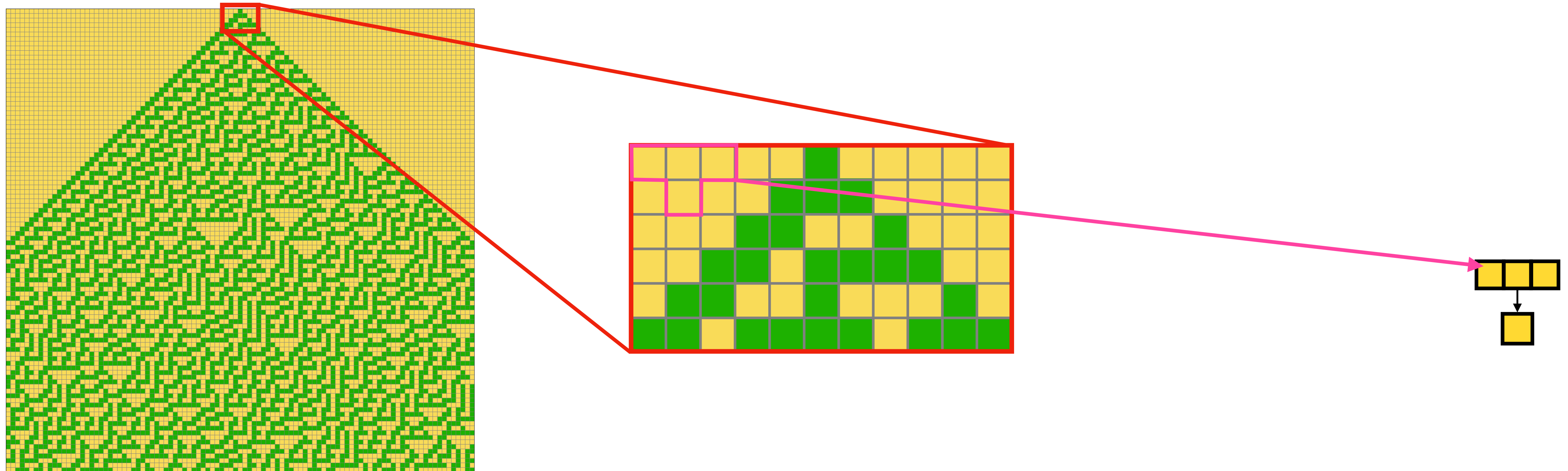
**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



**complete** space-time diagram

# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).

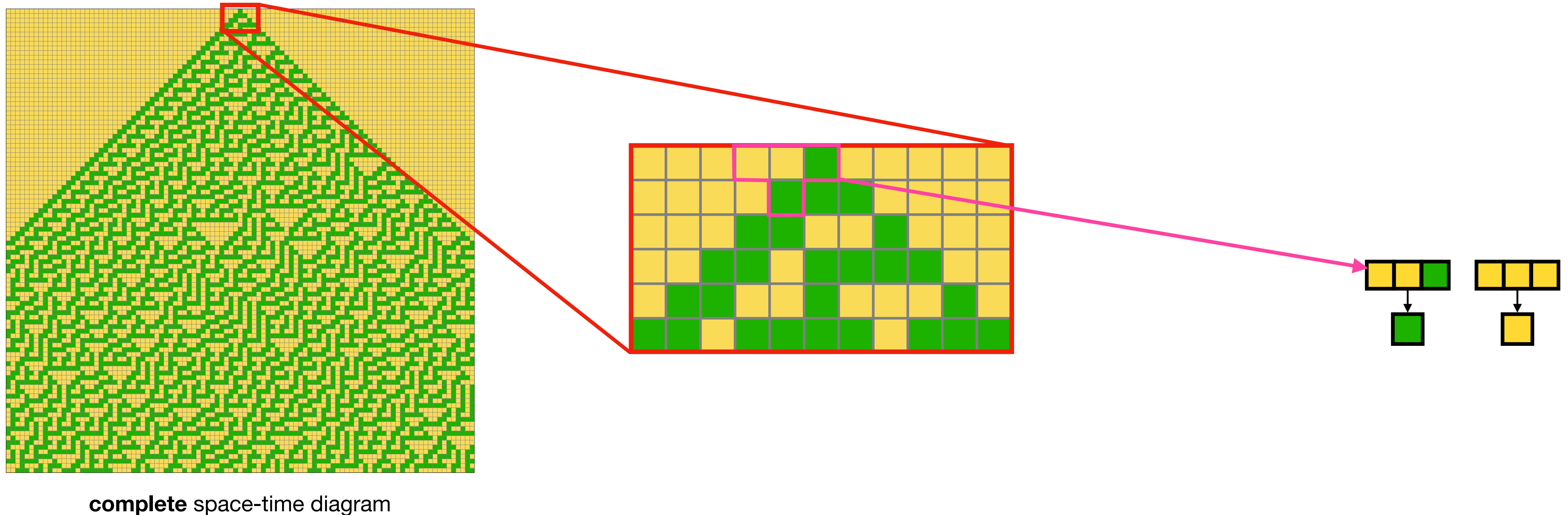


complete space-time diagram



# Identification problem

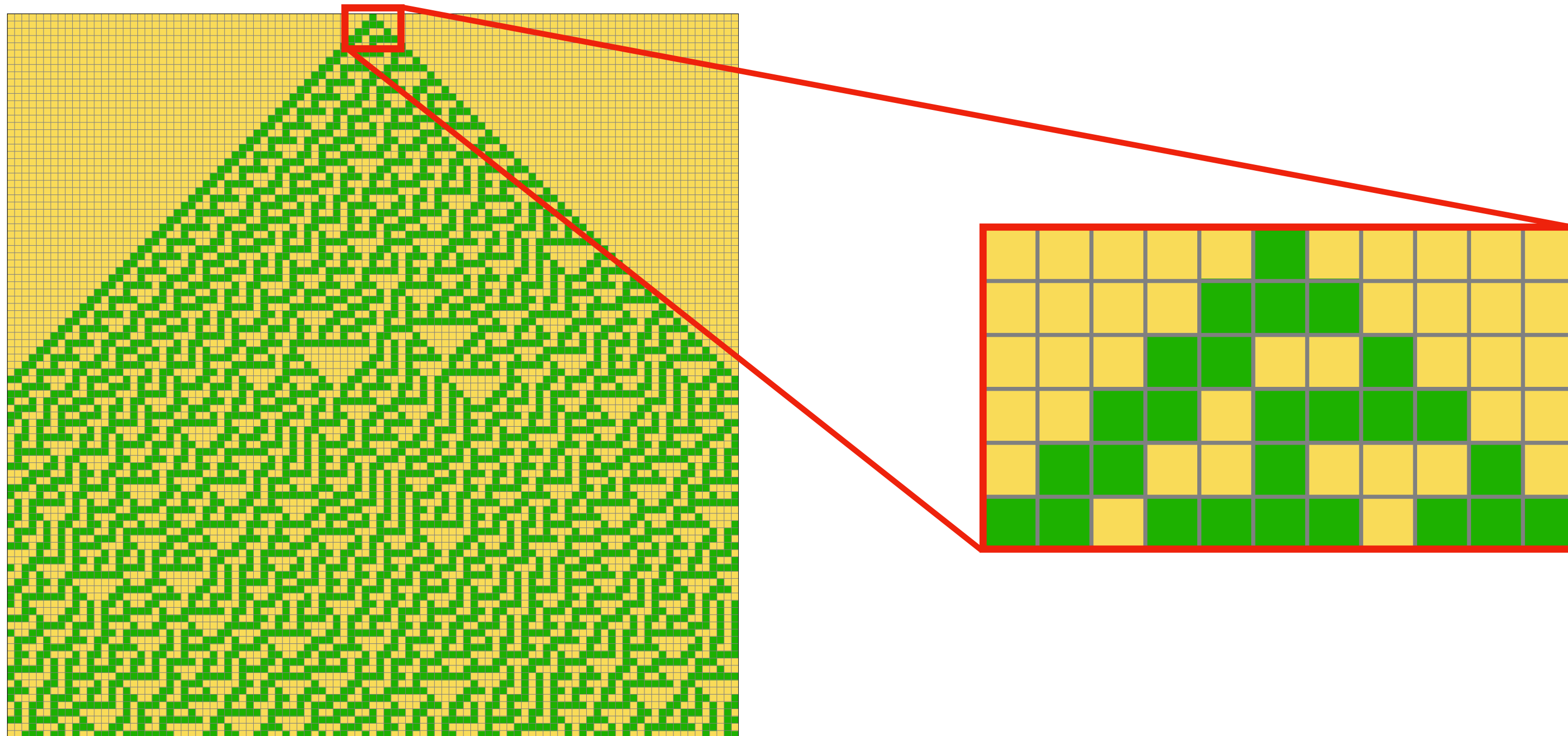
**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



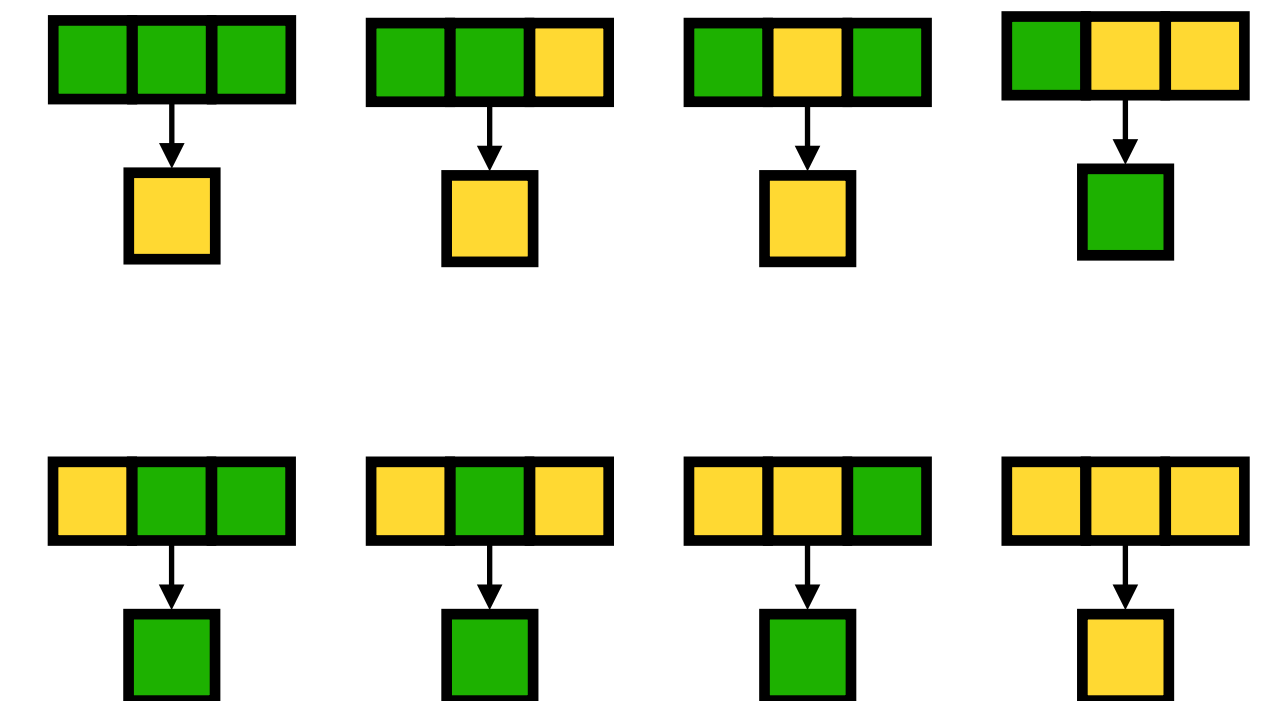


# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).

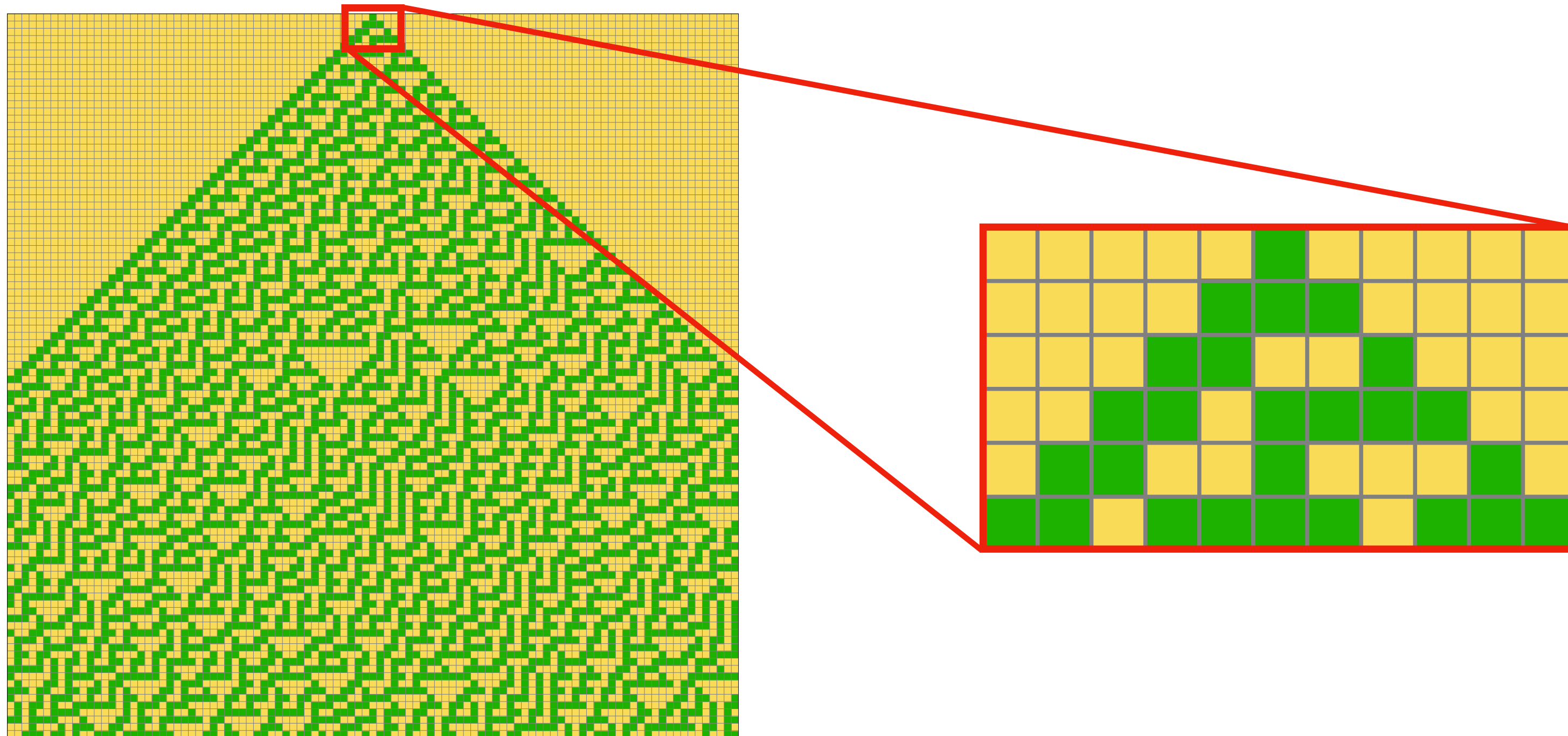


**complete** space-time diagram

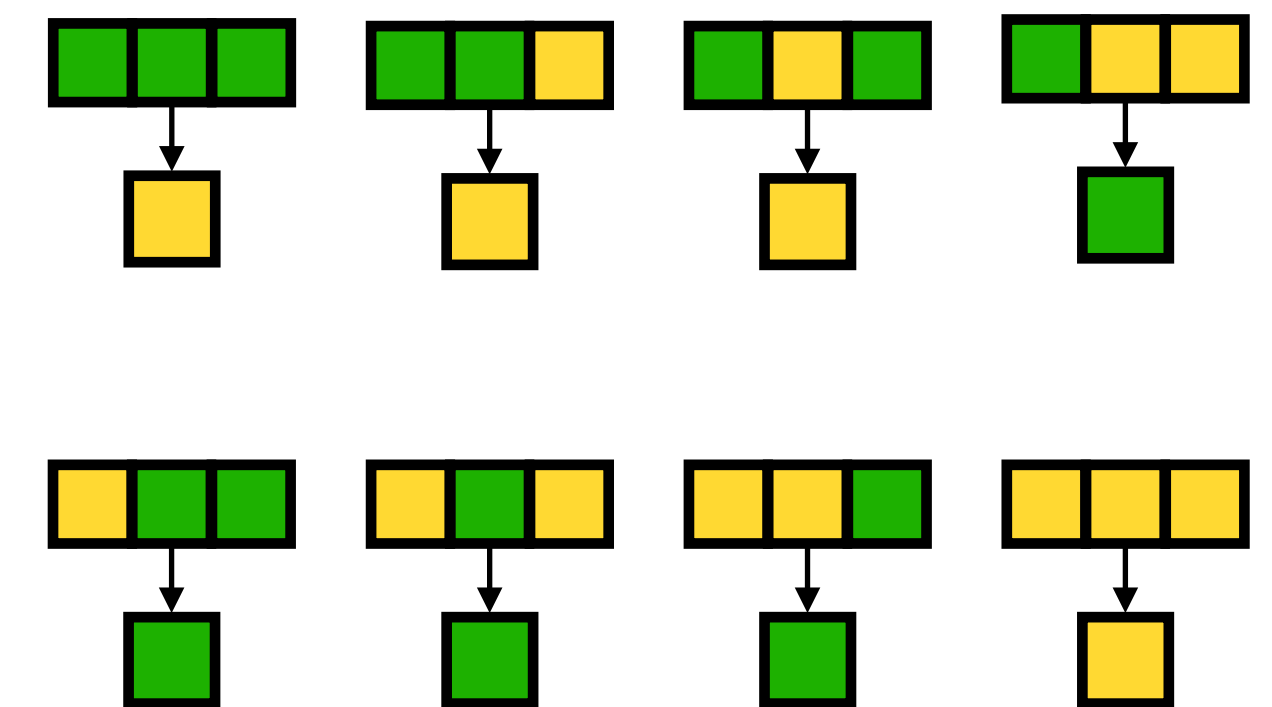


# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



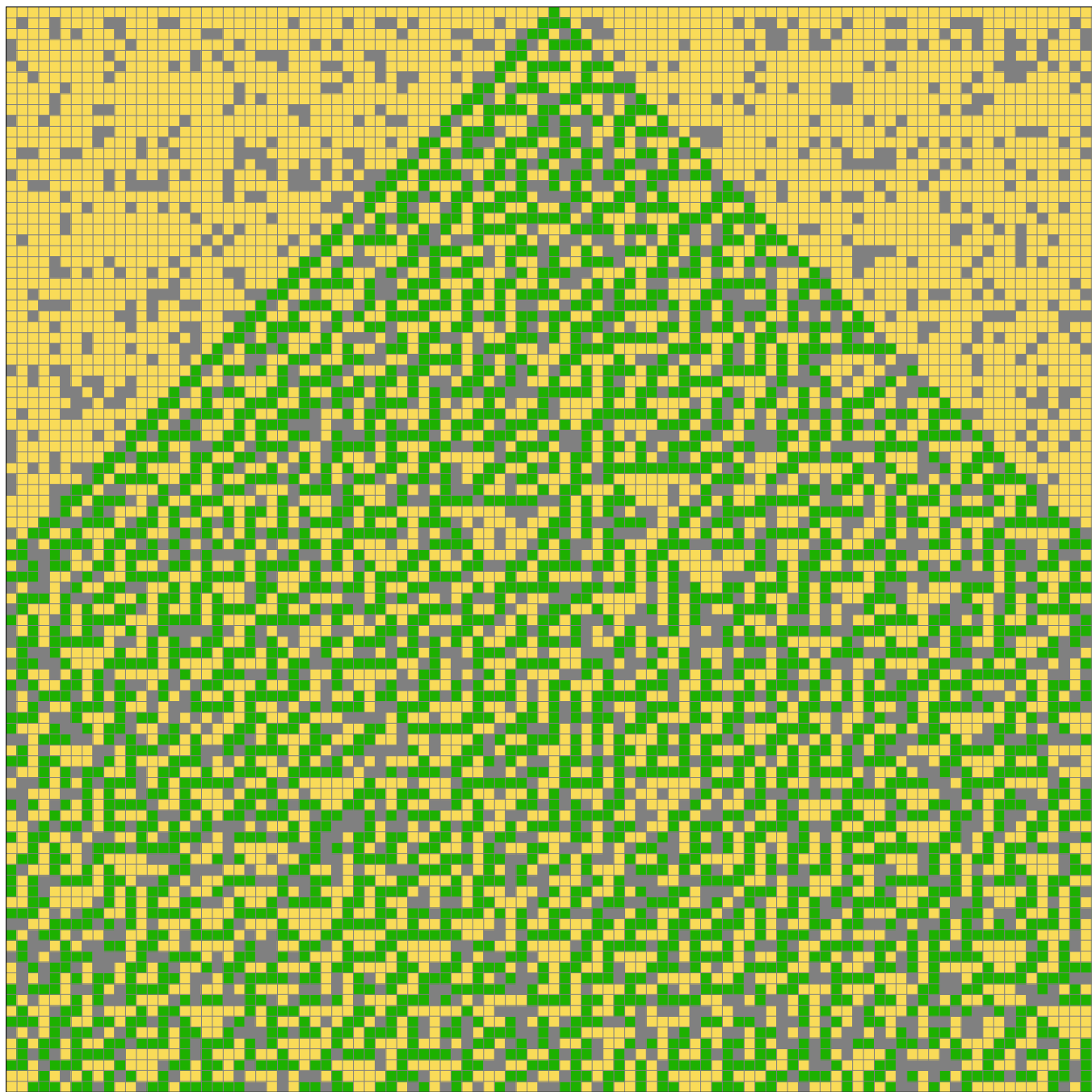
complete space-time diagram



We've found the **LUT**!

# Identification problem

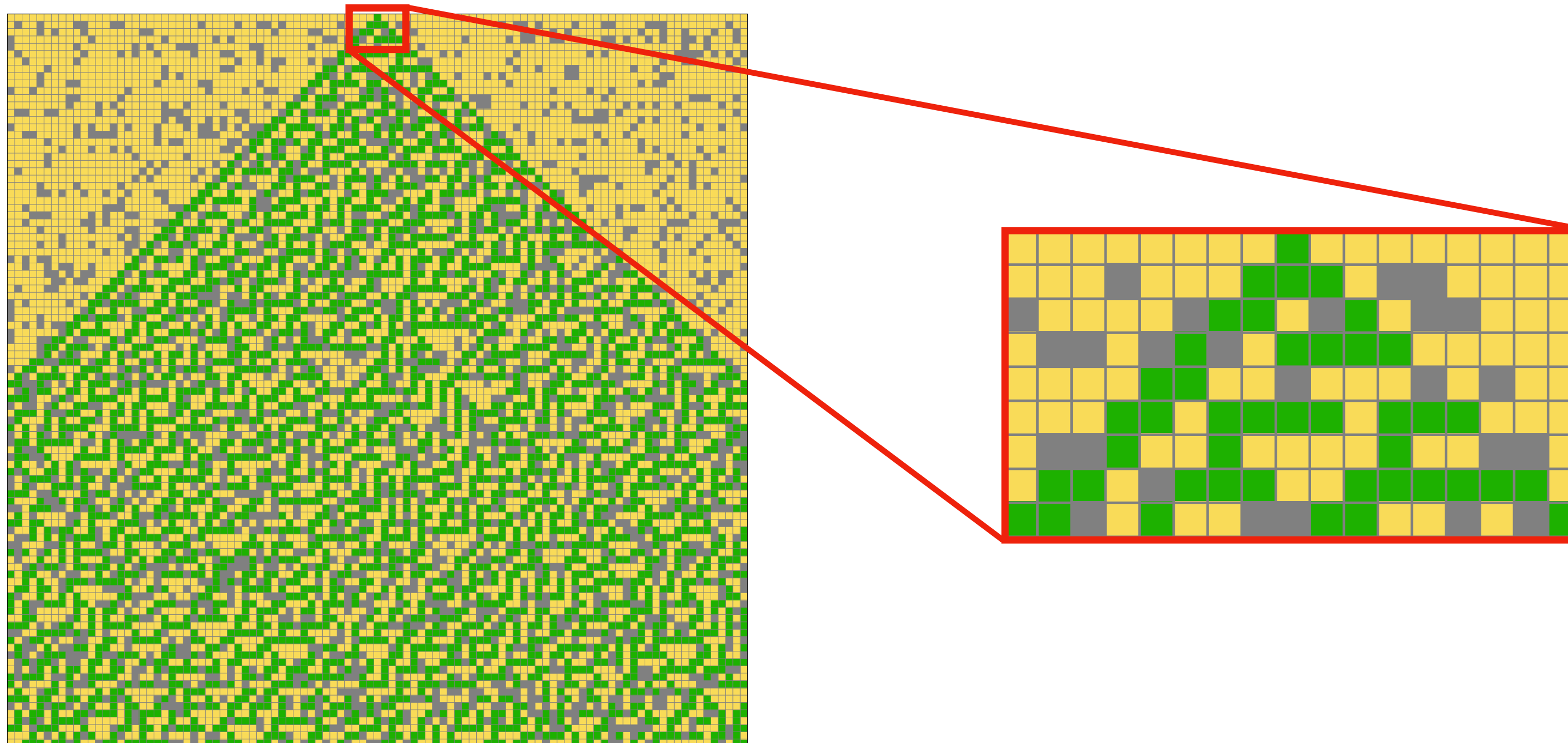
**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



**spatial partiality**  
missing cells are marked with grey

# Identification problem

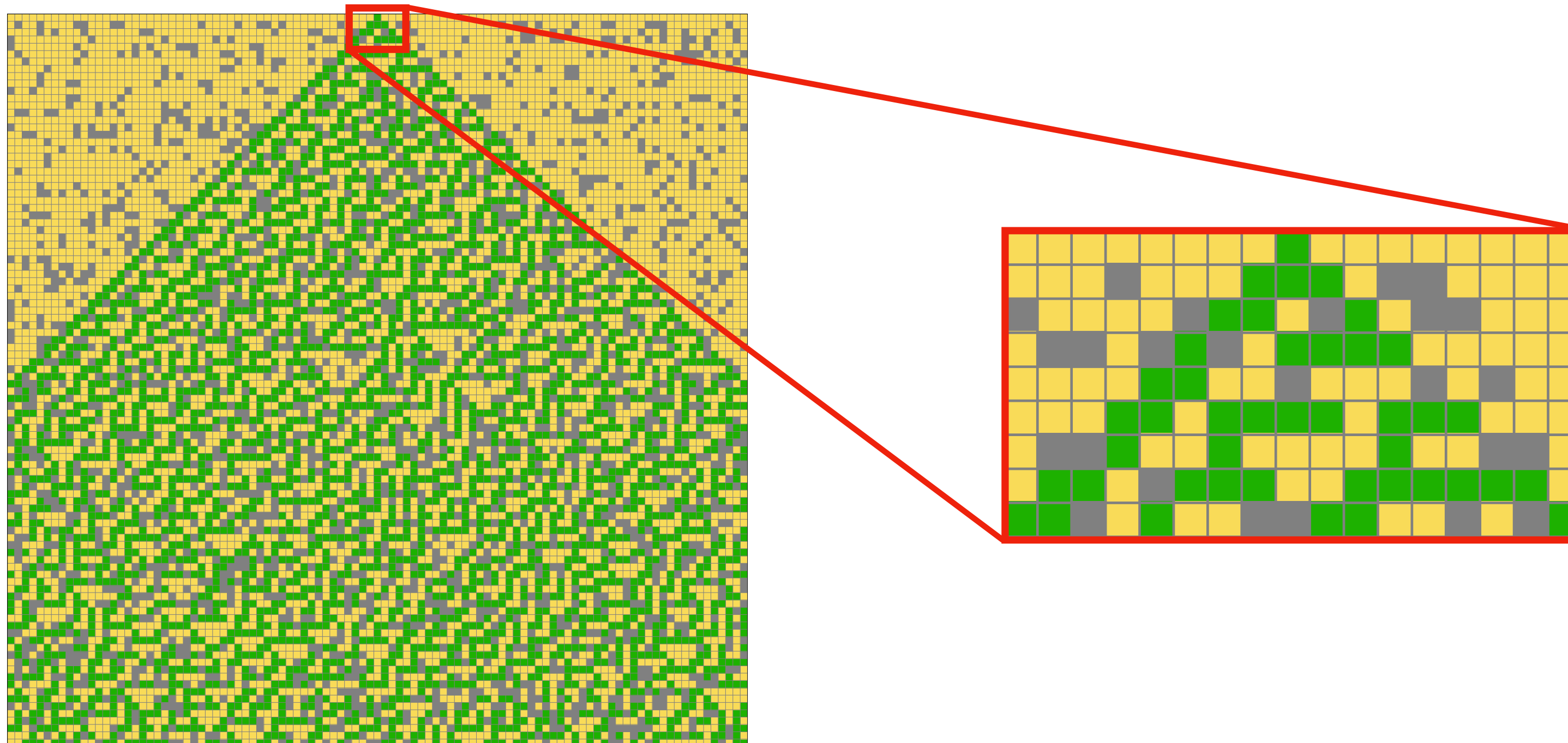
**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



**spatial partiality**  
missing cells are marked with grey

# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



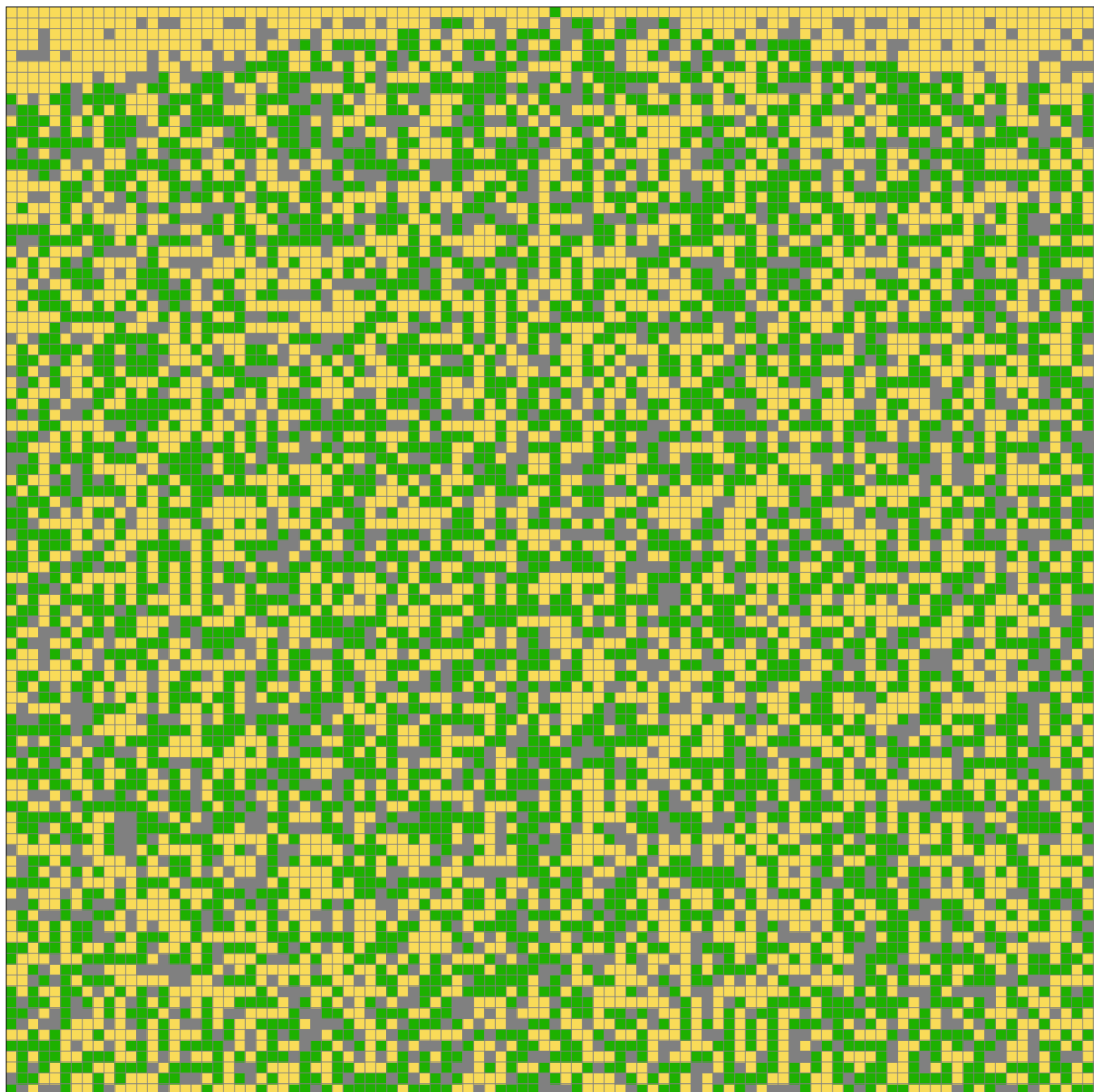
**spatial partiality**  
missing cells are marked with grey

Reading the transitions  
*may* still be possible...

But it is also a bit harder.

# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).

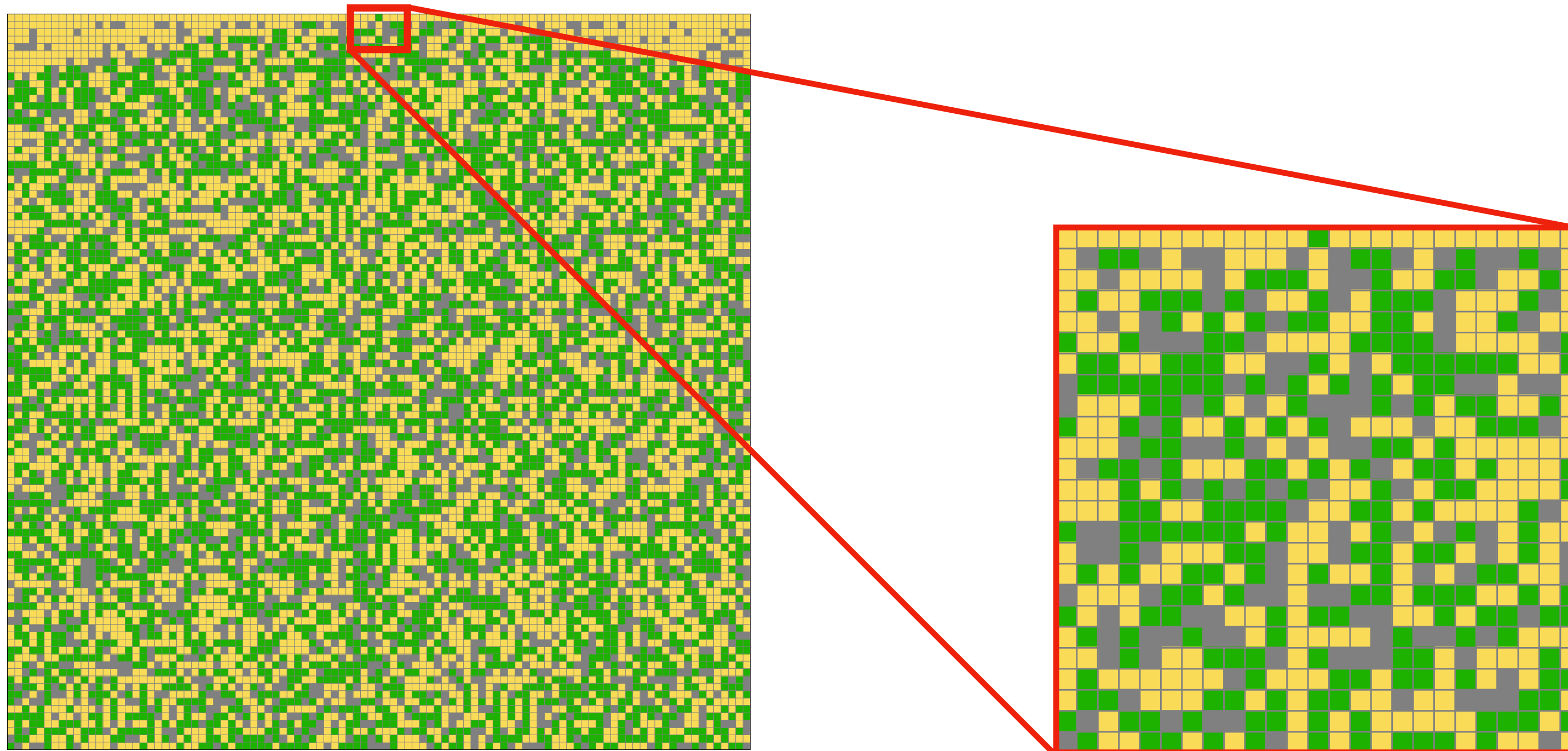


**spatial and temporal partiality**  
missing cells are marked with grey



# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).

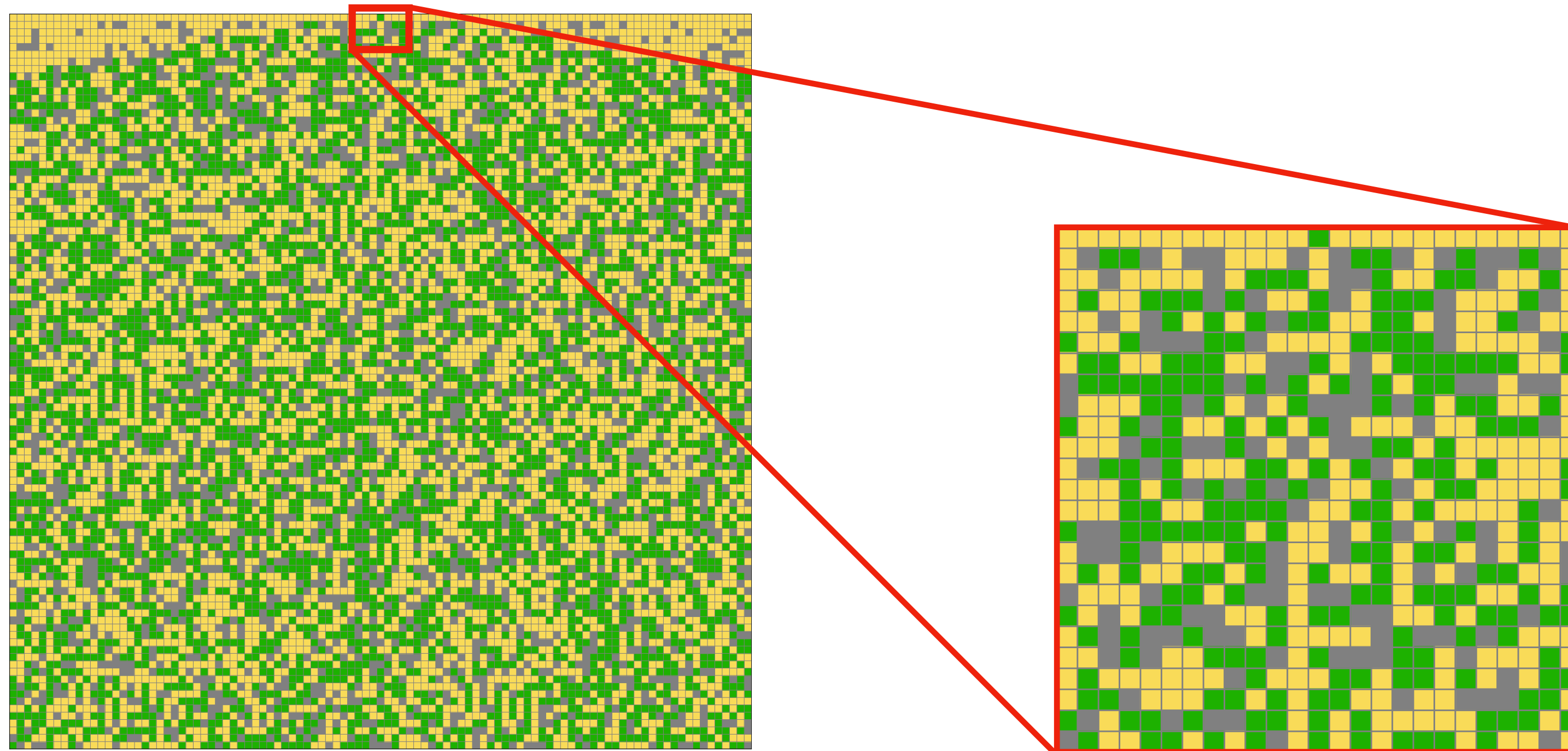


spatial and temporal partiality  
missing cells are marked with grey



# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



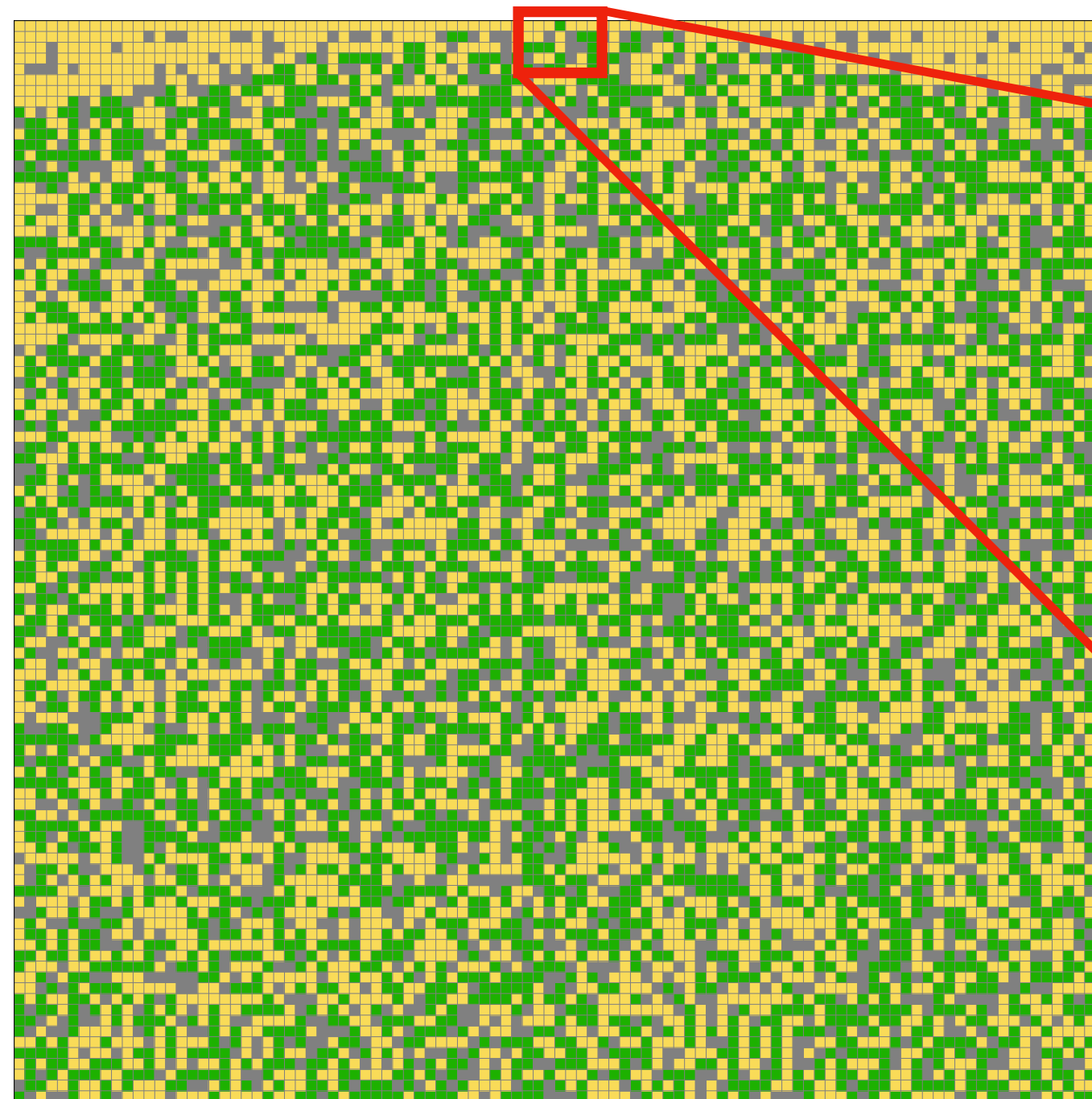
**spatial and temporal partiality**  
missing cells are marked with grey

Reading the transitions directly is **not** possible...

If we try to do it, we will get conflicting results!

# Identification problem

**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



**spatial and temporal partiality**  
missing cells are marked with grey

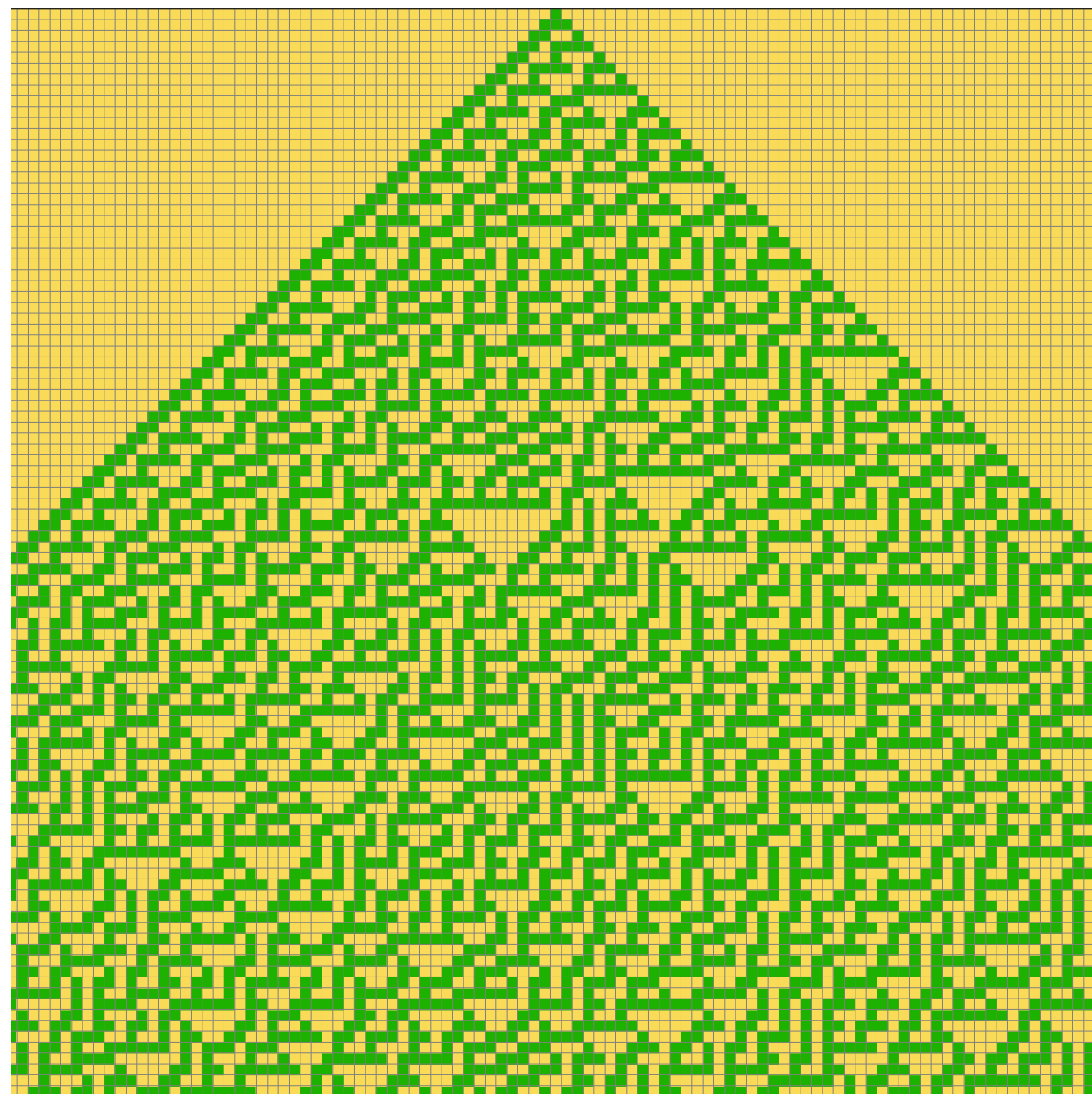
Reading the transitions directly is **not** possible...

If we try to do it, we will get conflicting results!

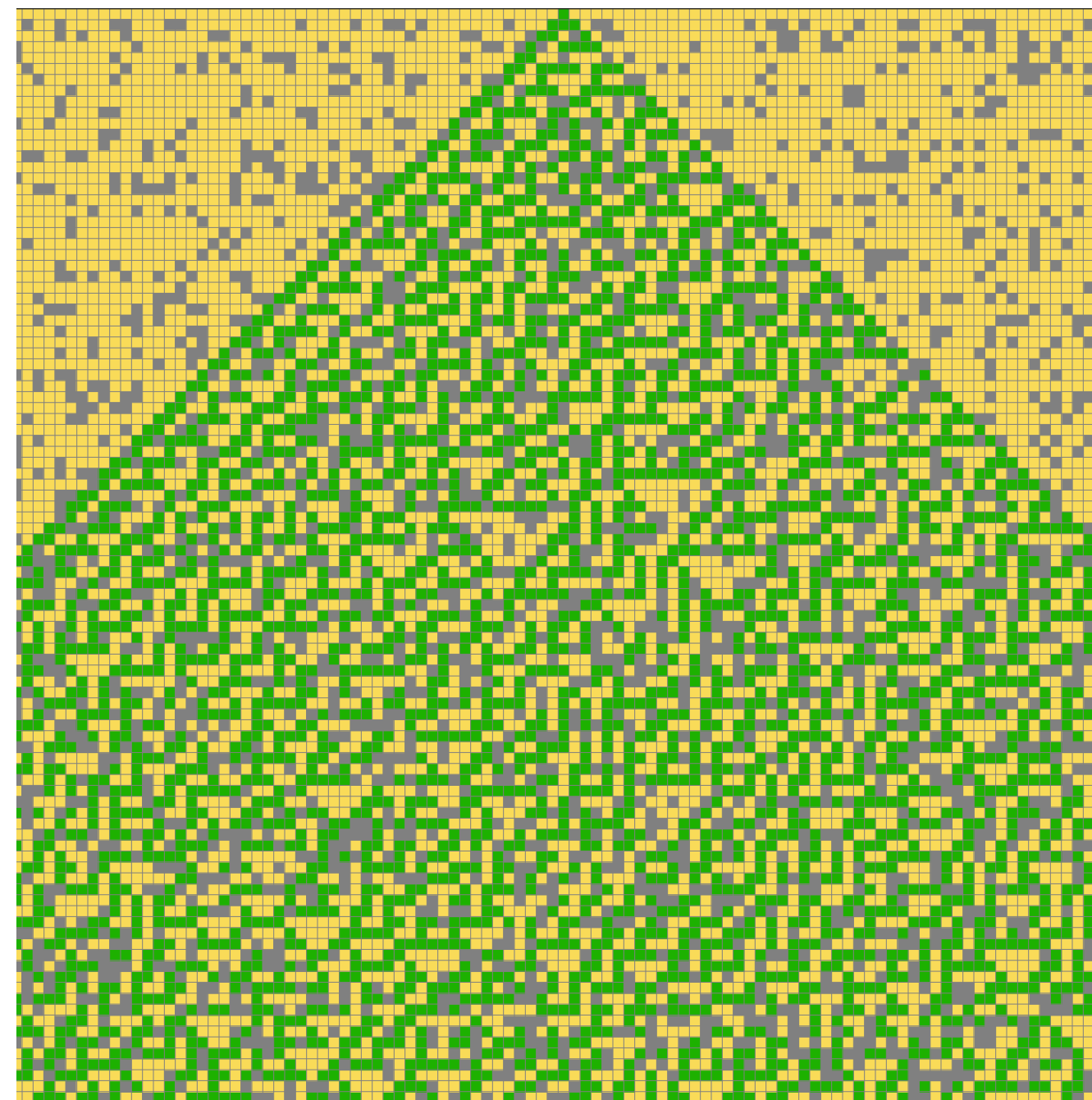


# Identification problem

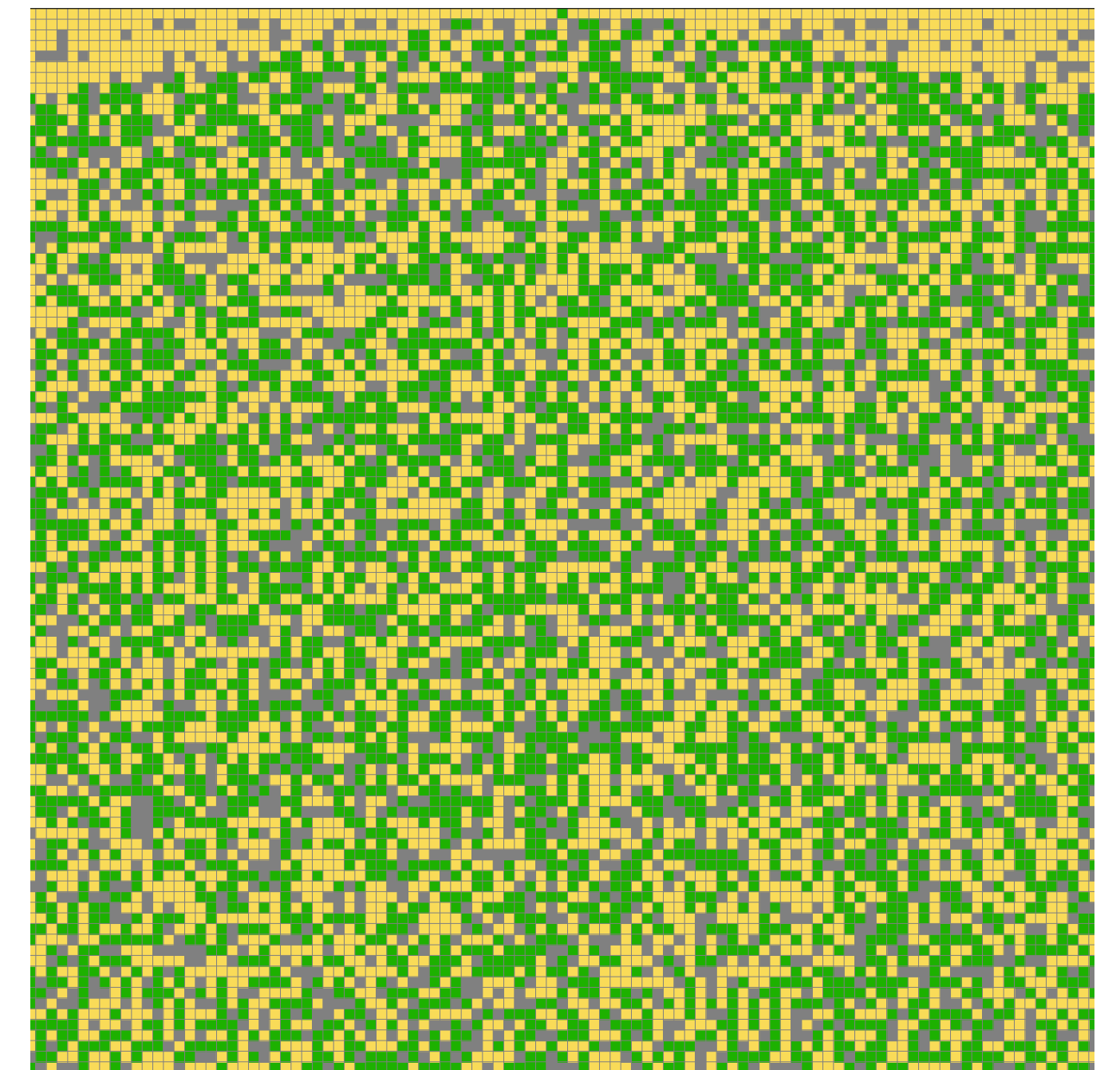
**Identification problem:** given “some **parts**” of space-time diagrams (partial information), generated by an unknown CA, reconstruct the **local rule** of this CA (and complete the space-time diagrams).



**complete** space-time diagram  
no missing information



**spatial partiality**  
missing states (gray squares)



**spatial & temporal partiality**  
missing states (gray squares) and time-steps

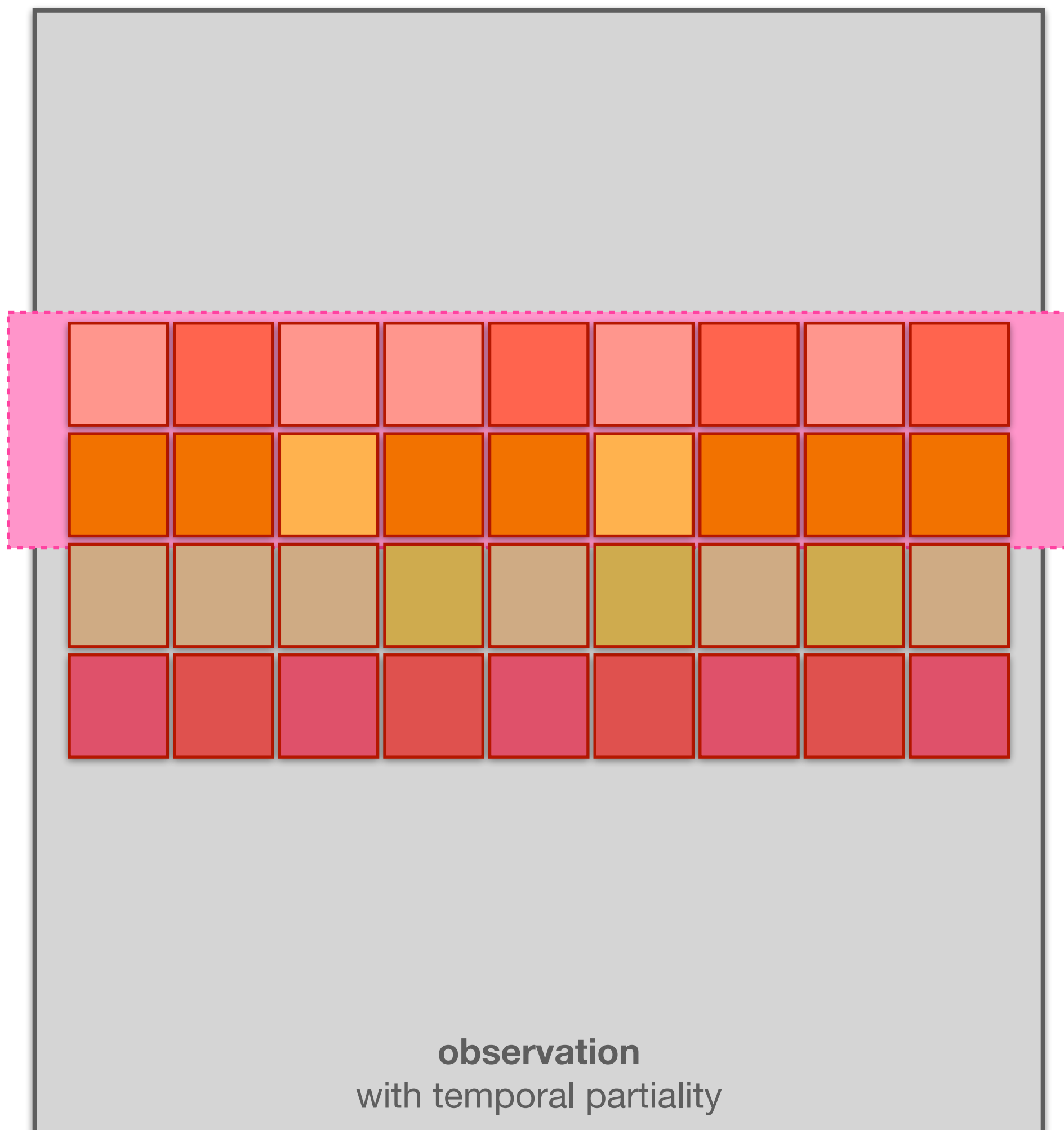
# Identification problem - assumptions

- State set is **known** (binary), number of cells is **known**.
- Periodic boundary conditions are **assumed**.
- Initial configuration has been **completely** observed (states of **all** the cells are known in the initial configuration).
- Neighborhood radius is **unknown**, yet an **upper bound**  $r_{\max}$  is known.
- Time gap lengths are **unknown**, yet an **upper bound**  $T_{\max}$  is known.
- Local rule is **unknown**.

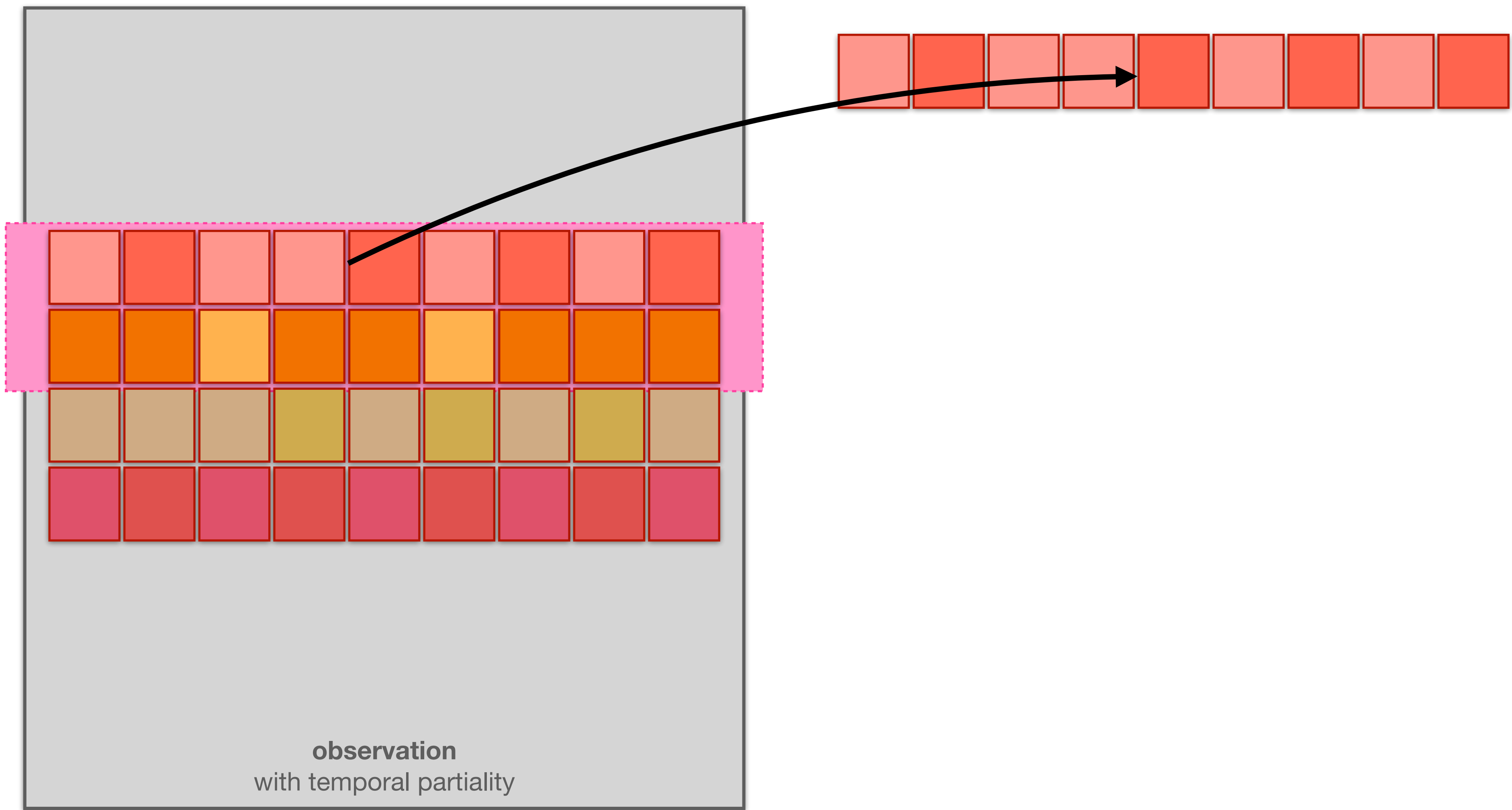
# Solution strategy: Genetic Algorithm

- **Search space:** All LUTs of the local rules with  $r = 1, 2, \dots, r_{\max}$  (for a given  $r$  **size** is  $2^{2^{r+1}}$ )
- **Population:** LUTs of candidate local rules, with different radii (different lengths)
- Genetic operators:
  - **Selection:** random with probability proportional to fitness values
  - **Mutation:** random changes in the LUT
  - **Up-scale / Down-scale:** changing the radius of the LUT
  - **Cross-over:** agreement of the radii + uniform cross-over
- **Fitness function:** proportional to “pair-wise reproduction error”, which corresponds to how well a given candidate CA *reproduces* transitions between consecutive **pairs of rows** in the observation

# Fitness function

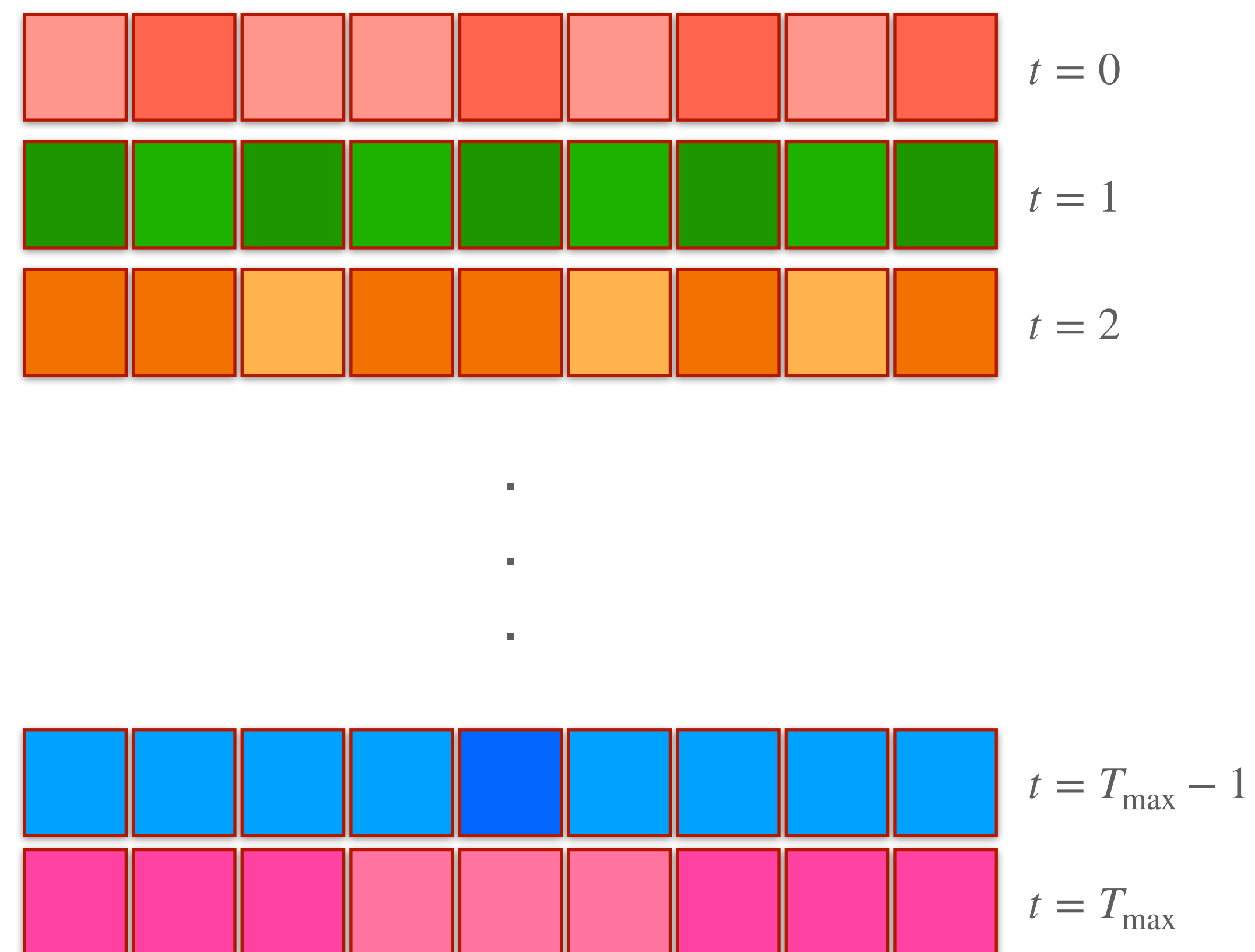
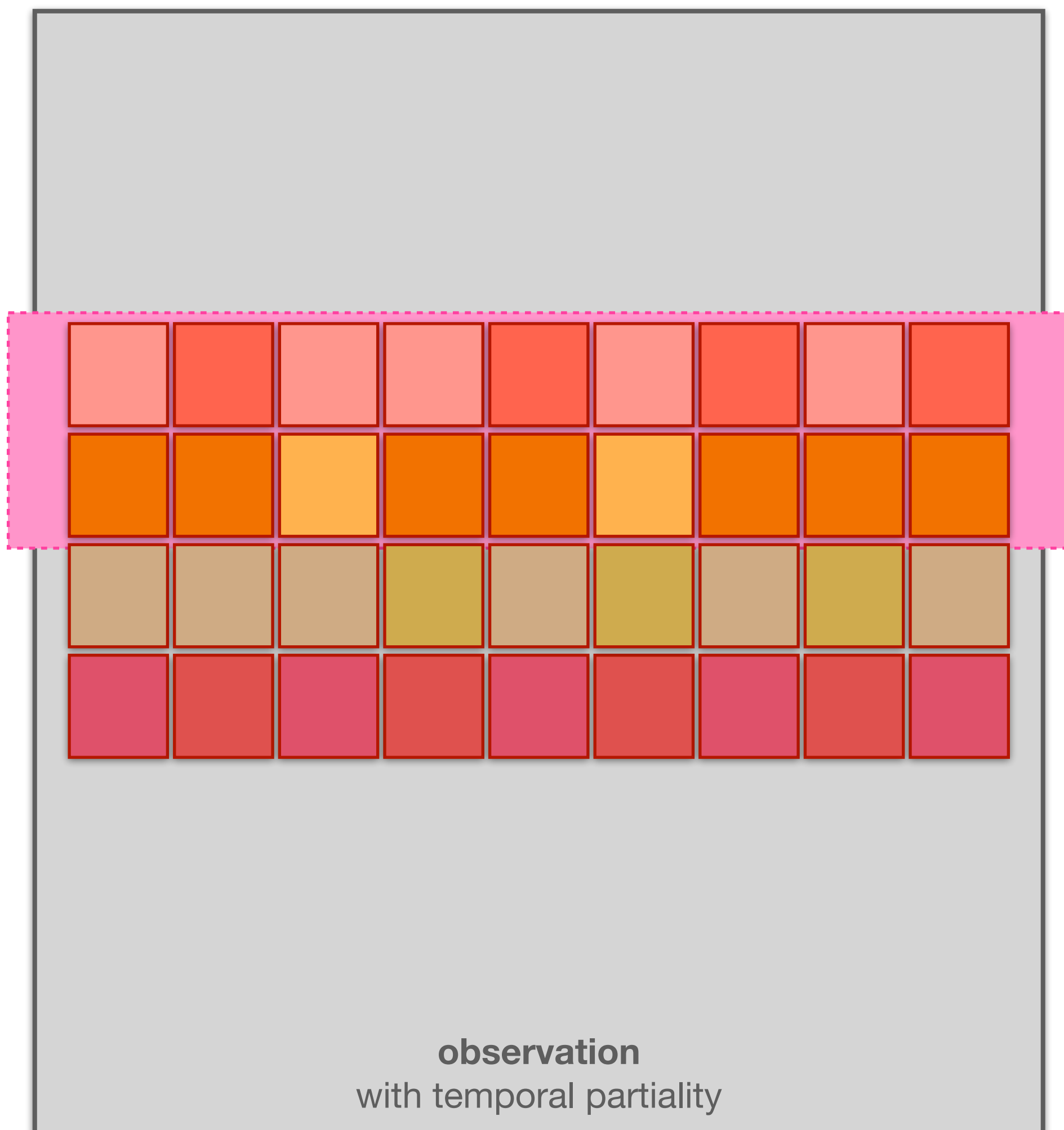


# Fitness function

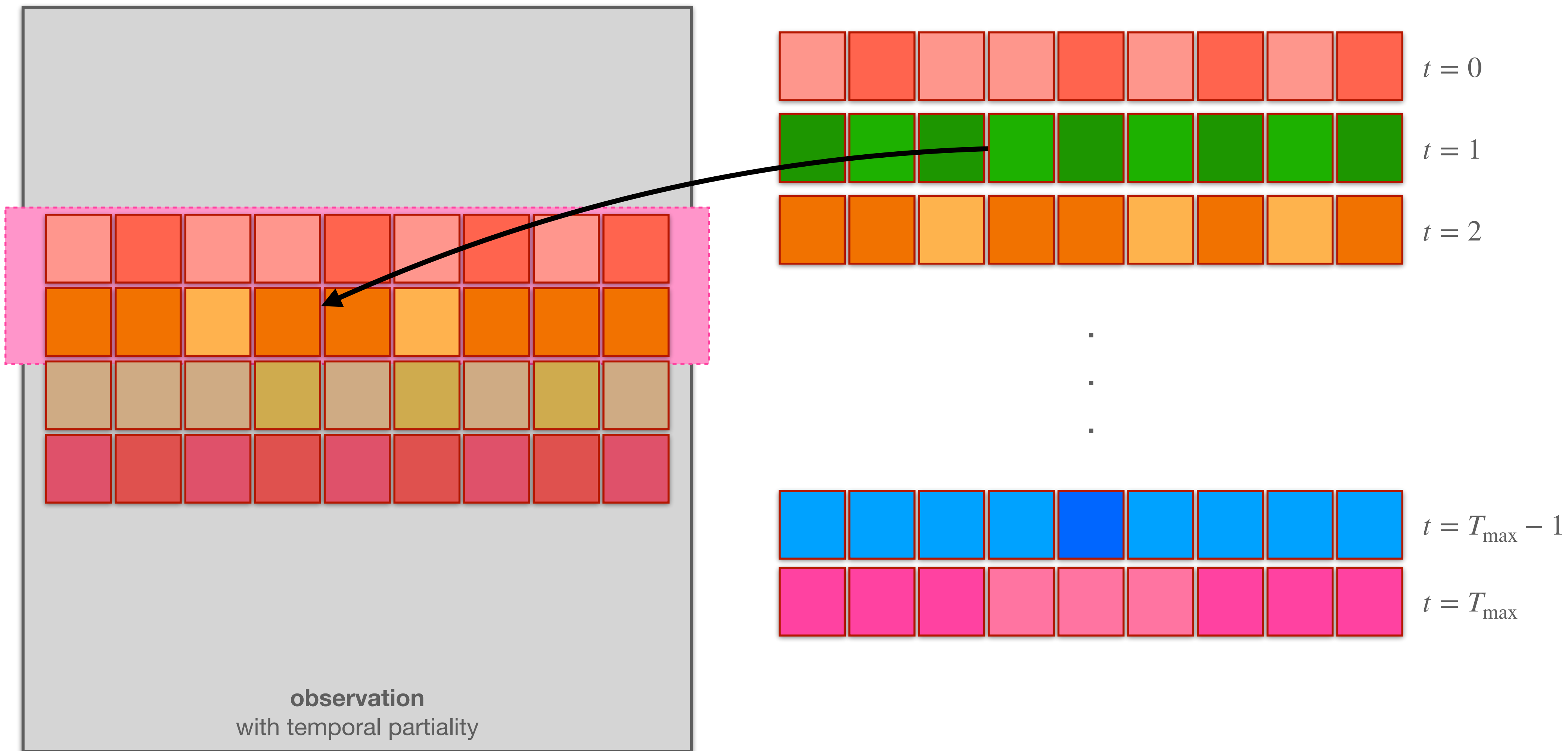




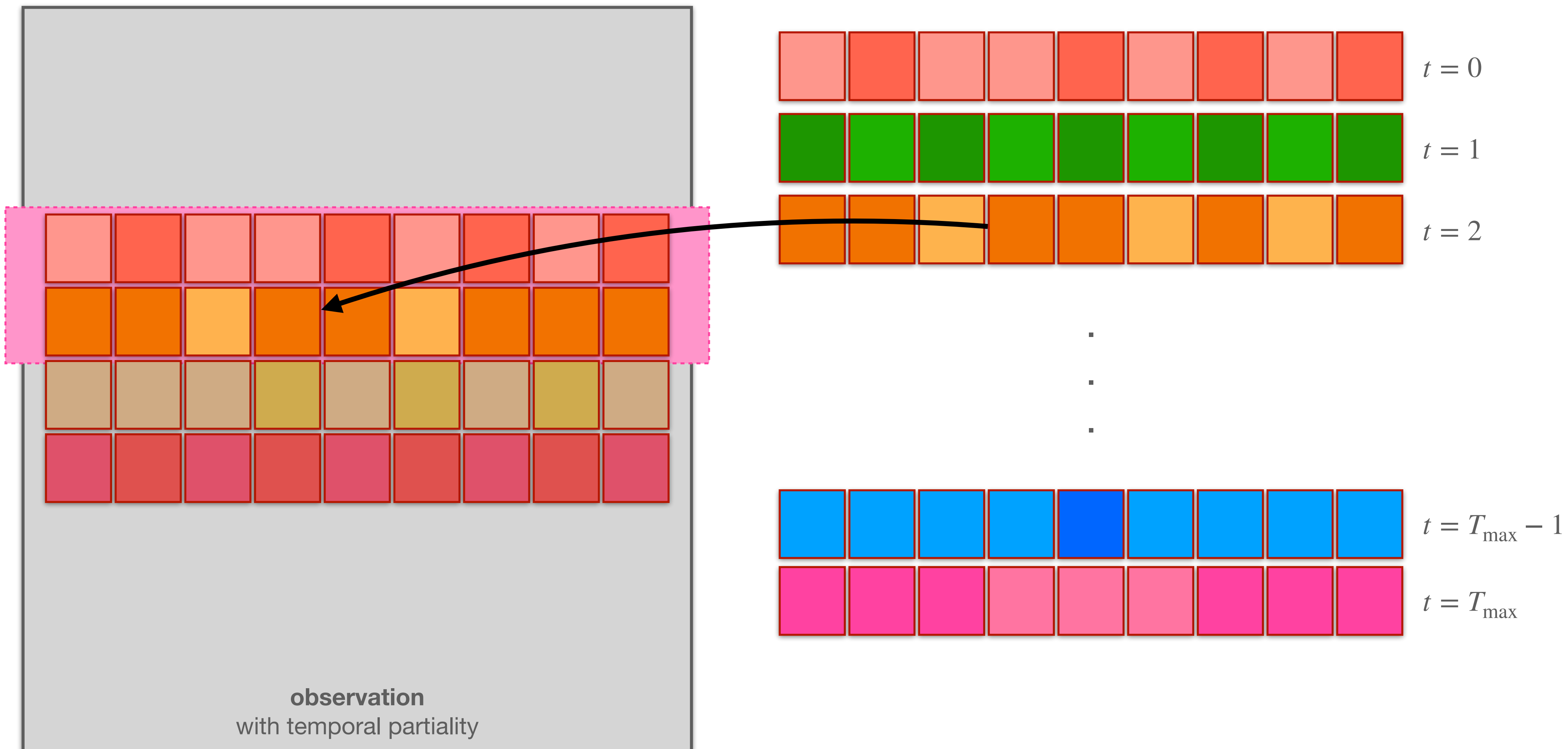
# Fitness function



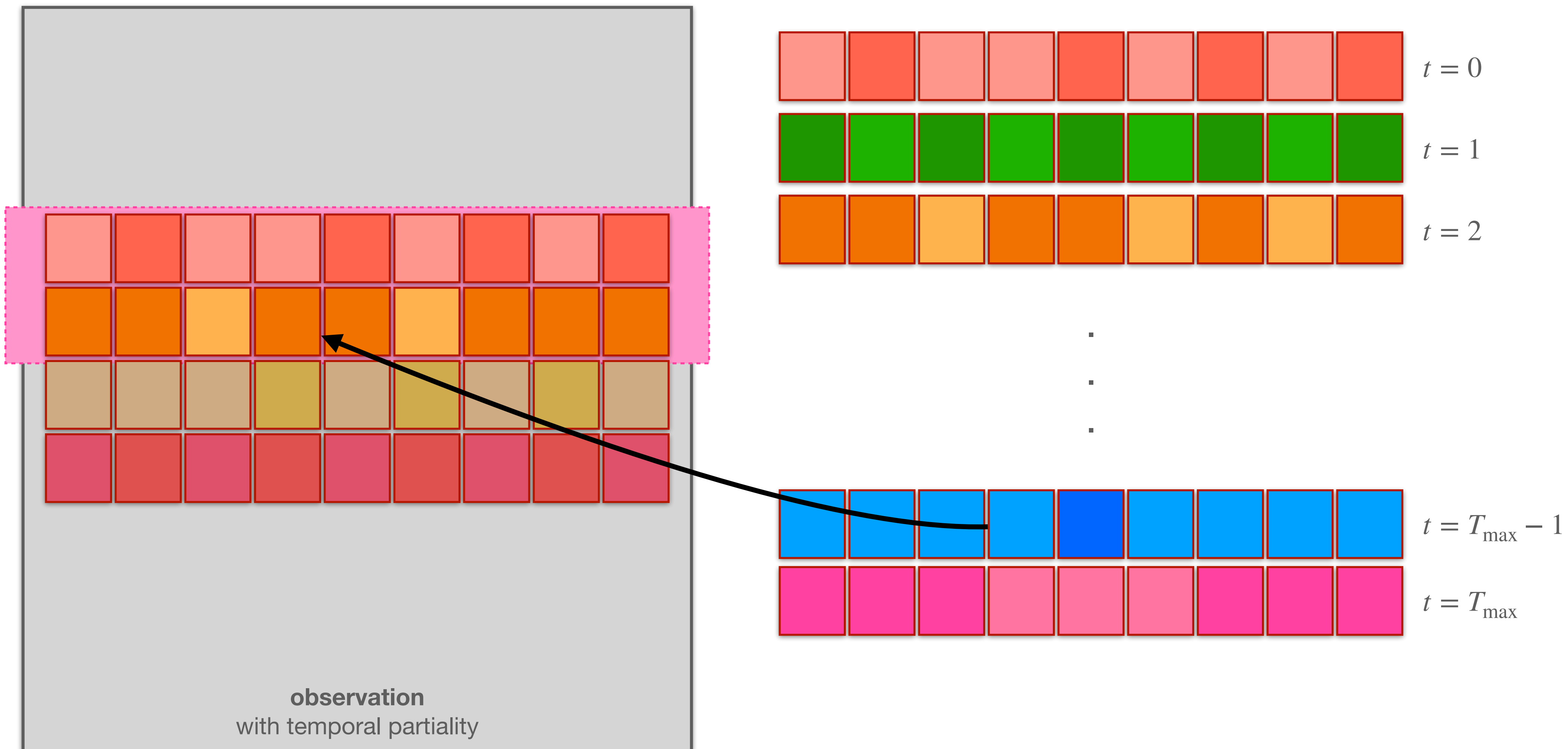
# Fitness function



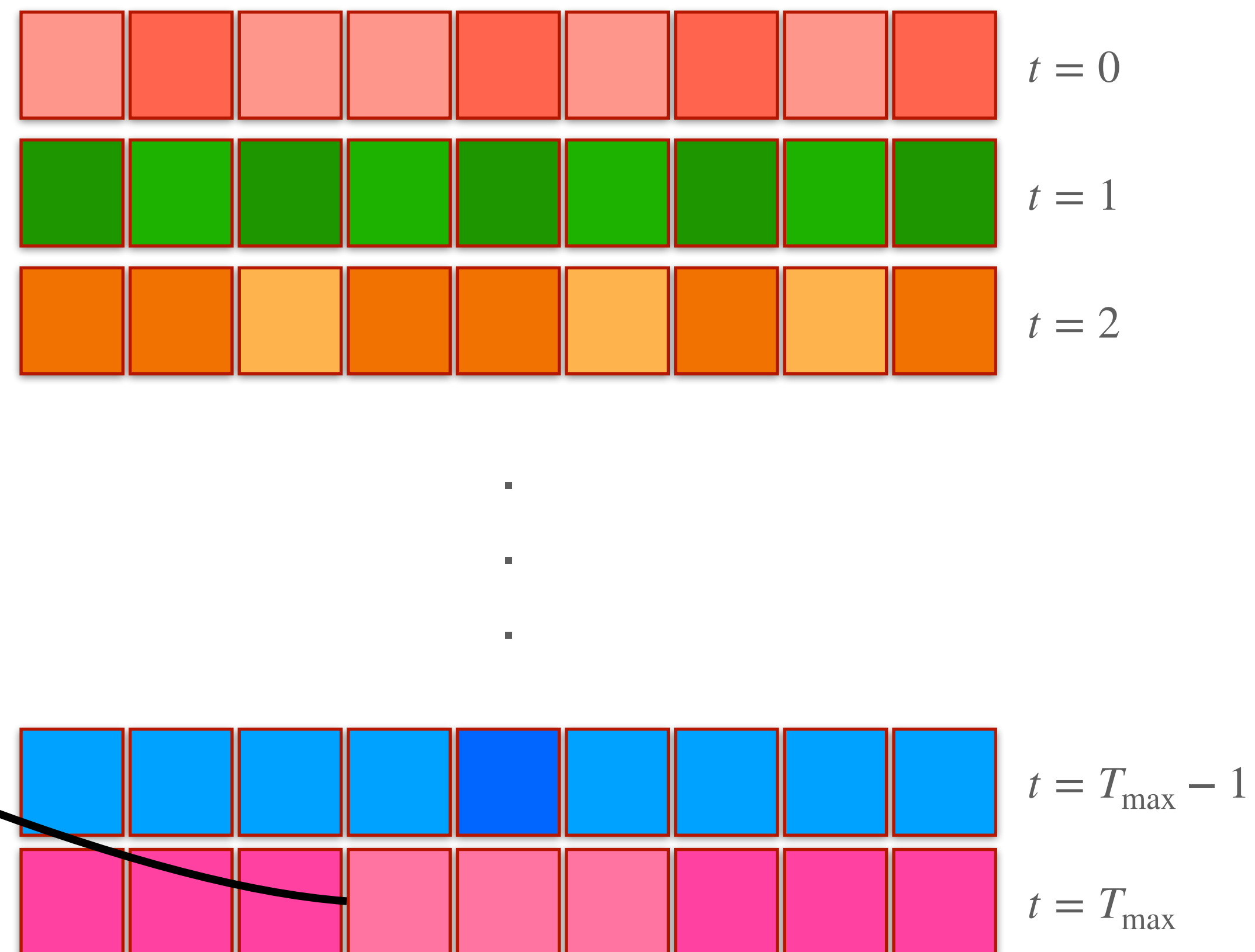
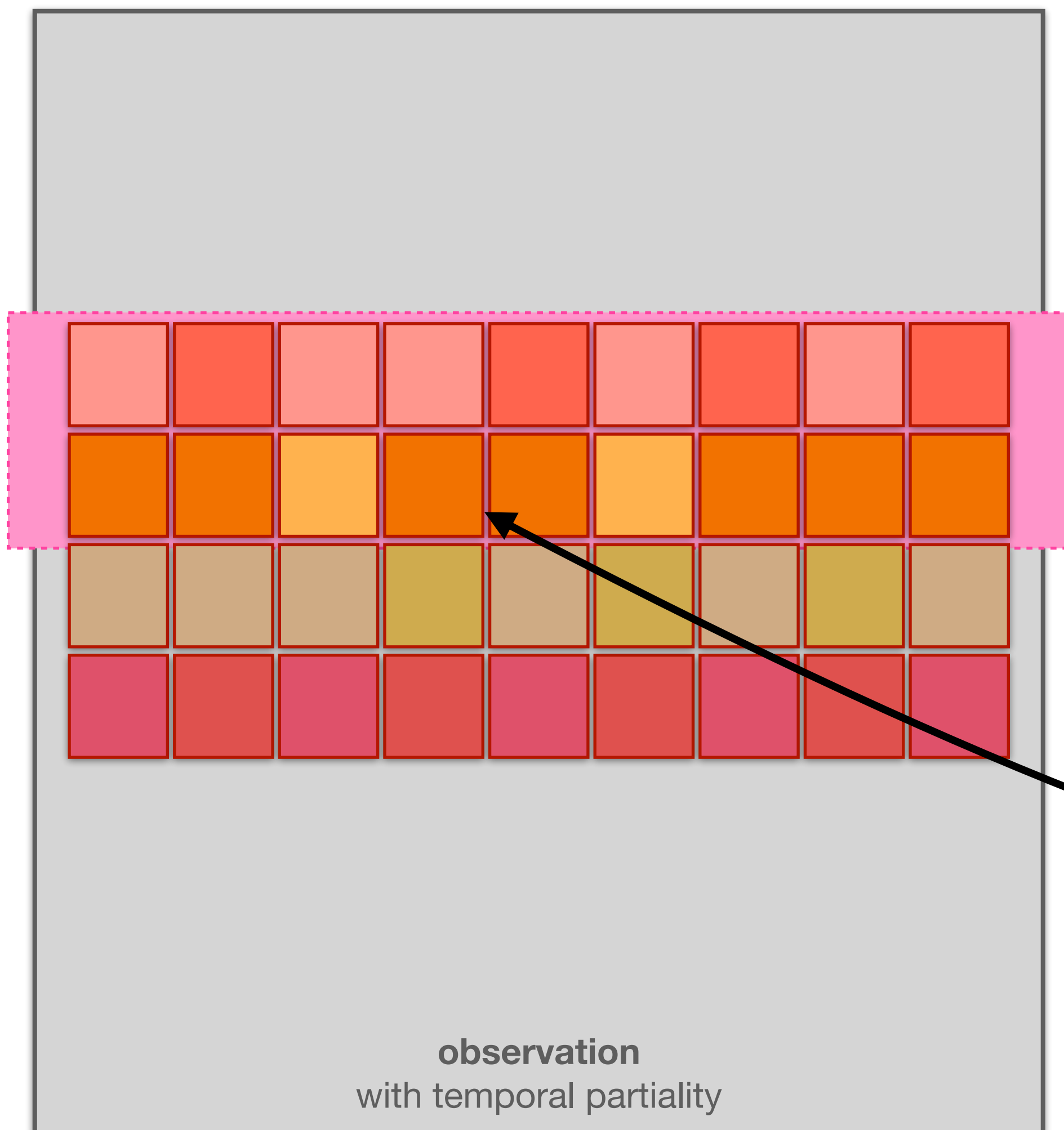
# Fitness function



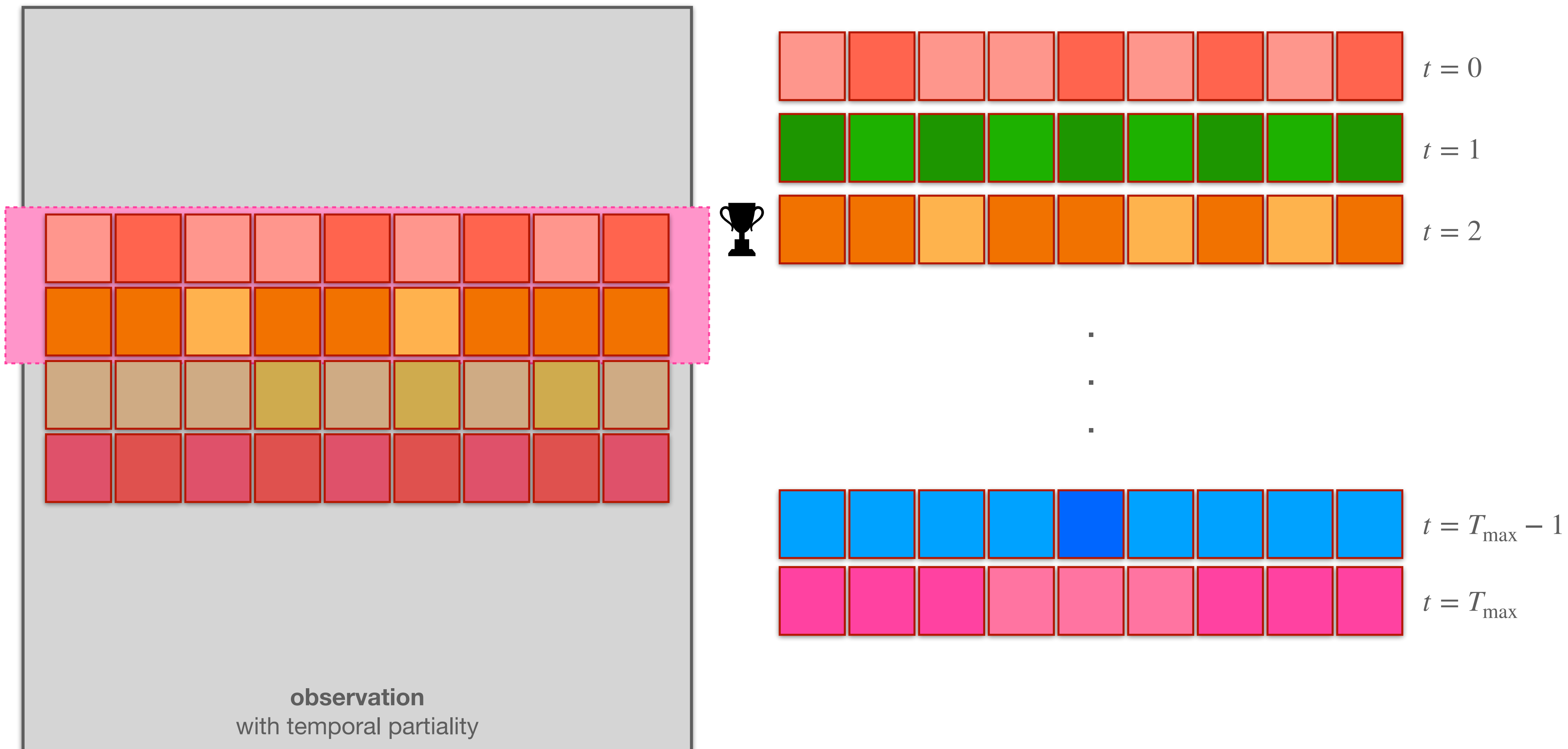
# Fitness function



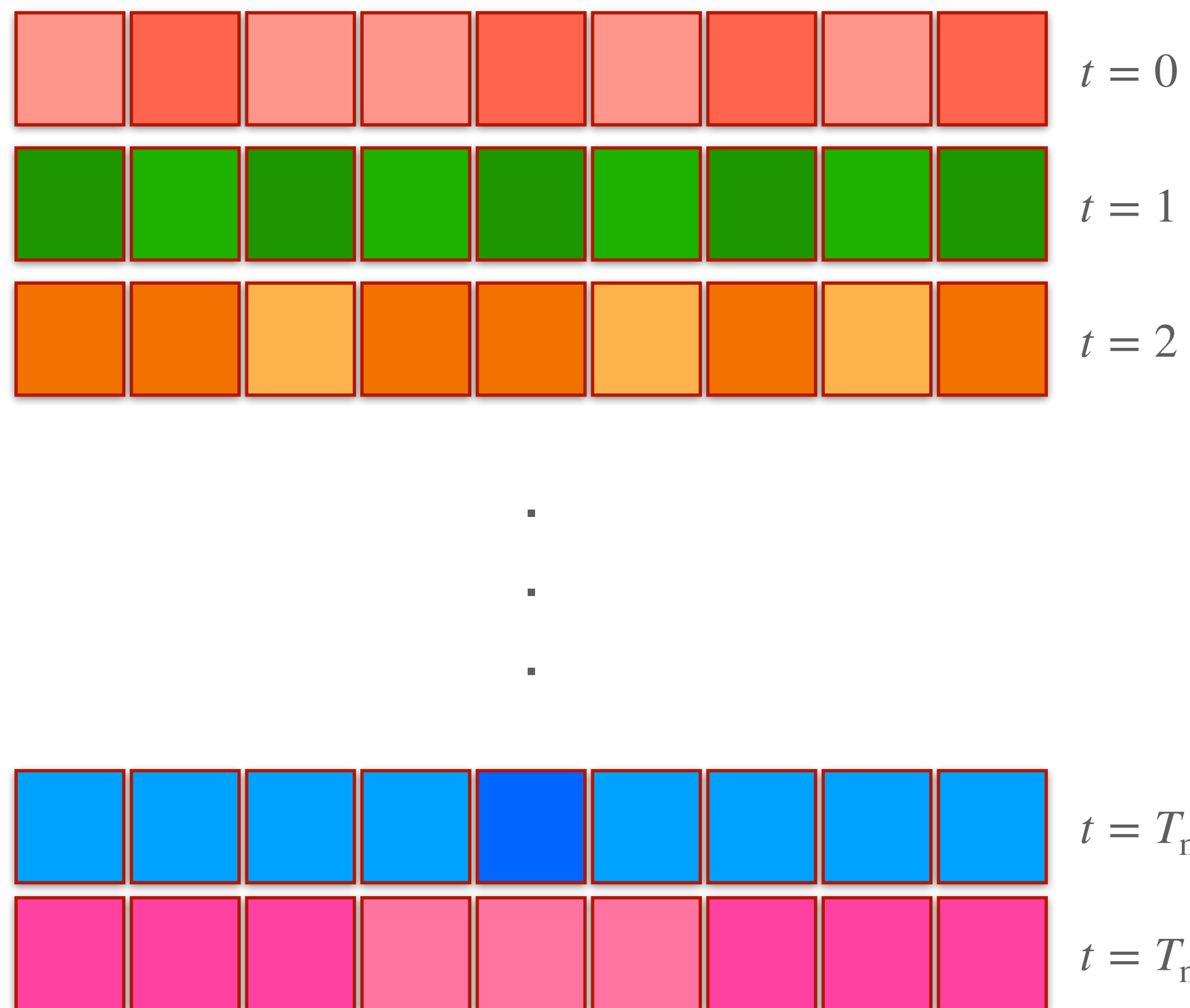
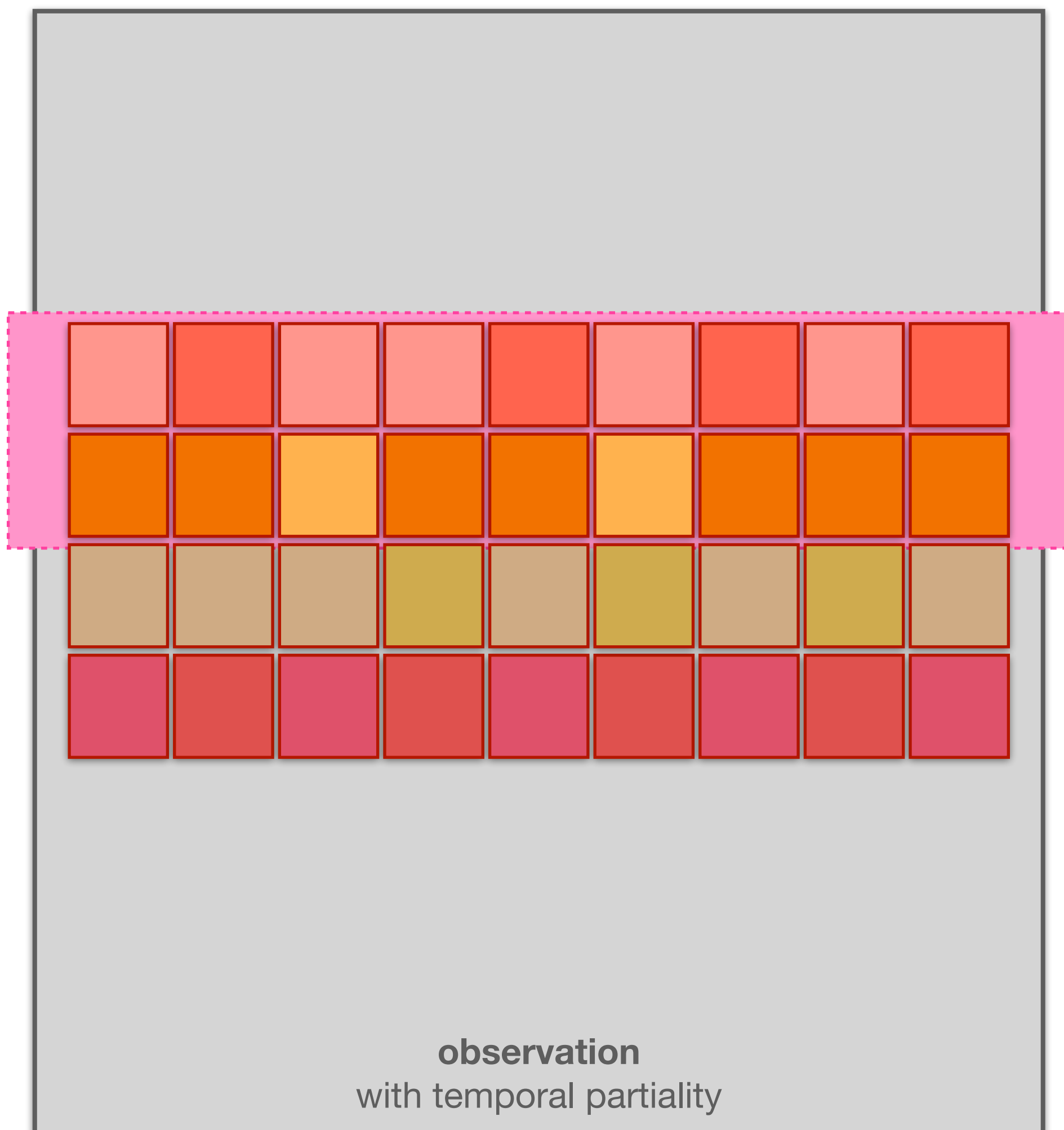
# Fitness function



# Fitness function

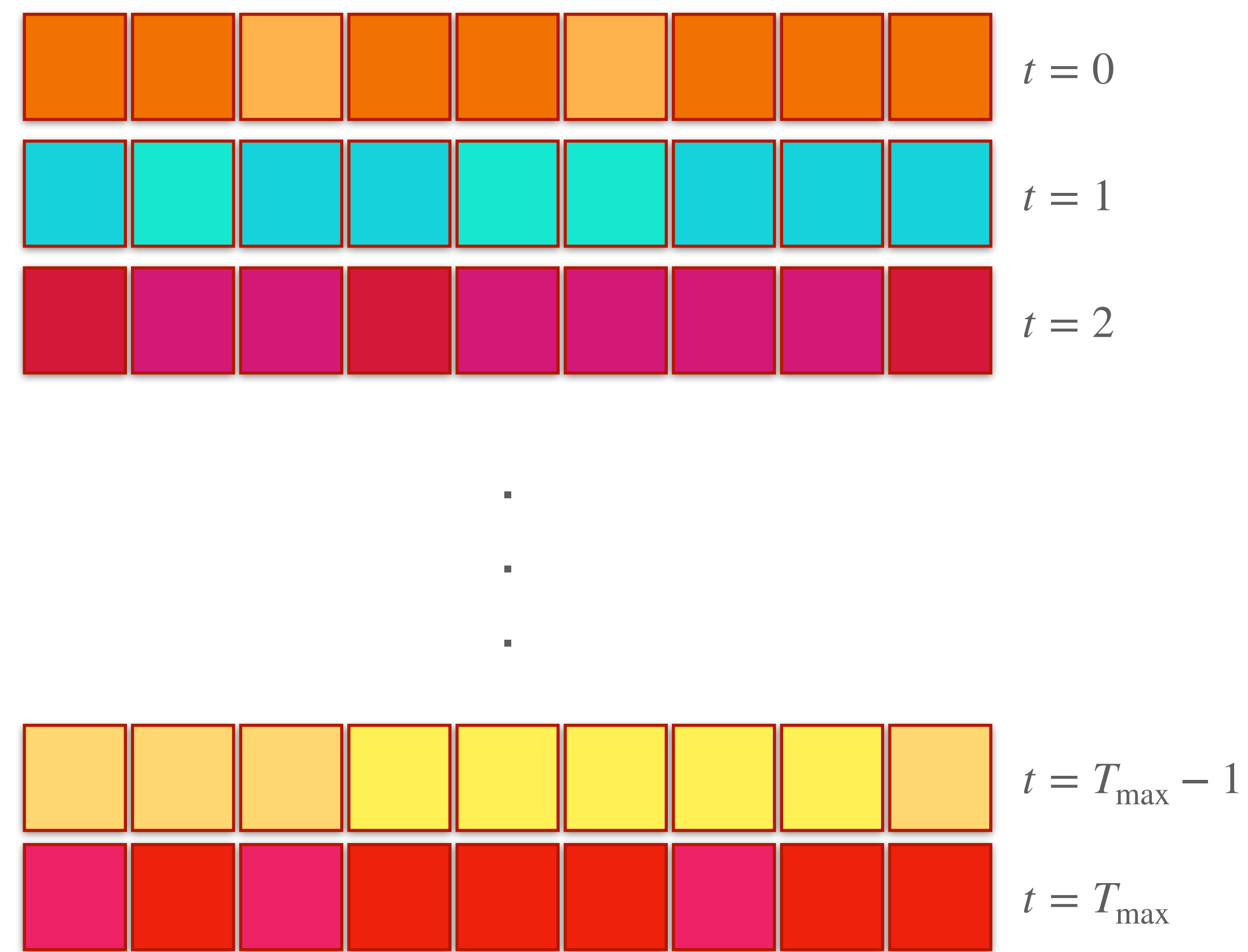
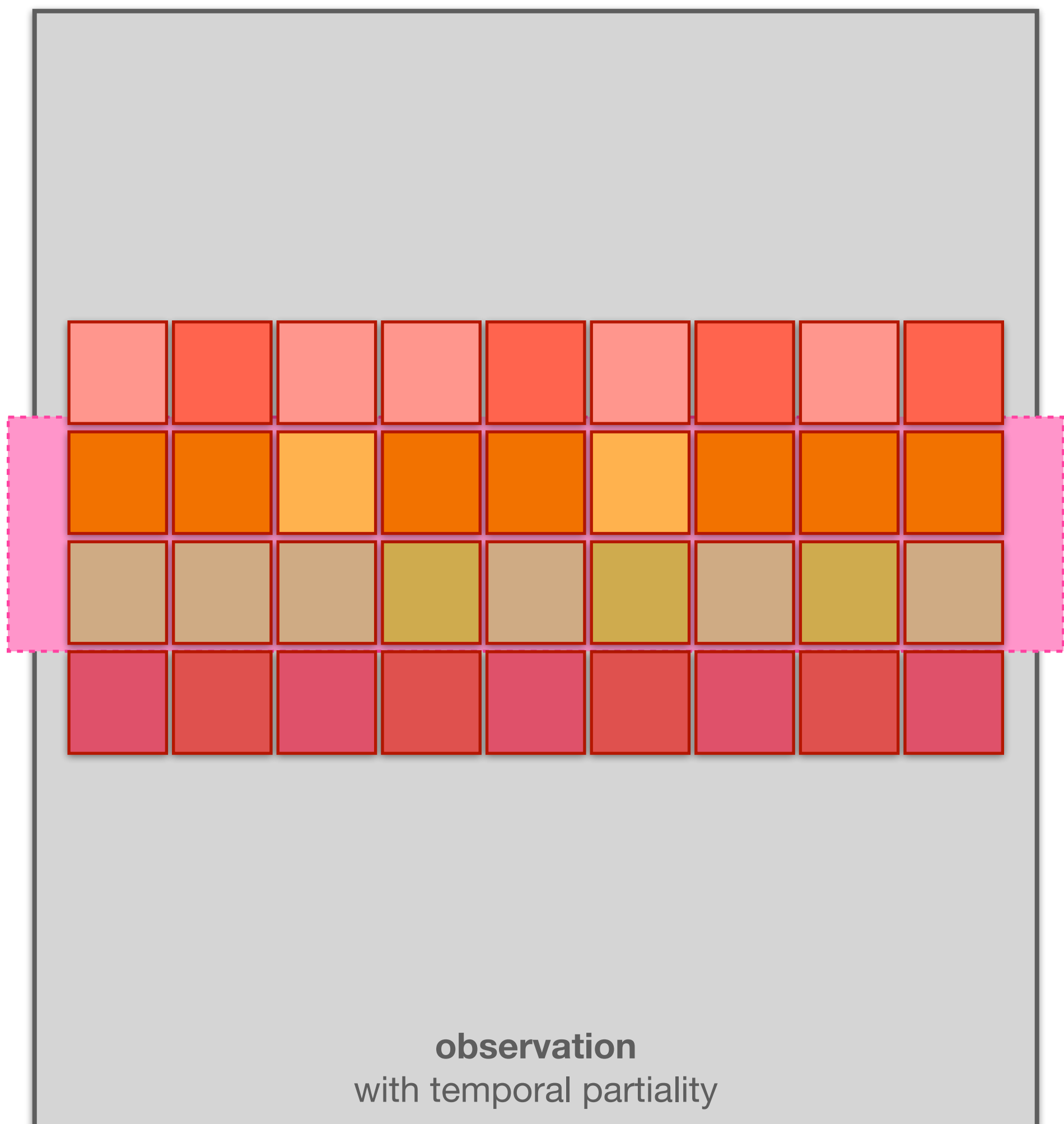


# Fitness function



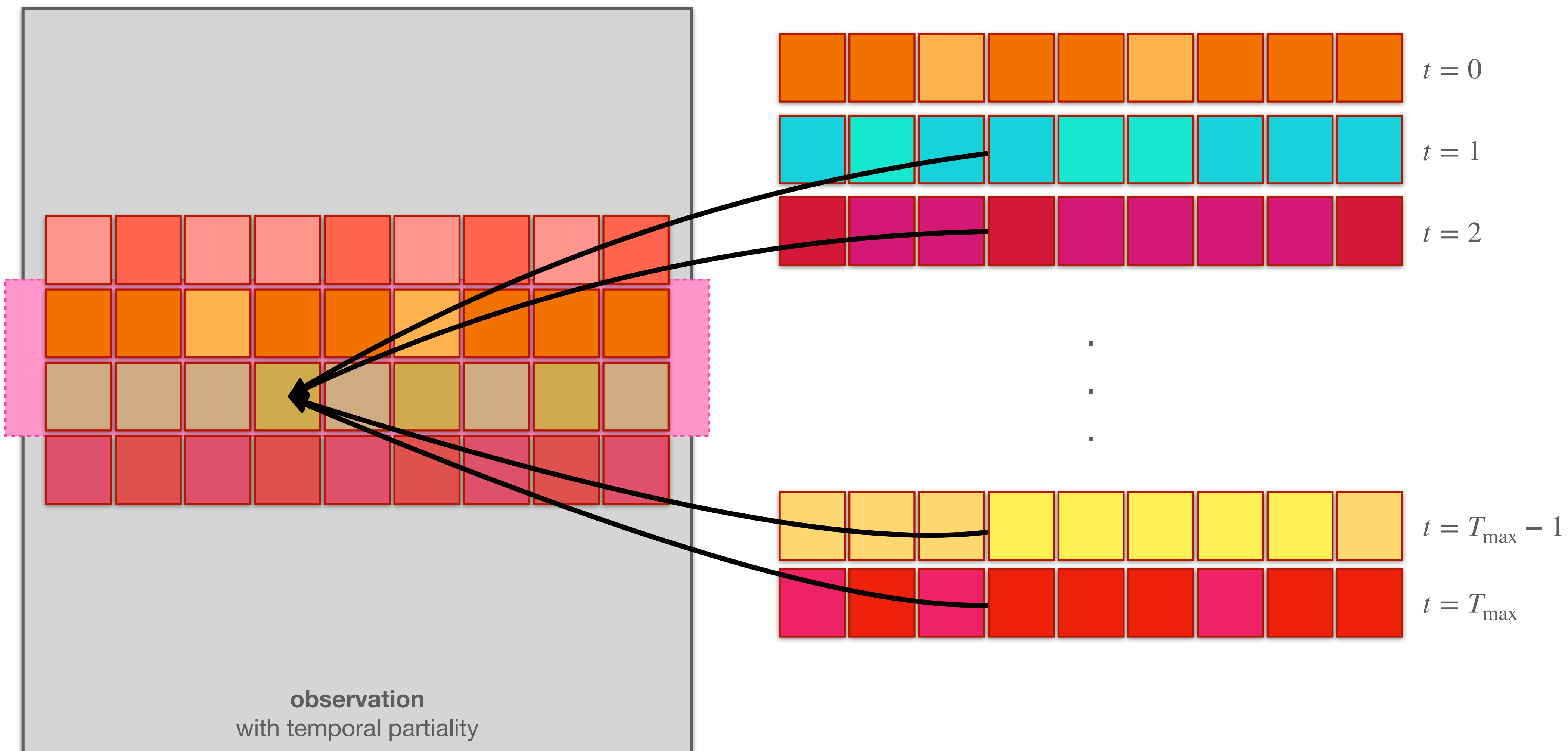
**Pairwise error** equals the number of differences between the observation and the “best” of the outcomes (the one with the smallest number of differences).

# Fitness function

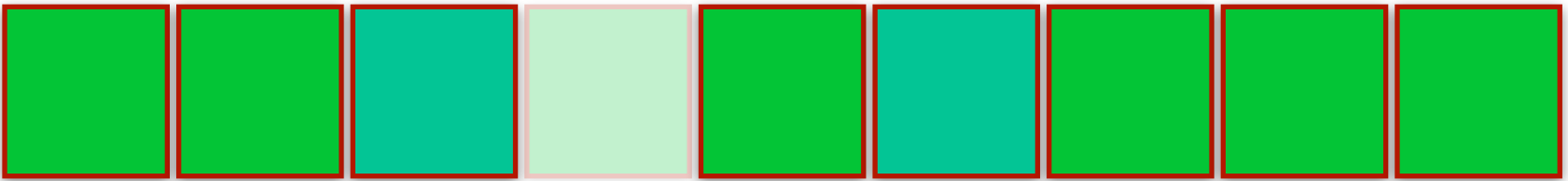
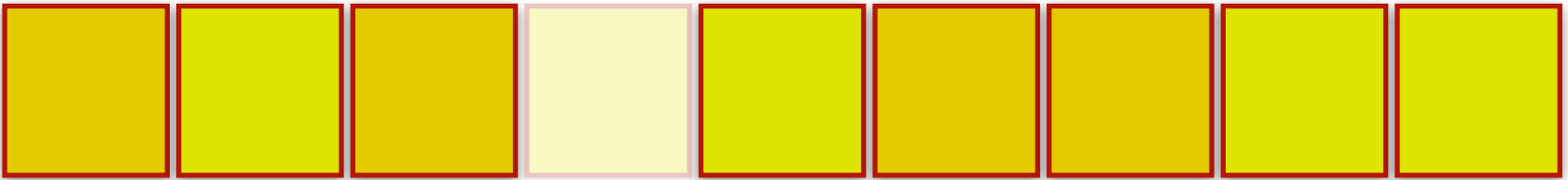
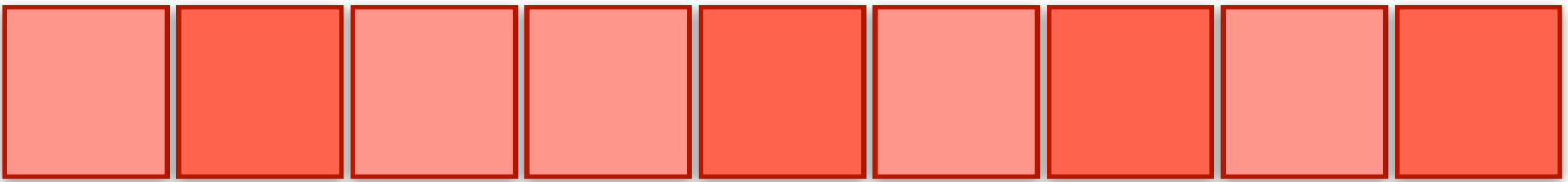
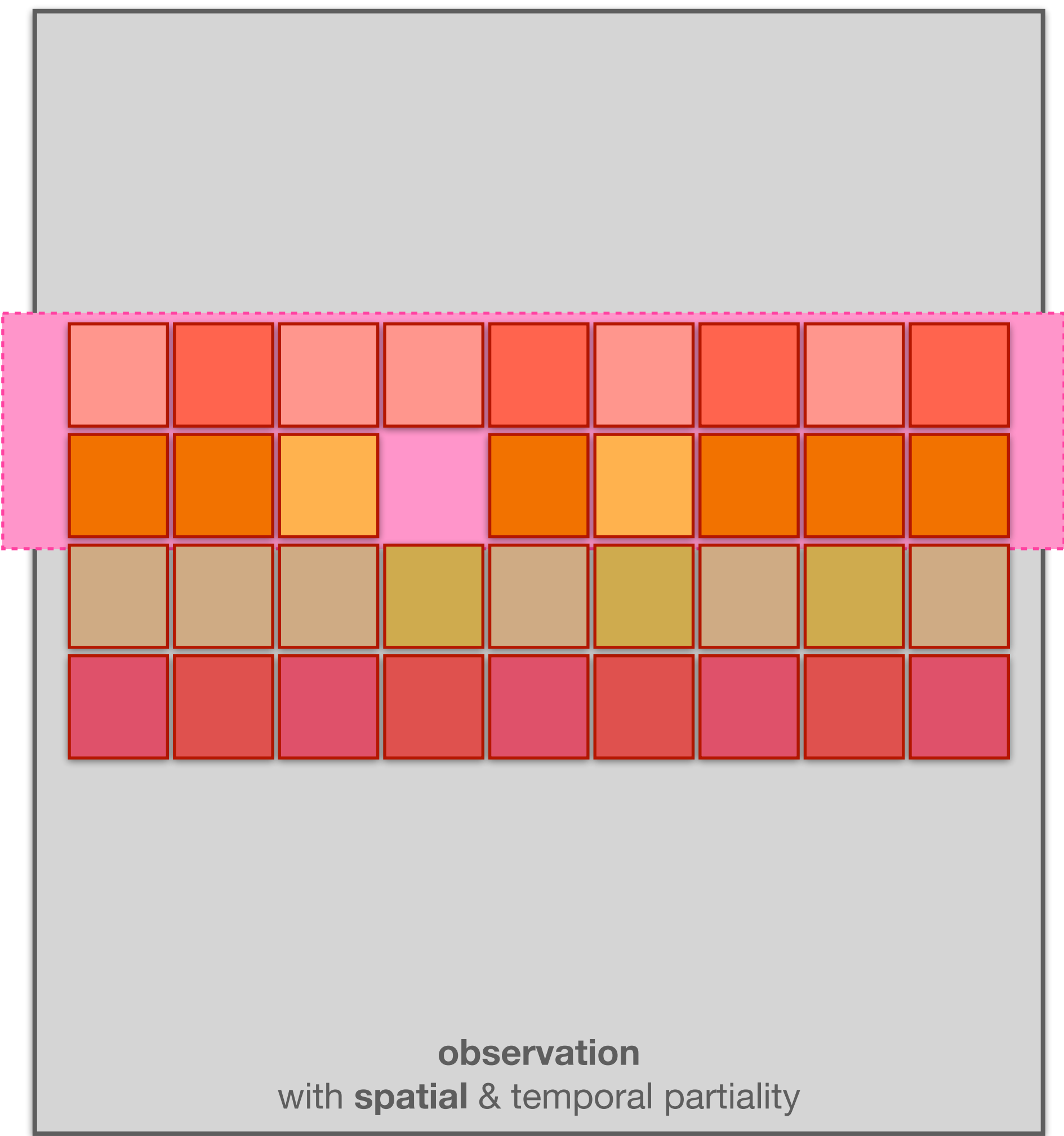




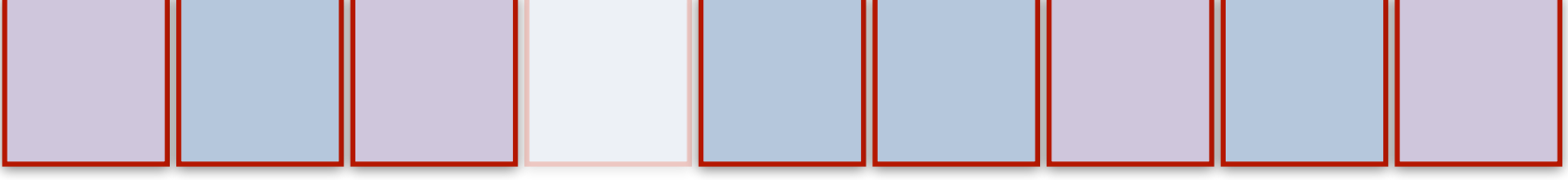
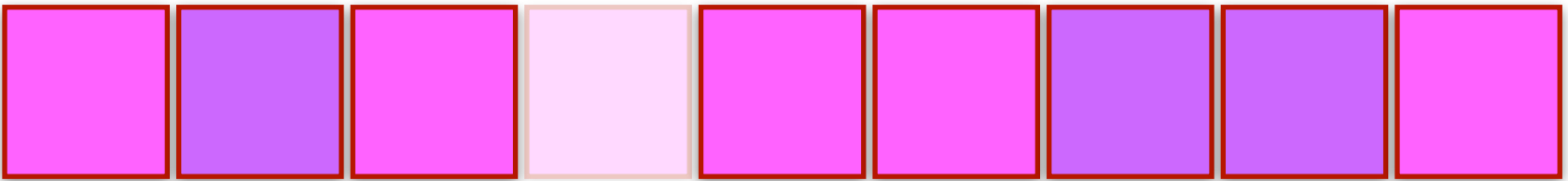
# Fitness function



# Fitness function

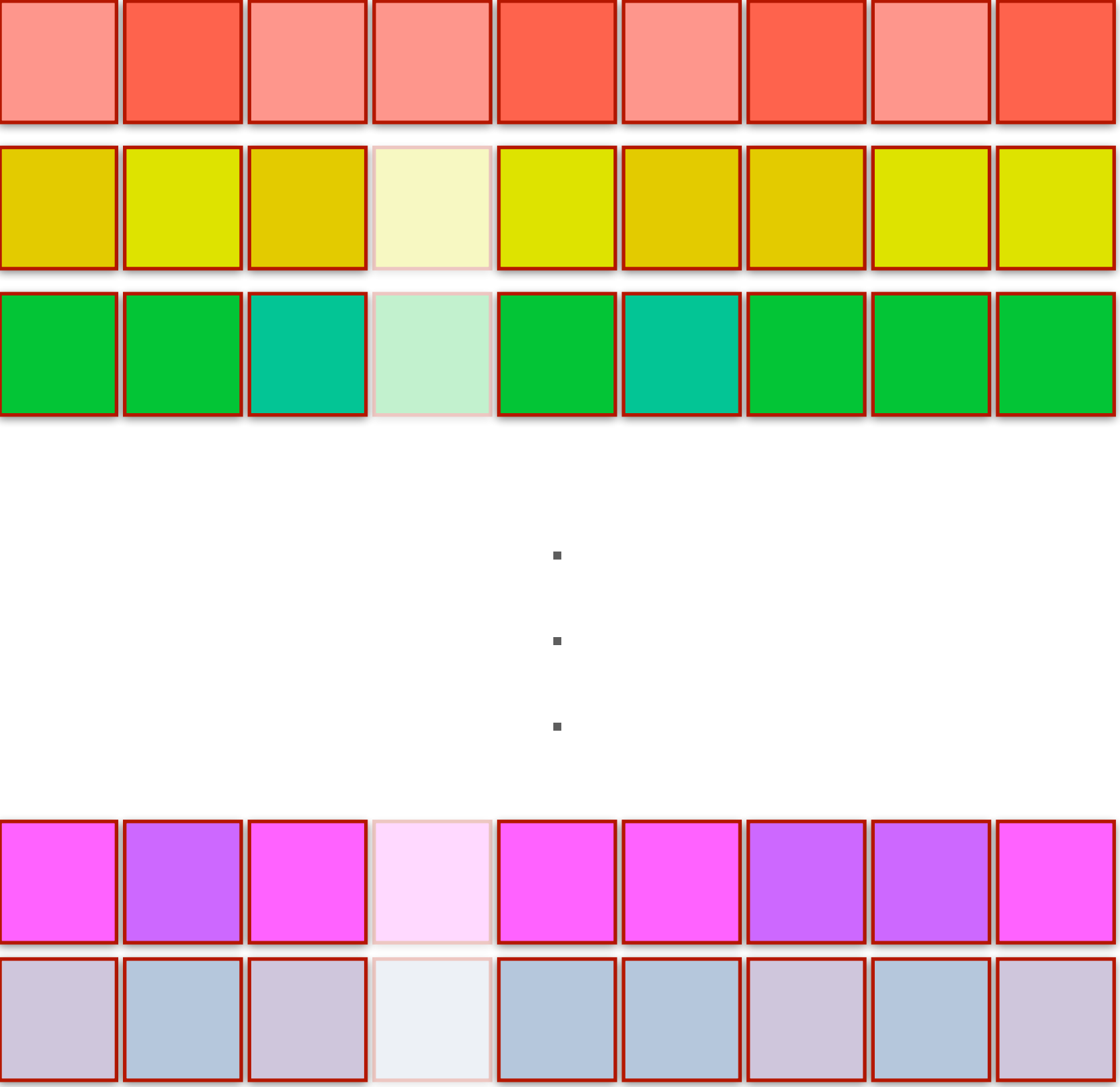
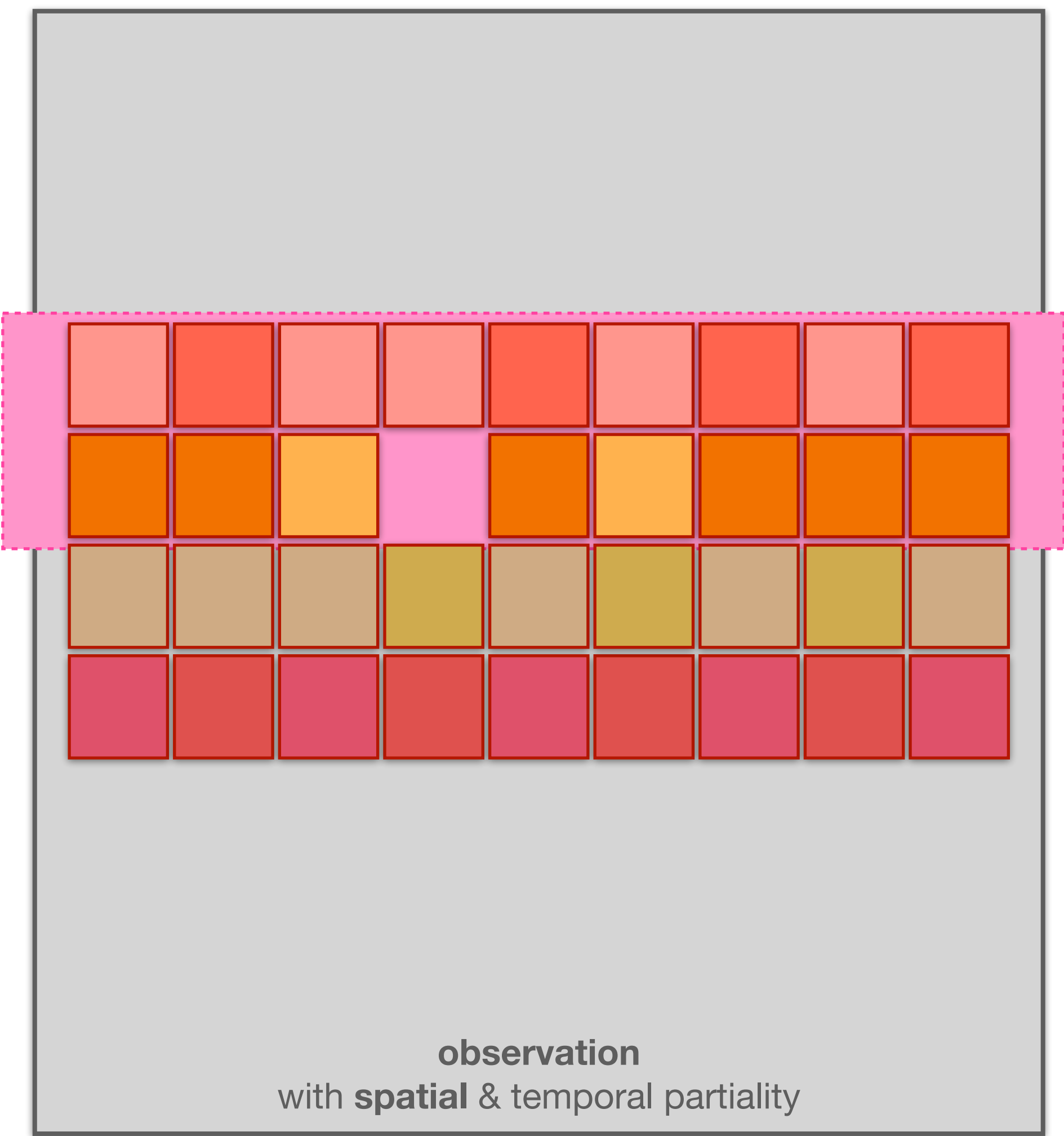


⋮



We **ignore** the missing cell in the error calculation.

# Fitness function



$t = 0$

$t = 1$

$t = 2$

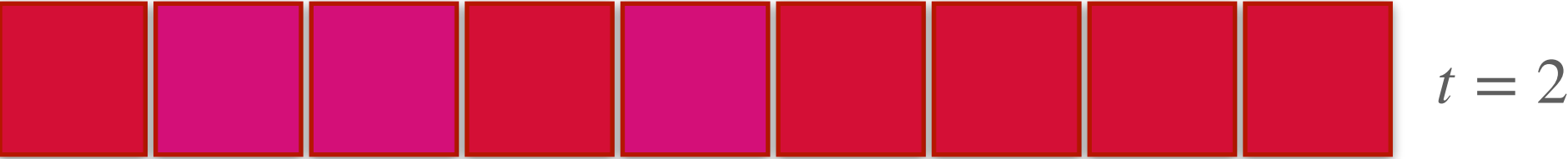
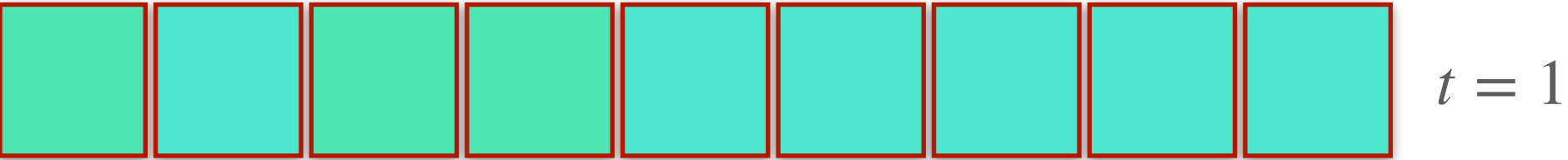
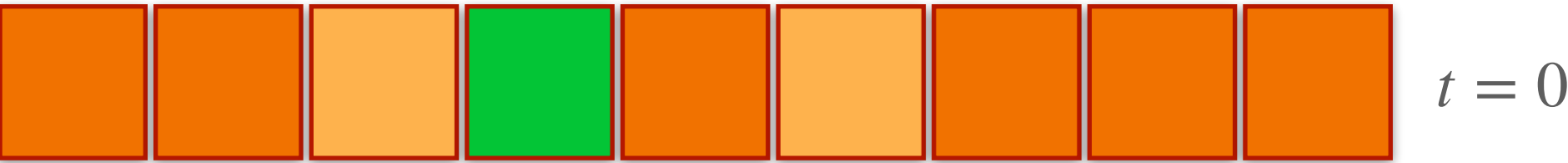
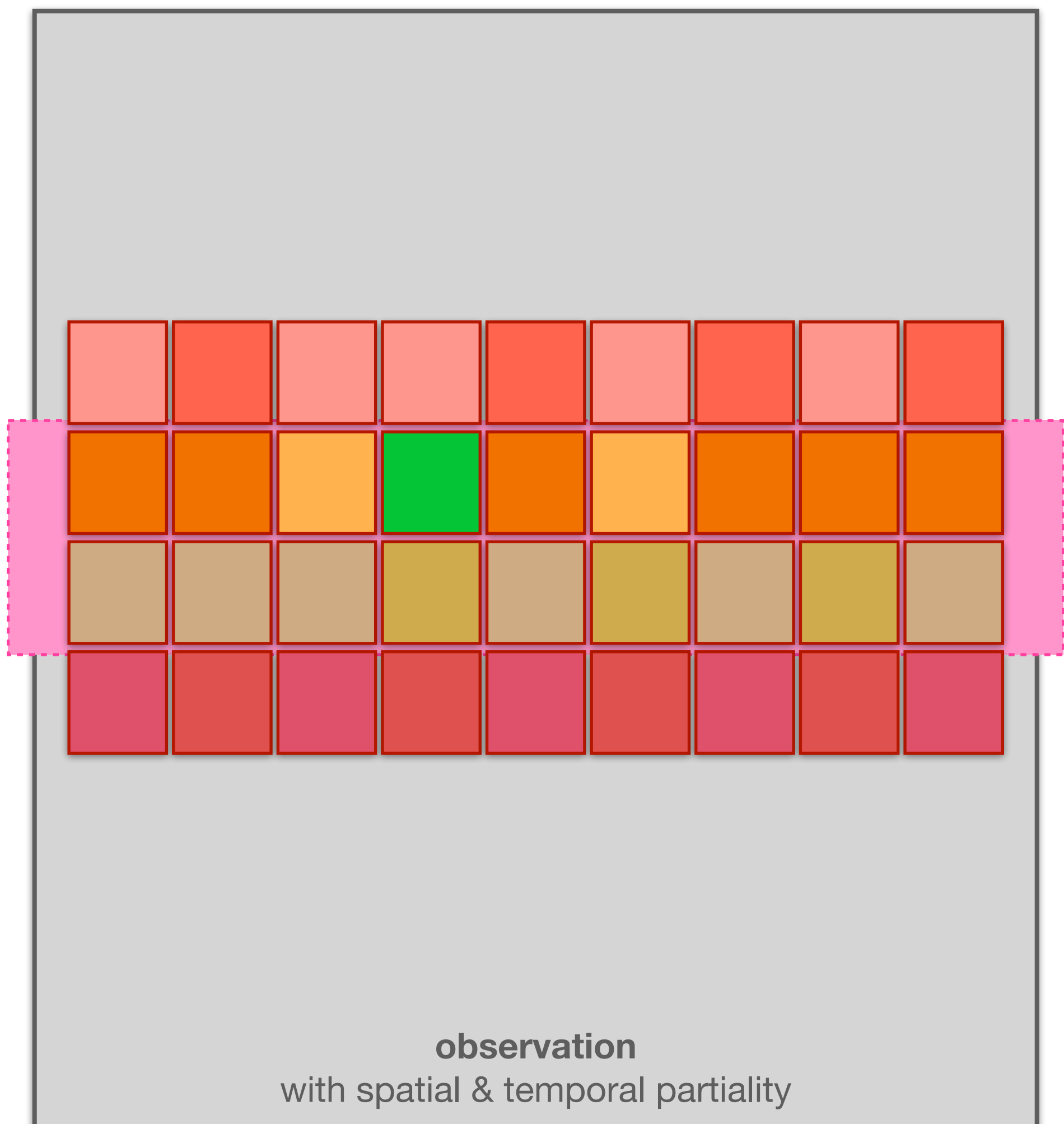
⋮

$t = T_{\max} - 1$

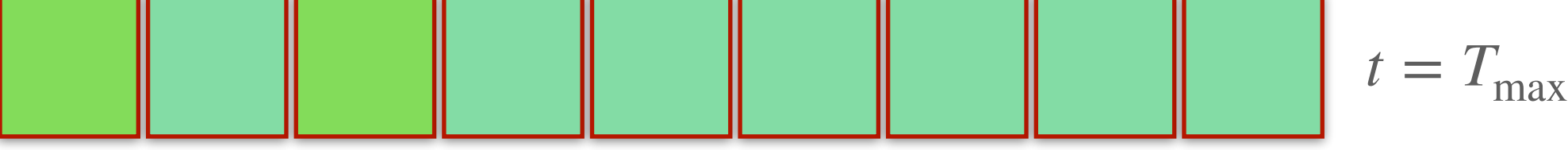
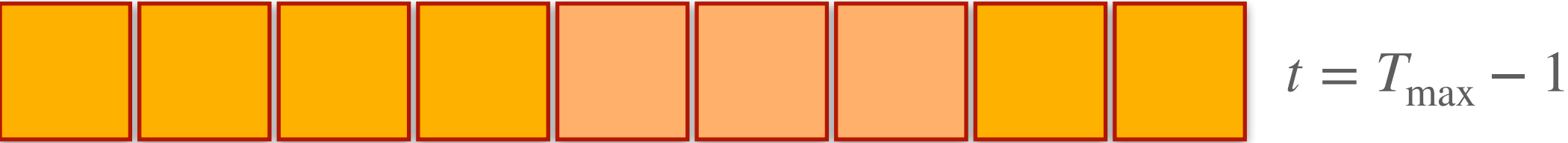
$t = T_{\max}$

We **ignore** the missing cell in the error calculation.

# Fitness function

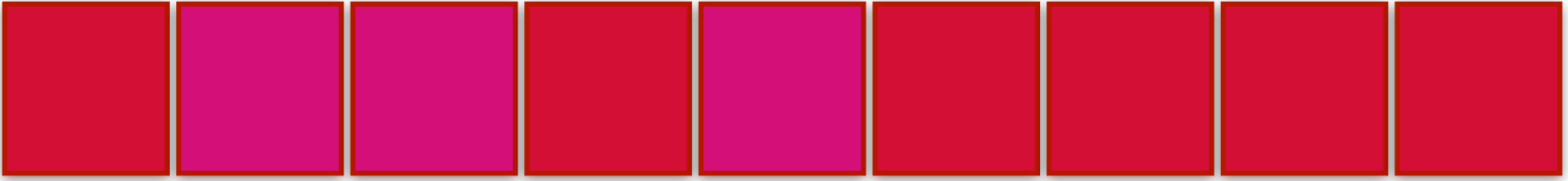
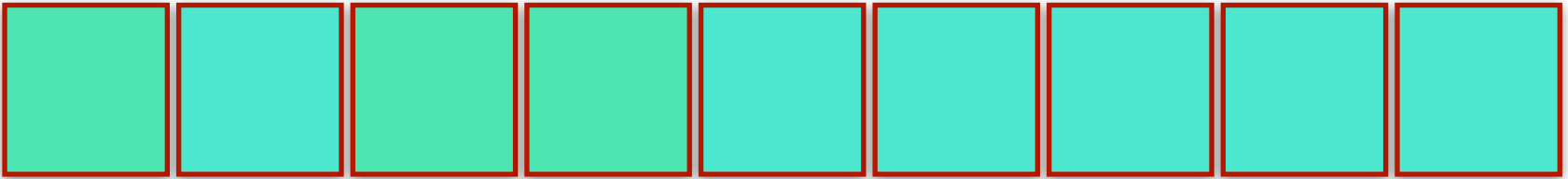
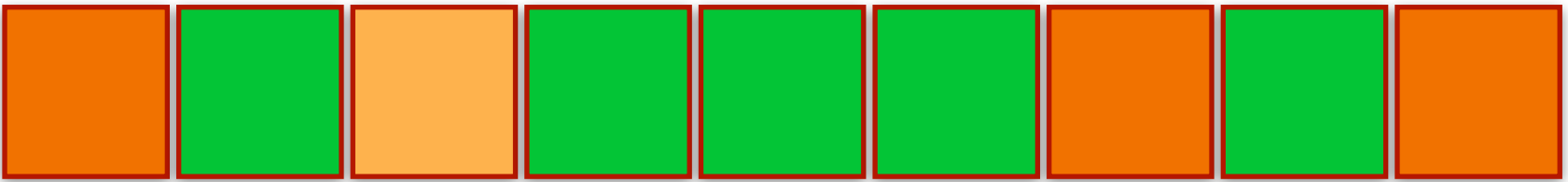
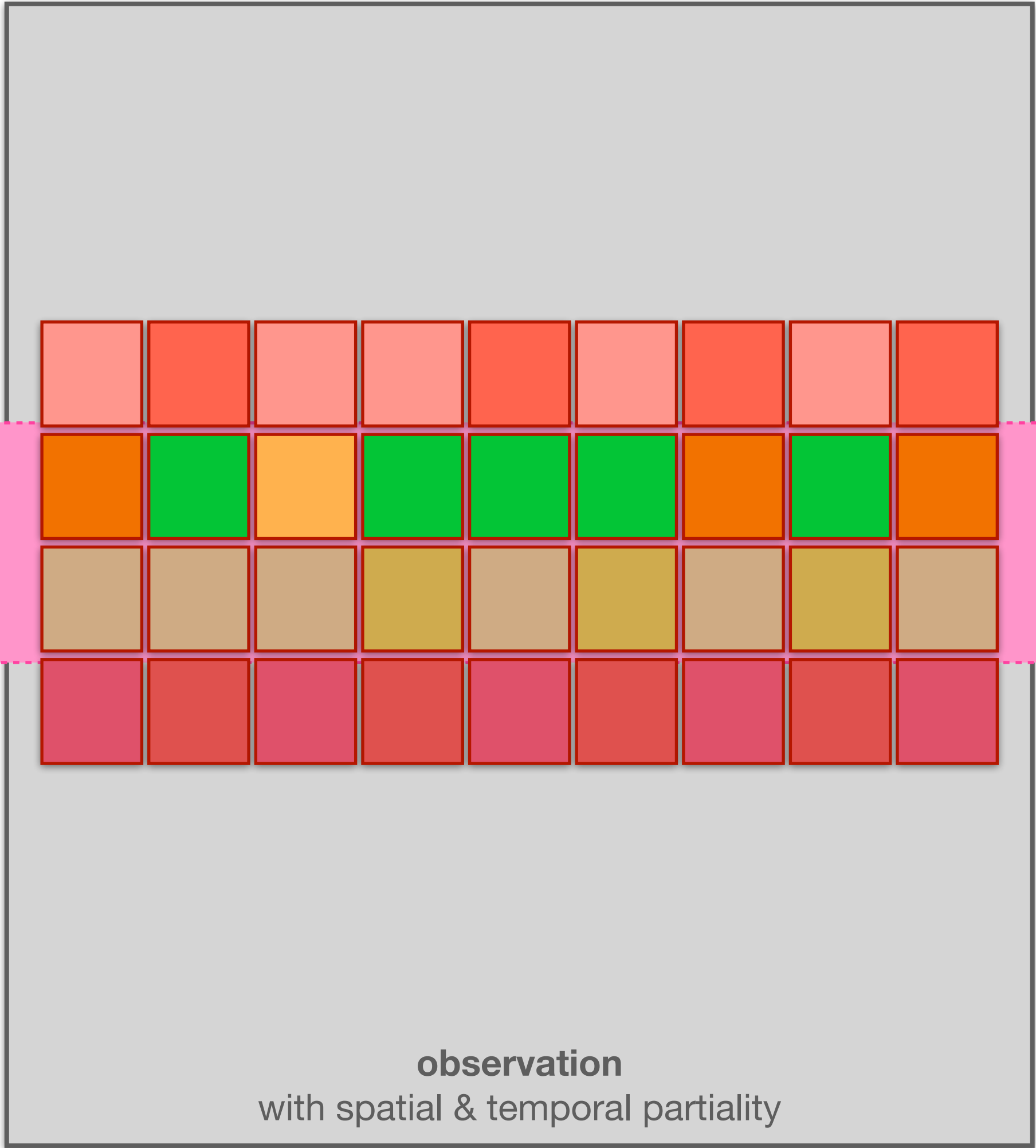


▪  
▪  
▪

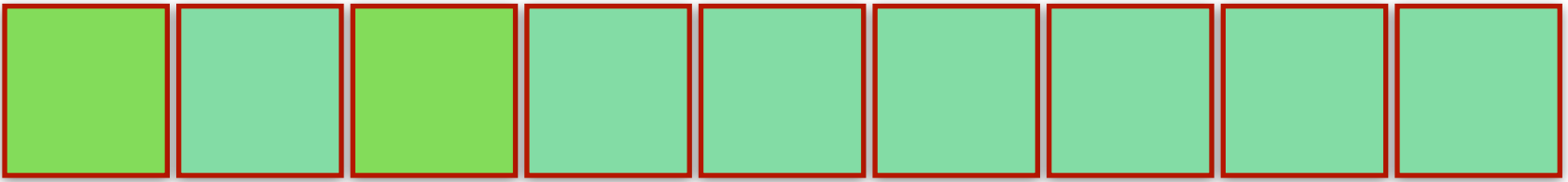
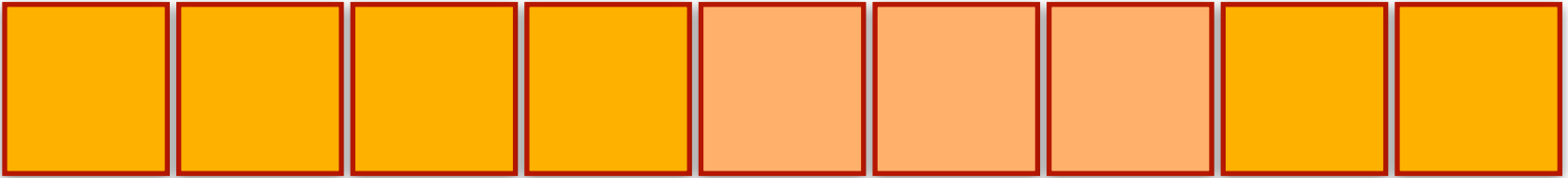


We **fill the gap** (for the purpose of error calculations only) with the value from the “best” row of the previous step of calculations.

# Fitness function



·  
·  
·



$t = 0$

$t = 1$

$t = 2$

$t = T_{\max} - 1$






$t = T_{\max}$

We **fill the gaps** (for the purpose of error calculation only) with the values from the “best” row of the previous step of calculations.

# Implementation

- **C** language for high performance
- **OpenMP** for multi-core processing - fitness calculations on population
- **HPC** batch distribution using Python / shell scripts (**no** inter-node communication for performance)
- **Python** for post-processing of the results
  - **Matplotlib** and **gnuplot** for visualization
- Source code available on **GitHub**
- Alternative implementation available in **Python** - easier to read, slower to run

# Experimental results [B5]

Experiment	Setup / source of observations	Question	Answer
1	temporally partial observations of ECAs GA search space: local rules with radii between 2 and 5	Are there differences in identification <b>efficiency</b> depending on the dynamical properties of ECAs?	<b>yes</b>  more <i>complex</i> rules are harder to identify
2	same as in Exp. 1, but the source rules are randomly selected from radius-2 space	Are ECAs treated as radius-2 rules <b>special</b> in any way?	<b>no</b>  on average similar effort is required for radius-2 rules
3	same as in Exp. 1, but time gaps now have constant length (different variants)	Is <b>randomness</b> of the time gap lengths important?	<b>yes</b>  e.g., odd lengths are harder
4	same as in Exp. 1, but spatial partiality is added (different variants)	Is there a link between dynamical complexity and <i>sensitivity</i> to the amount of <b>spatial partiality</b> in observations?	<b>yes</b>  on average more complex rules are less tolerant
5	randomly selected radius-2, radius-3 and radius-4, temporal partiality	Is the GA able to correctly <b>identify</b> the unknown <b>radius</b> of the solution?	<b>yes</b>  entire population converges to the “correct” radius before finding the final solution

# Identification of Stochastic Cellular Automata



# Informal definition of a Stochastic CA

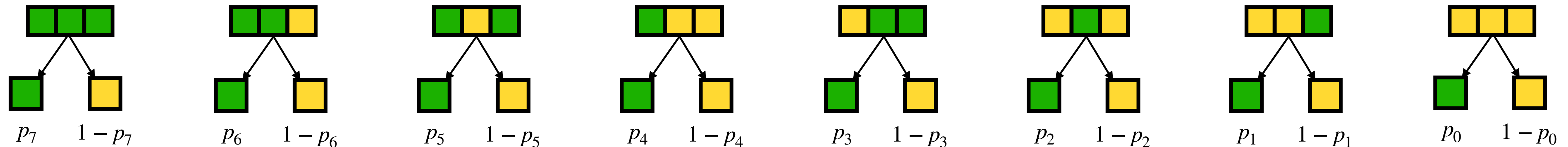
- **Stochastic CA (SCA)** is *just like* a deterministic CA, but with a **non-deterministic local rule**, which is described by a probabilistic LUT (pLUT):

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0 \end{bmatrix}.$$

# Informal definition of a Stochastic CA

- **Stochastic CA (SCA)** is *just like* a deterministic CA, but with a **non-deterministic local rule**, which is described by a probabilistic LUT (pLUT):

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0 \end{bmatrix}.$$



# Informal definition of a Stochastic CA

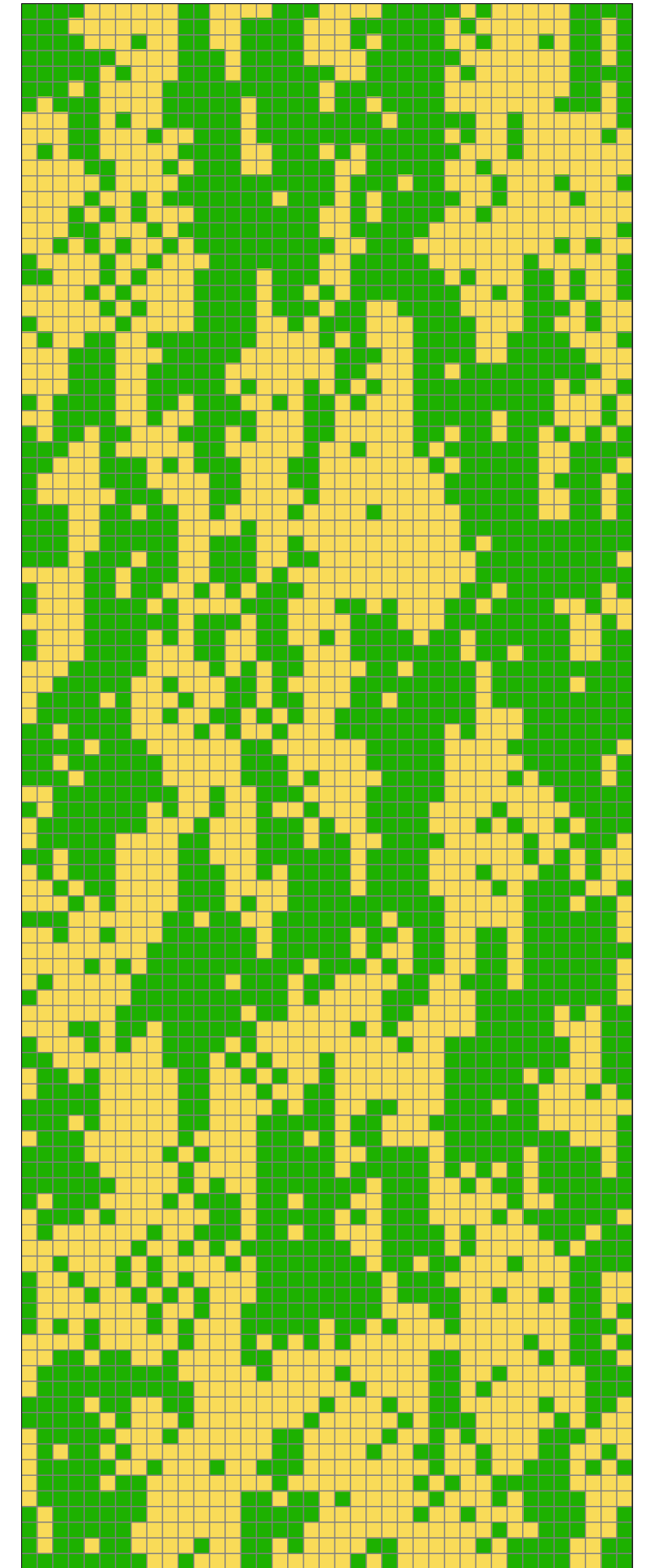
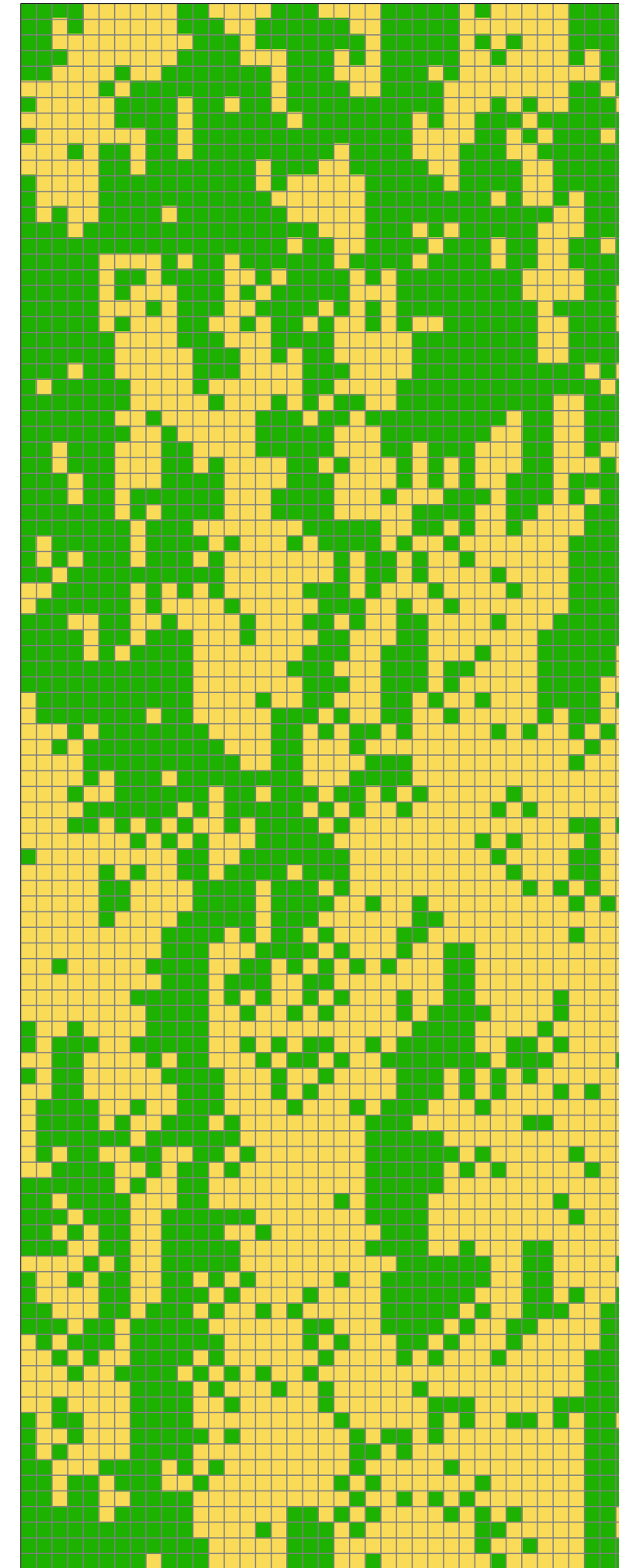
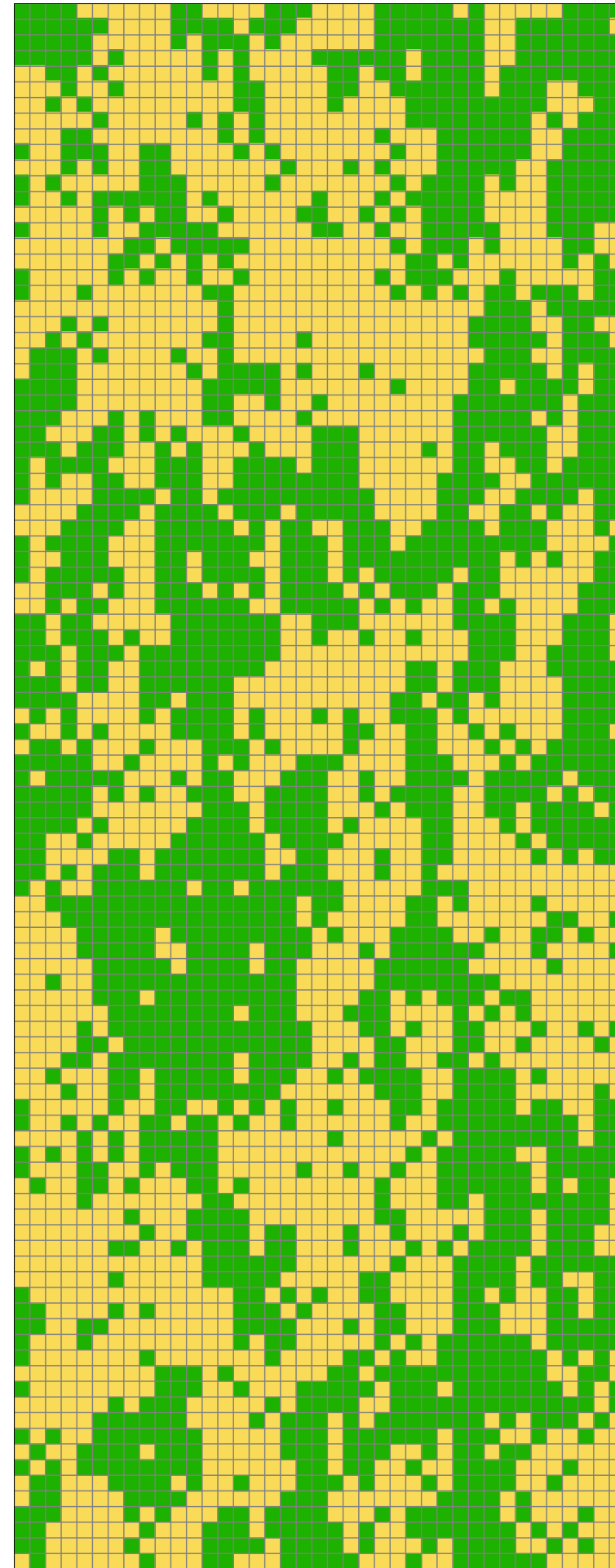
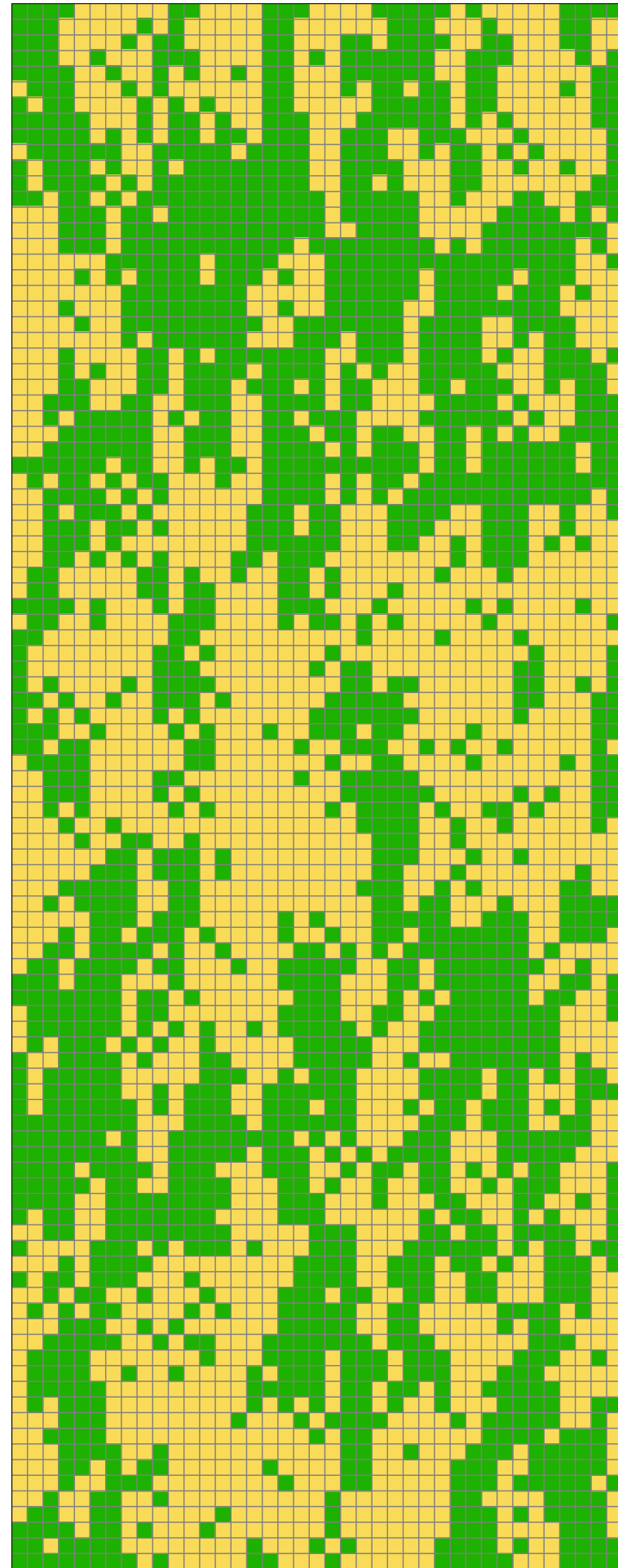
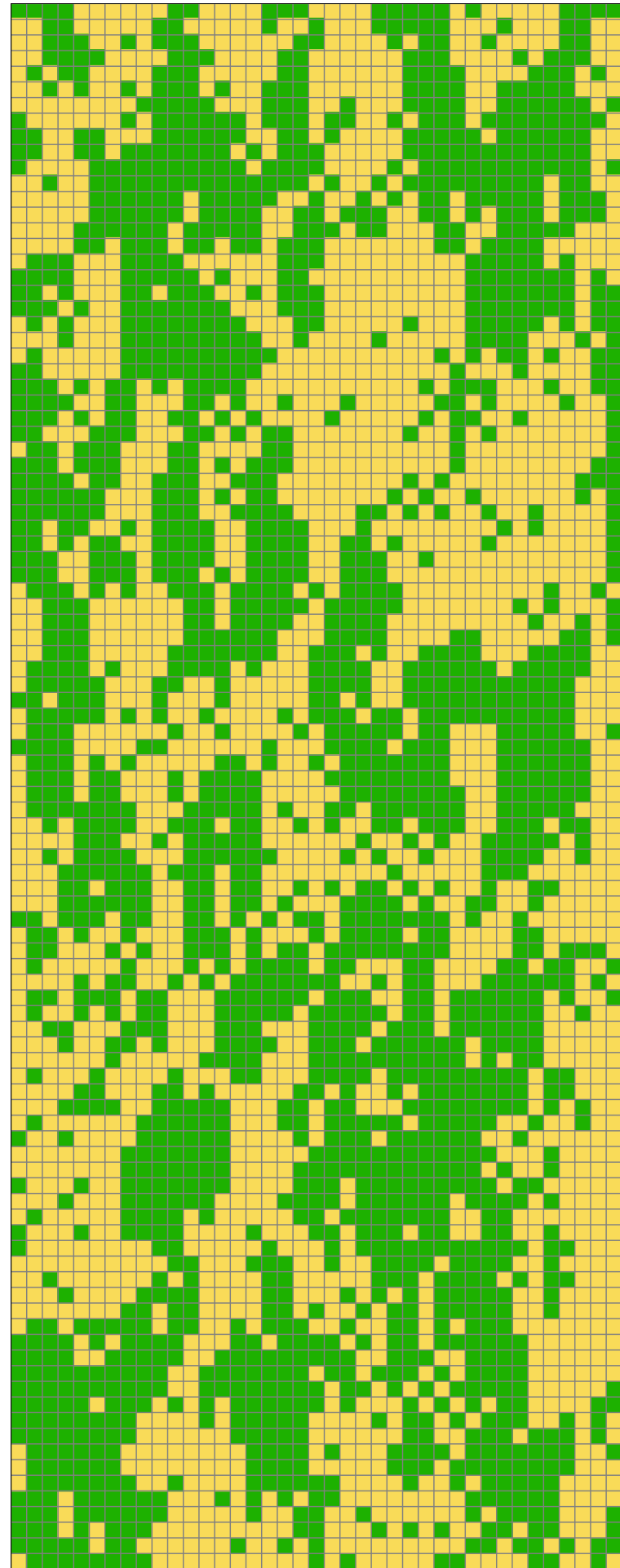
- **Stochastic CA (SCA)** is *just like* a deterministic CA, but with a **non-deterministic local rule**, which is described by a probabilistic LUT (pLUT):

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0 \end{bmatrix}.$$

- An SCA **evolves in time** according to these “local” probabilities.
- Therefore, an SCA is a **stochastic process** in the global sense.
- Yet, in this work we view SCAs in a **simplified** way as computational experiments (*program*).
- Here, every *execution* of an SCA yields a **binary space-time diagram** which is just one of the possible trajectories in the global state space.
  - An SCA can have **many different space-time diagrams** starting from the **same** initial configuration.

Let the pLUT of an SCA be: [0.88 0.75 0.48 0.30 0.83 0.21 0.30 0.14].

Below we show 5 different **space-time diagrams** of the SCA defined with this pLUT, starting from the same initial configuration.



# Different types of SCAs

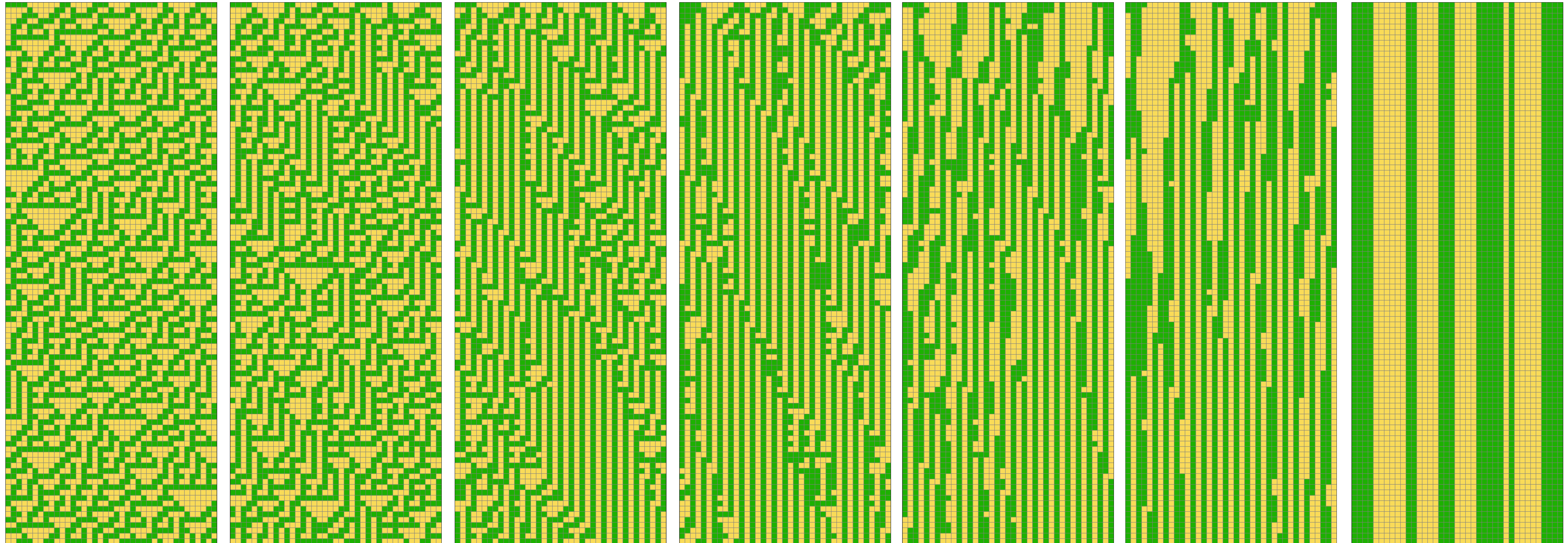
- **$\alpha$ -asynchronous CA** ( $\alpha$ -ACA) - constructed by a single deterministic CA. At every time step, independently for every cell this single CA is either applied with probability  $\alpha$ , or the previous cell state is kept (with probability  $1 - \alpha$ ).
- **Diploid CA** - instead of a single CA (as in  $\alpha$ -ACA), there are two deterministic CAs  $A_1, A_2$ , and at every time step, independently for each cell, a random decision is made which of these two CAs is used for a state update.  $A_1$  is used with prob.  $\lambda$  and  $A_2$  with prob.  $1 - \lambda$ . We write  $(A_1, A_2)_\lambda$ .
- **Triploid CA** - as in Diploid CA, but here we have **three** deterministic CAs...
- ... (*we can continue*) Quadriploid? Pentaploid? ...
- **Stochastic Mixture of CAs** - here we have a finite sequence of deterministic CAs and a corresponding sequence of probabilities (defining a probability distribution). At every time step, independently for every cell, a CA is selected following the given probability distribution. (**This is the most general case - each SCA is a stochastic mixture.**)

# Different types of SCAs

- **$\alpha$ -asynchronous CA** ( $\alpha$ -ACA) - constructed by a single deterministic CA. At every time step, independently for every cell this single CA is either applied with probability  $\alpha$ , or the previous cell state is kept (with probability  $1 - \alpha$ ).
- **Diploid CA** - instead of a single CA (as in  $\alpha$ -ACA), there are two deterministic CAs  $A_1, A_2$ , and at every time step, independently for each cell, a random decision is made which of these two CAs is used for a state update.  $A_1$  is used with prob.  $\lambda$  and  $A_2$  with prob.  $1 - \lambda$ . We write  $(A_1, A_2)_\lambda$ .
- **Triploid CA** - as in Diploid CA, but here we have **three** deterministic CAs...
- ... (*we can continue*) Quadriploid? Pentaploid? ...
- **Stochastic Mixture of CAs** - here we have a finite sequence of deterministic CAs and a corresponding sequence of probabilities (defining a probability distribution). At every time step, independently for every cell, a CA is selected following the given probability distribution. (**This is the most general case - each SCA is a stochastic mixture.**)



Below we show examples of **space-time diagrams** of  $\alpha$ -ACAs generated with ECA 30 for different values of synchrony rate  $\alpha$ . Each space-time diagram starts from the same initial configuration.



$$\alpha = 1$$

$$\alpha = 0.9$$

$$\alpha = 0.75$$

$$\alpha = 0.5$$

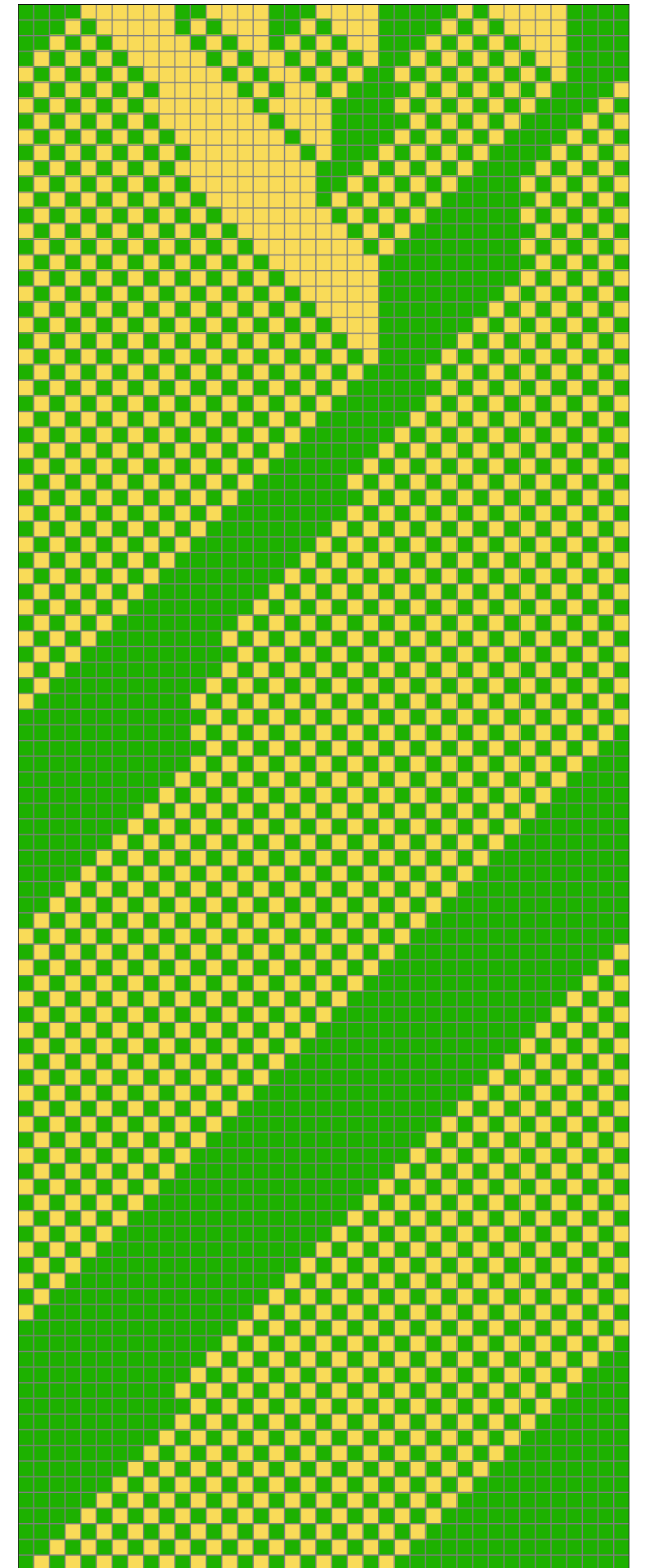
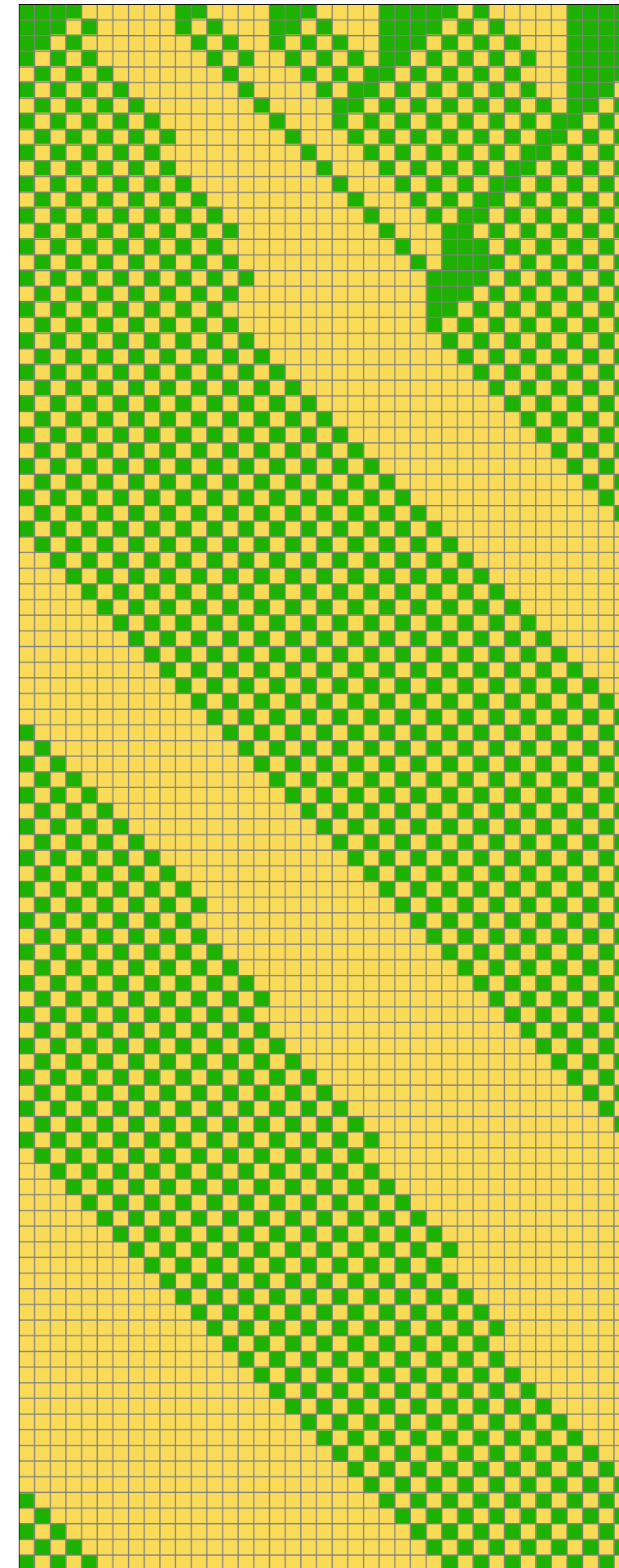
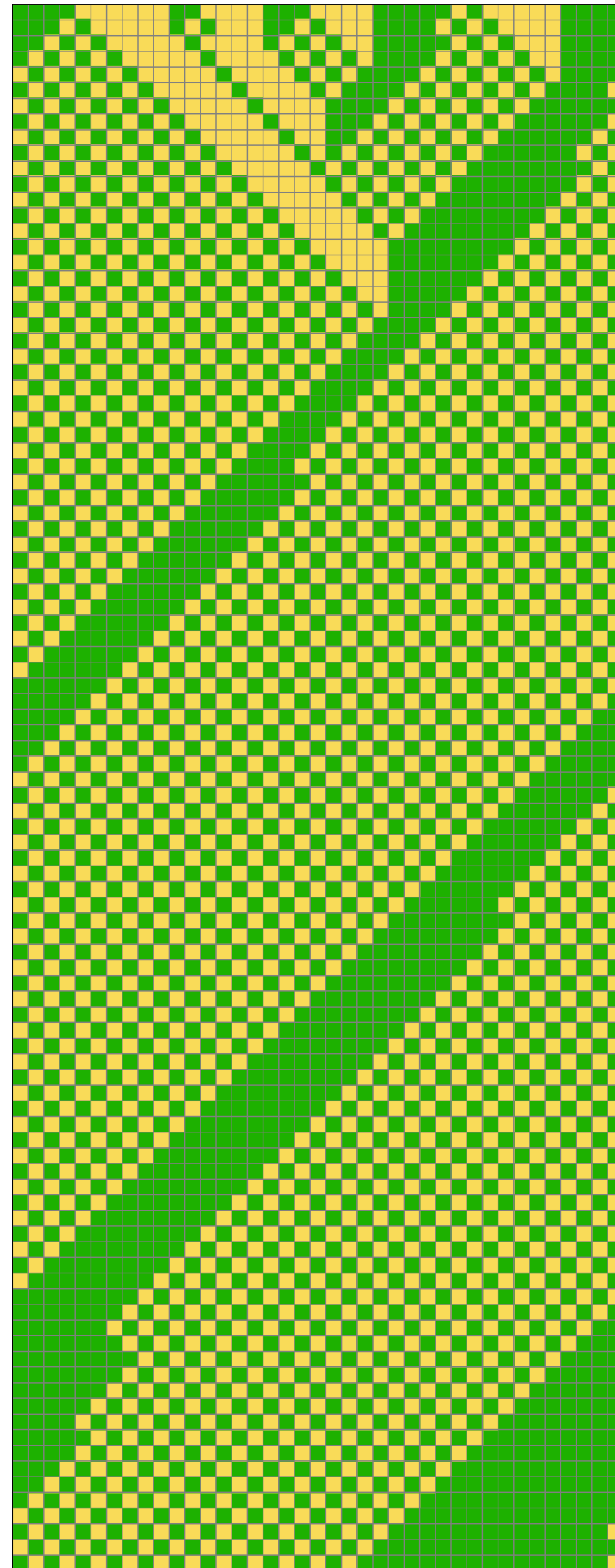
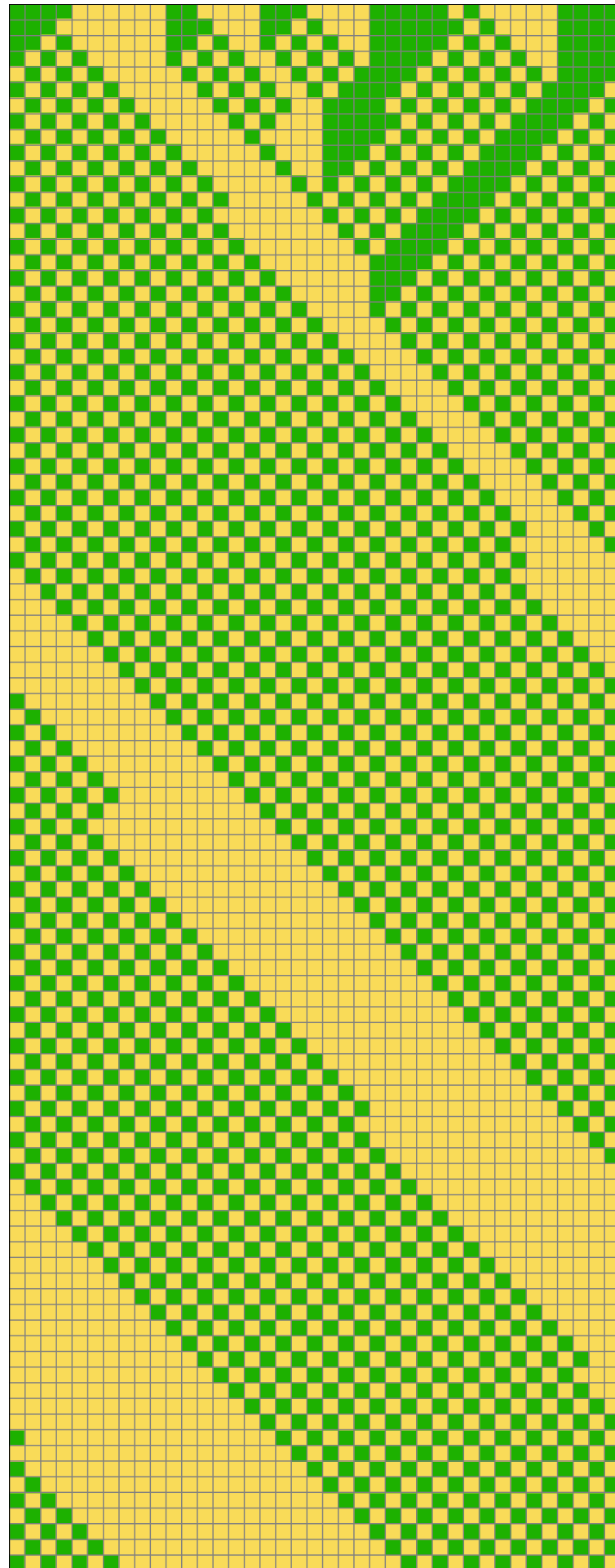
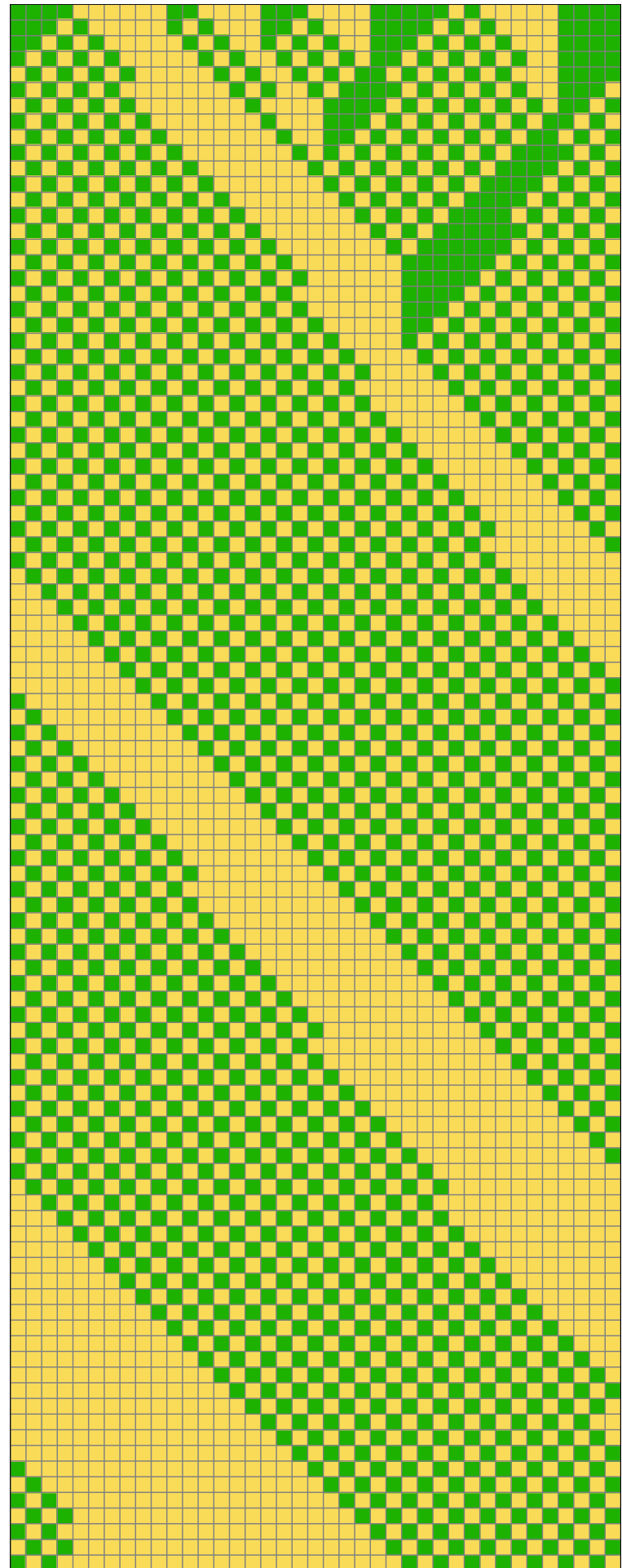
$$\alpha = 0.25$$

$$\alpha = 0.1$$

$$\alpha = 0$$

Let the pLUT of an SCA be given by:  $[1 \quad 0.1 \quad 1 \quad 0.9 \quad 1 \quad 0 \quad 0 \quad 0]$ .

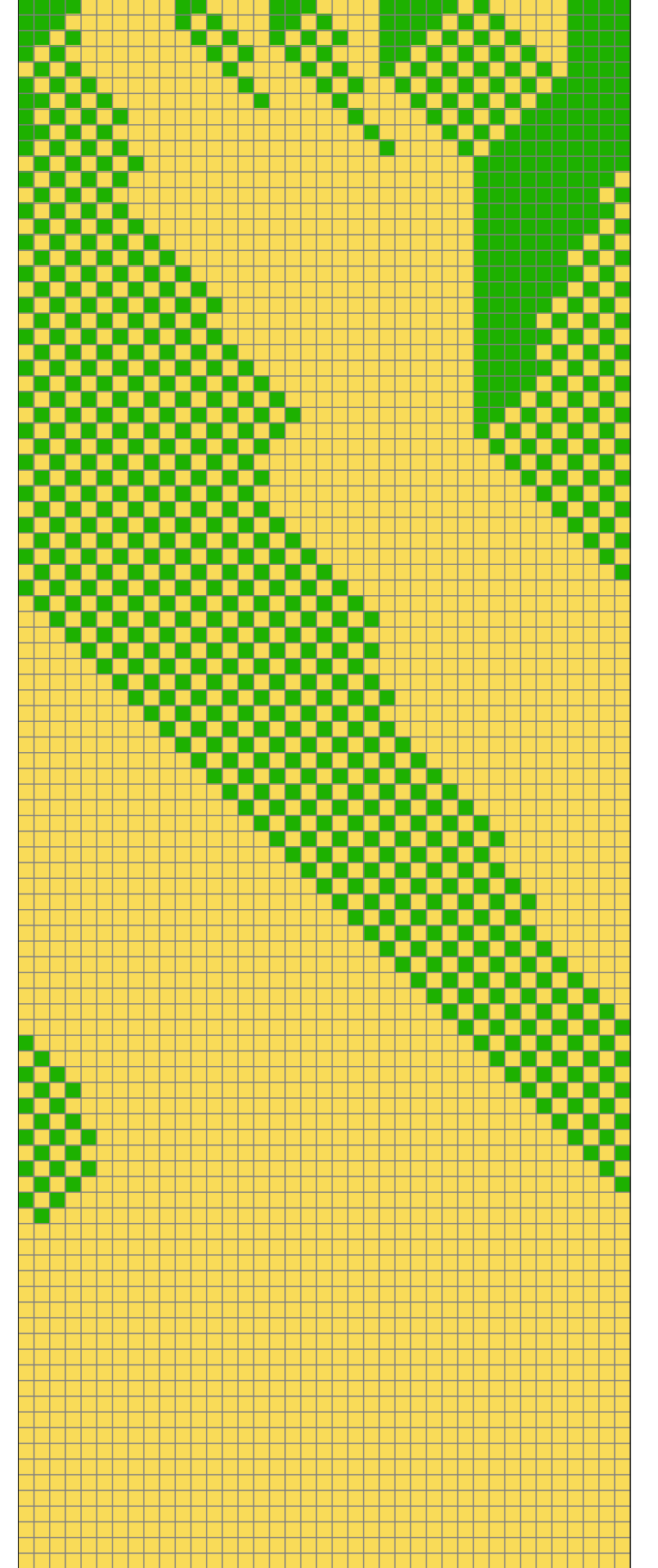
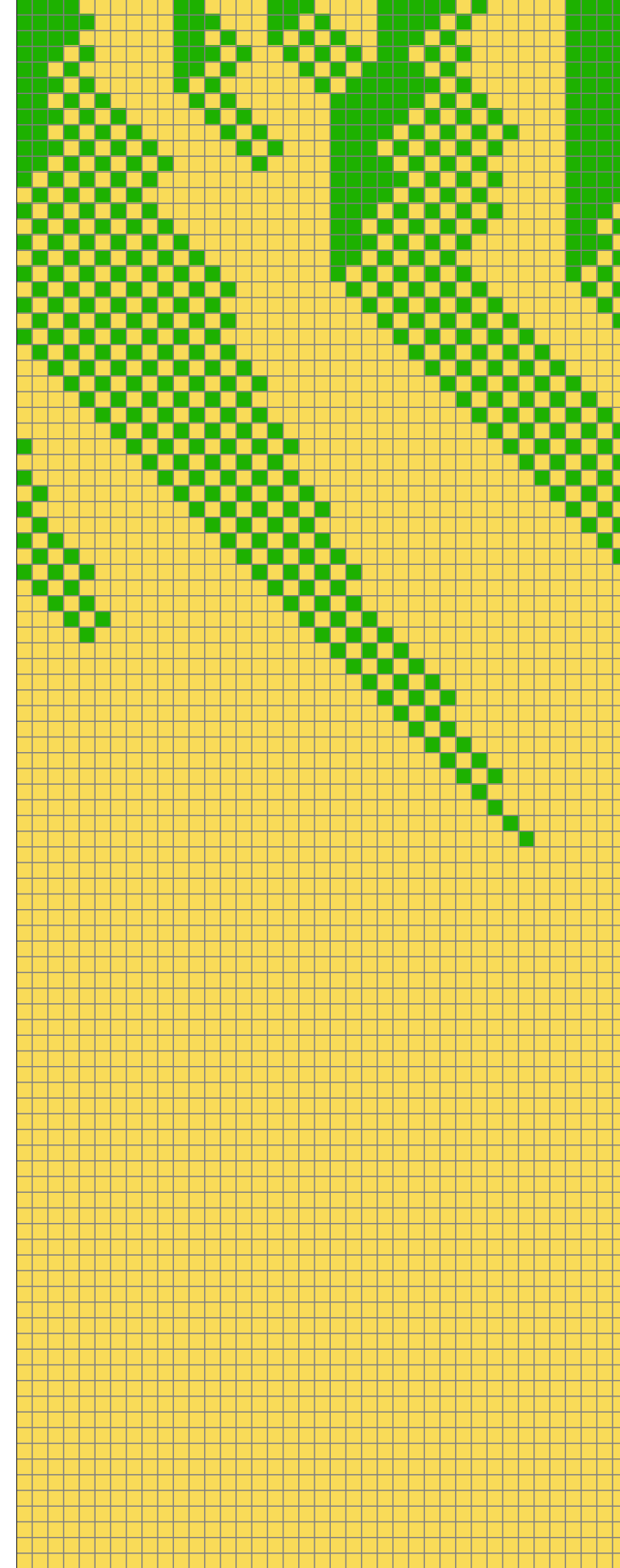
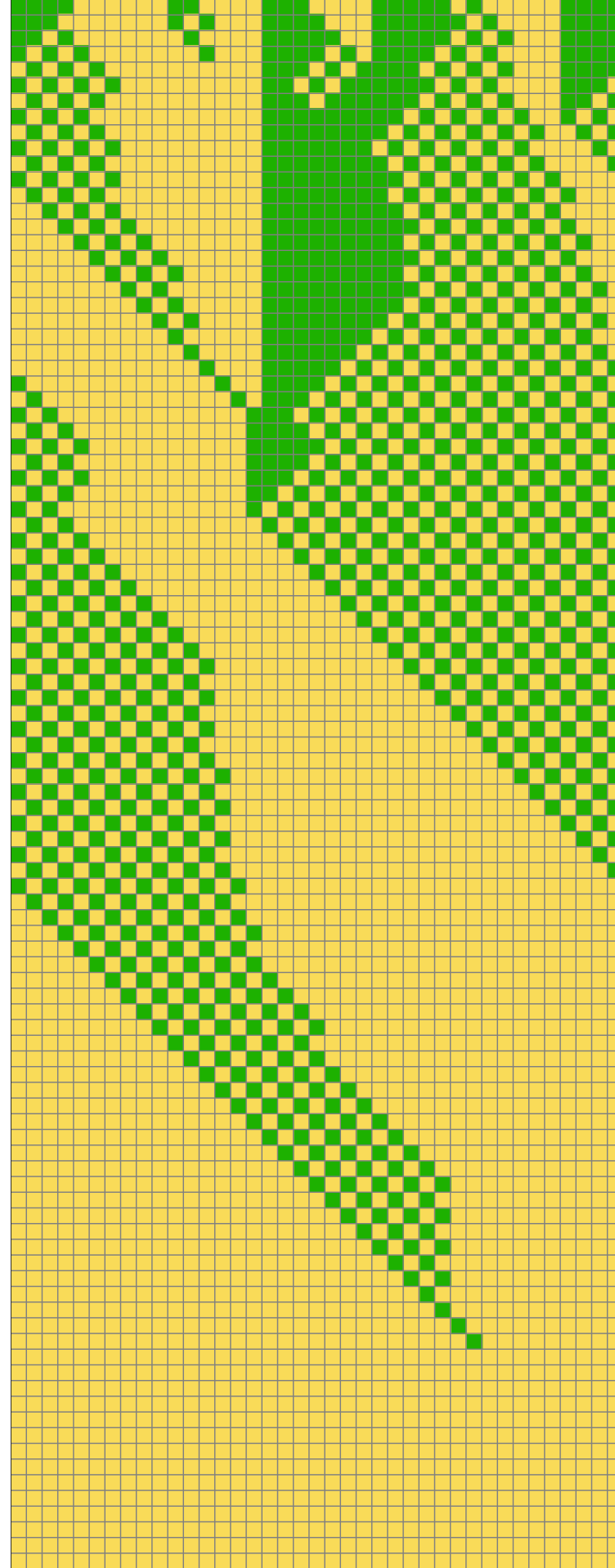
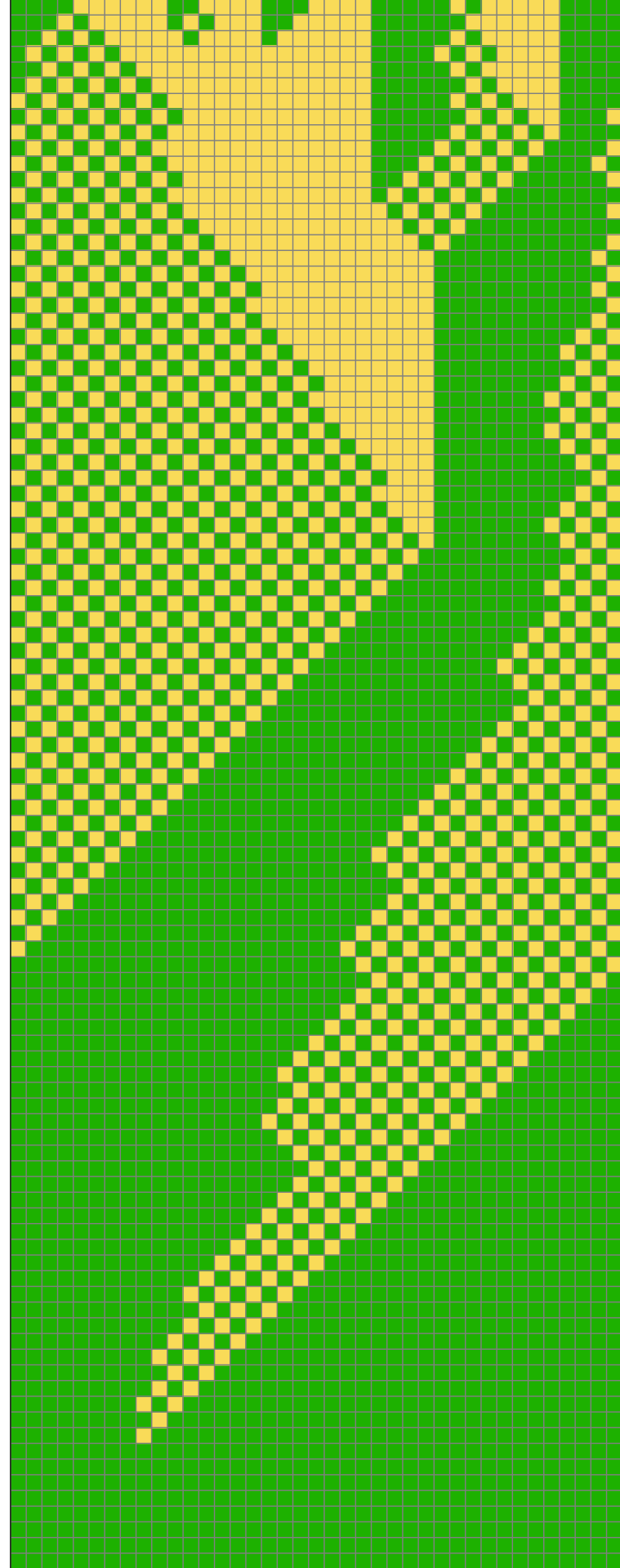
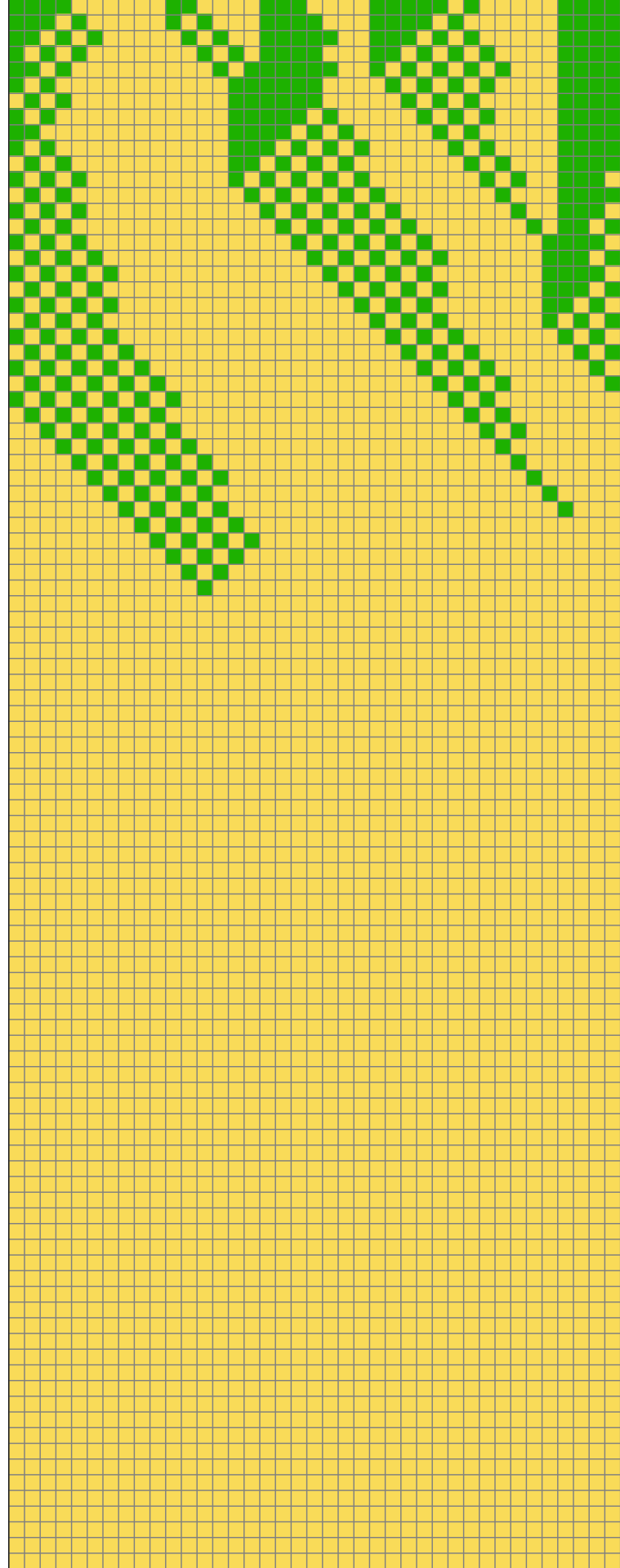
Below we show 5 different **space-time diagrams** of the SCA defined with this pLUT, starting from the same initial configuration.





Let the pLUT of an SCA be given by:  $[1 \quad 0.3 \quad 1 \quad 0.7 \quad 1 \quad 0 \quad 0 \quad 0]$ .

Below we show 5 different **space-time diagrams** of the SCA defined with this pLUT, starting from the same initial configuration.



# Identification of Diploid CAs

- Problem statement: Given a **set of observations** (partial space-time diagrams) of an **unknown**  $(A_1, A_2)_\lambda$ , find:
  - **local rules** for  $A_1, A_2$ , and an **estimate** of  $\lambda$ ;
  - the **most likely** values for the gaps in observations (*i.e.*, fill the gaps).
- Strong assumptions:
  - **No time gaps** and only **isolated** spatial partiality, meaning that two gaps cannot be *close* to each other.
  - Probability  $\lambda$  is separated from 0 and 0.5:  
 $\lambda \in ]a, b[$  where  $a > 0$  and  $b < 0.5$ .
  - **Neighborhood radius:**  $r = 1$ .

# Solution approach: Diploid CAs

- For each observed neighborhood configuration count its occurrences and the cases when the outcome is 1. Based on this calculate the typical **proportion**.
- **Naïve approach:** use these proportions to get estimates for each of the entries of the pLUT. This works assuming that we have observed each of the neighborhood configurations *frequently* enough.
- This naïve approach **ignores** the fact that we know that we are looking for a Diploid CA.
- We improve the naïve approach, by using the fact that in the case of Diploid CAs, there can only be values from the set:  $\{0, \lambda, 1 - \lambda, 1\}$  in the pLUT, and thus **some** neighborhoods can be treated **jointly** not separately!

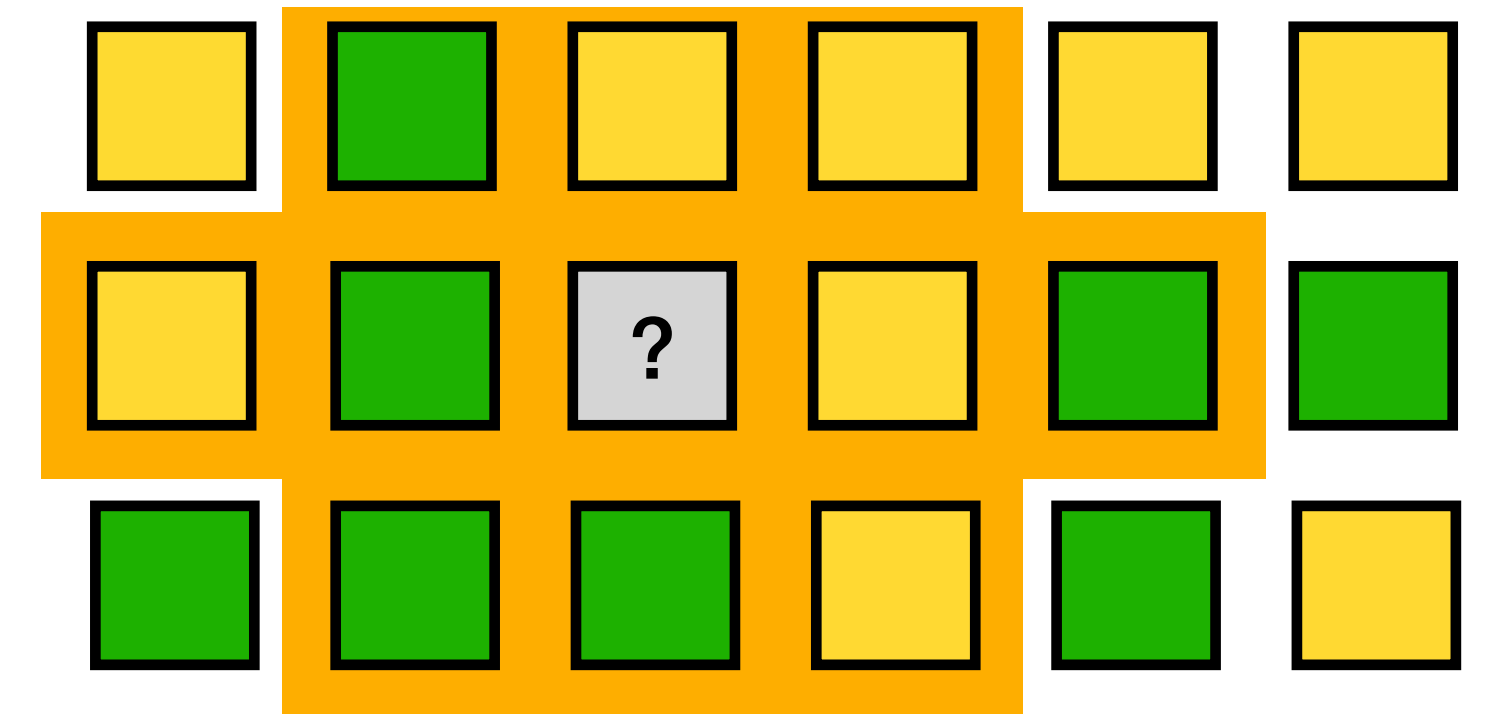
# Gap filling

- We use the point estimate for  $\lambda$  (center of the confidence interval).
- We analyze both the transitions forward and backward in time.
- This allows finding the two probabilities for the two potential states. (*Sometimes* one of these two probabilities is 0.)
- We pick the state for which the probability is higher.

$t - 1$

$t$

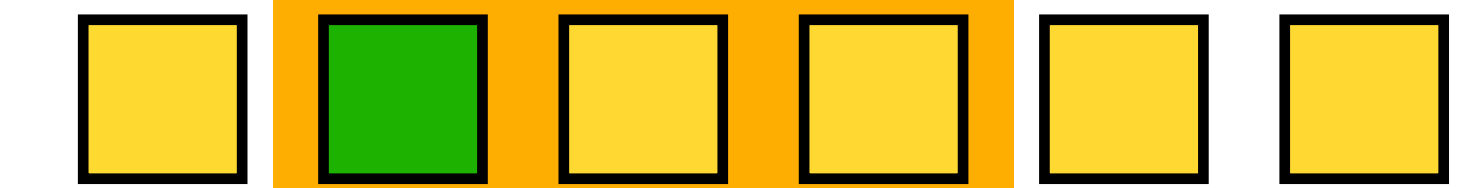
$t + 1$



# Gap filling

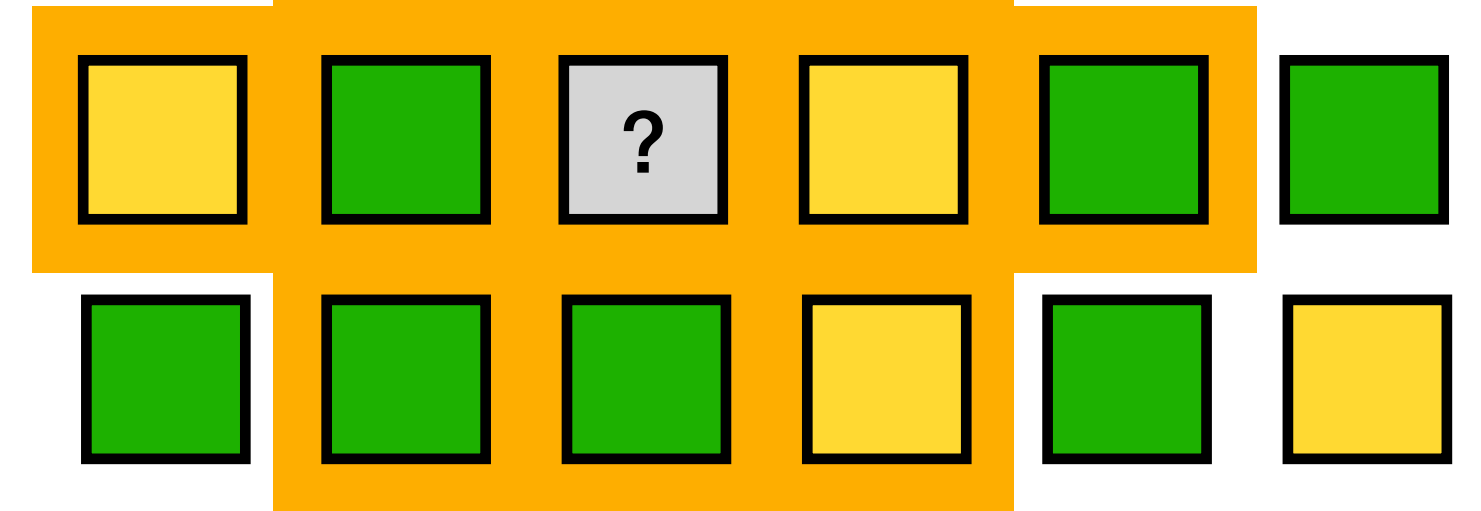
- We use the point estimate for  $\lambda$  (center of the confidence interval).
- We analyze both the transitions forward and backward in time.
- This allows finding the two probabilities for the two potential states. (*Sometimes* one of these two probabilities is 0.)
- We pick the state for which the probability is higher.

$t - 1$

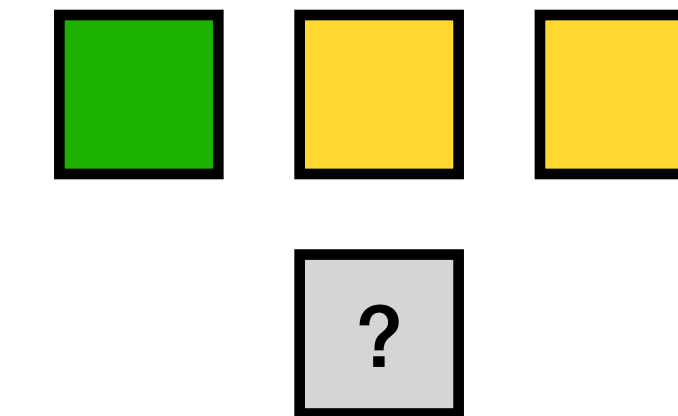


$t$

$t + 1$



$t - 1 \rightarrow t :$



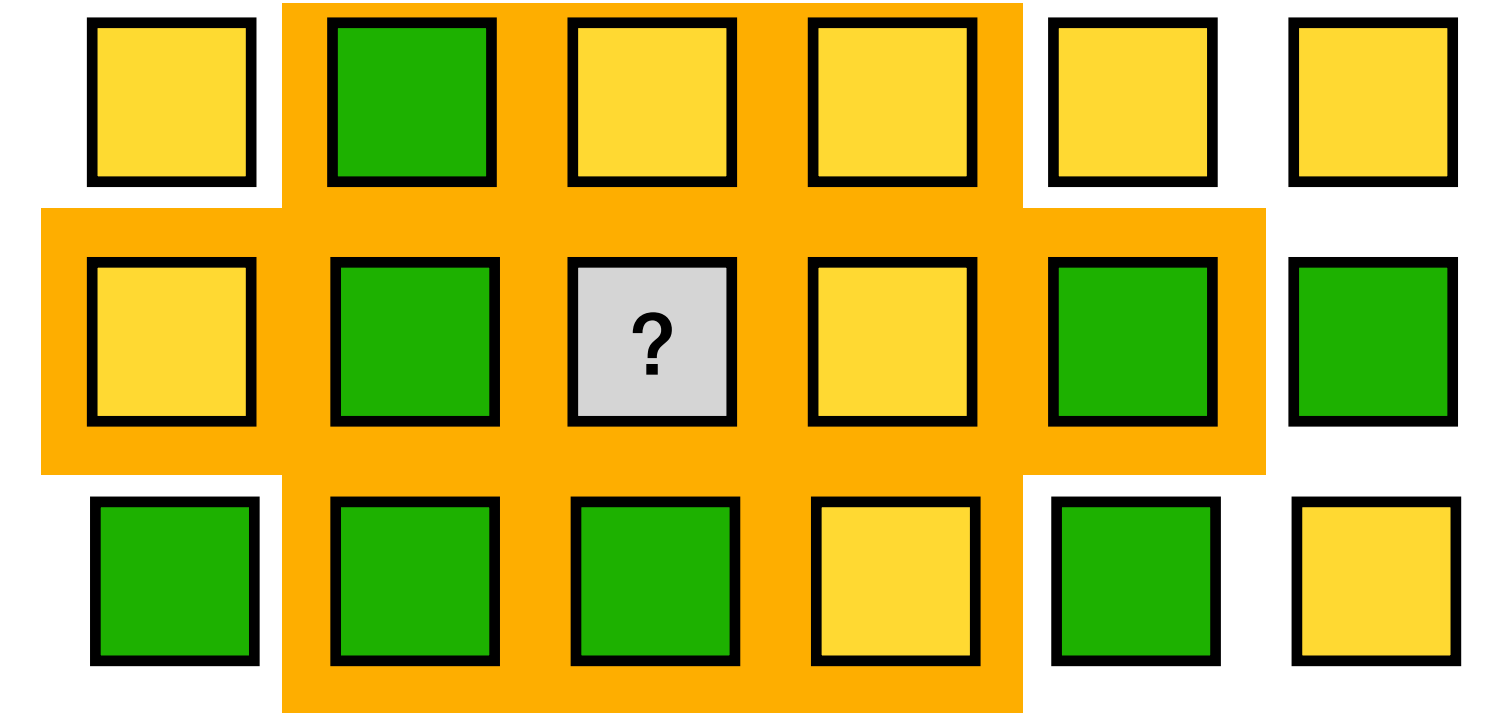
# Gap filling

- We use the point estimate for  $\lambda$  (center of the confidence interval).
- We analyze both the transitions forward and backward in time.
- This allows finding the two probabilities for the two potential states. (Sometimes one of these two probabilities is 0.)
- We pick the state for which the probability is higher.

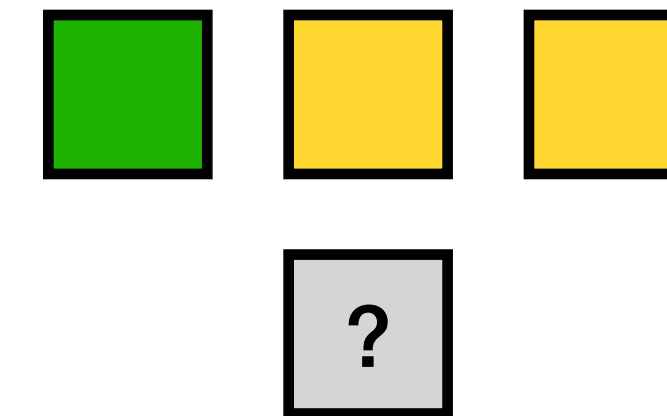
$t - 1$

$t$

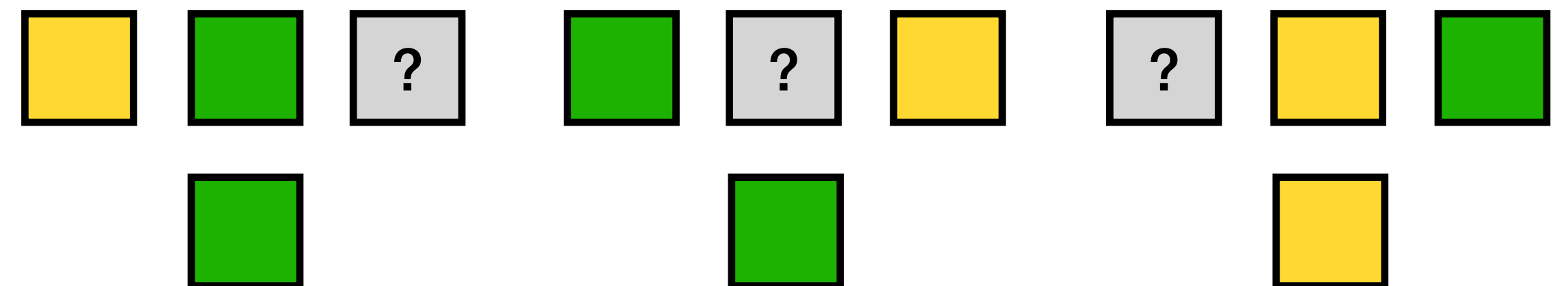
$t + 1$



$t - 1 \rightarrow t :$



$t \leftarrow t + 1 :$



# Implementation

- The computation cost of the identification algorithm for Diploid CAs is significantly lower compared to the deterministic CAs case, all simulations were implemented in **Python** using popular scientific libraries (numpy, scipy).
- The experiments were executed on a **HPC** cluster - jobs managed by **Python** and *shell* scripts.
- Data post-processing: **Python**
- Source code available on **GitHub**



# Experiments

Paper	Experiment	Results
[B2]	<b><math>\alpha</math>-ACAs constructed with ECAs</b> (255 cases, each with different values of $\alpha$ )	<p>The ECAs used to build the <math>\alpha</math>-ACA were always correctly identified.</p> <p>On average the relative error of point estimate for <math>\alpha</math> was <b>0.51%</b>. Moreover, for more than <b>80%</b> of the cases the relative error was lower than <b>1%</b>.</p> <p>The average success rate of gap filling was <b>96%</b>.</p>
[B3-B4]	<b>Diploid CAs constructed with ECAs</b> (256 x 255 cases for $\lambda = 0.1, 0.2, 0.3, 0.4$ )	<p>The deterministic components of Diploid CAs were always correctly identified.</p> <p>The average maximum relative error for <math>\lambda</math>-point estimate was <b>2.33%</b>.</p> <p>The average success rate of gap filling was <b>95.74%</b>.</p>

A number of connections between the dynamical properties of ECAs and the **accuracy** of estimates have been uncovered. **In general, more “complex” rules were easier to identify.**  
Yet, the number of differing entries in the LUTs of Diploid’s components played the most important role in determining the identification accuracy.



# Summary

# Summary

- The **identification problem** has been presented and solved for two classes of CAs:
  - 1D, binary, deterministic CAs,
  - 1D, binary Diploid CAs.
- In both cases the identification considered **partial observations** with state information missing for *some* of the cells. In the deterministic case, also **entire time steps** were missing.
- The proposed solutions turned out to be *effective*, providing *good* results

# Future work

- **Noisy observations** - some observed states are wrong, i.e., we observed a flipped state.
- Identification of **broader classes of CAs**:
  - **Affine Continuous CAs (ACCA)** - usage of Differential Evolution over the space of rules parametrized by a real-valued vectors similar to LUTs.
  - **Higher-dimensions and / or bigger state sets** - the formulation is straight forward, yet the performance needs to be evaluated.
  - **Triploids, ..., Stochastic Mixtures** - assessment of the gap filling procedure.
- **Usage of Machine Learning** - training a general CA identification network over a big dataset of observations. Potential for CNNs or RNNs (and similar) models.

# What did I / we learn from this work?

- **Cellular Automata** are:
  - a very important tool for **understanding** the *world*;
  - an empirical proof that (radically) simple rules applied on a “big” scale produce behavior which *seems more complex* than *anything* “designed by hand”.
- **Determinism** (of CAs) is a **very strong** assumption. If the rules of the “universe” are deterministic, uncovering them is significantly easier, even with **imperfect observations**.
- **Evolutionary algorithms** are surprisingly powerful and mostly overlooked these days (due to ML popularity). Early experiments with ML-based CA identification show that the evolutionary path may be “better”.
- **Thread safety** is very hard in software engineering. Couple months of work on the implementation of the identification algorithm were spent on debugging a very subtle bug. It turned out to be a thread unsafe random numbers generator, which under high load stopped being “random”. (The fix for it was ~3 lines of code.)

# **What did I learn from this work?**

# What did I learn from this work?

- **Hard work** and **patience** pays off. My PhD has been severely delayed. But due to this project and during this time a lot of **significant, positive** and **important** “side effects” emerged, which allowed me already to be proud and satisfied.

# What did I learn from this work?

- **Hard work** and **patience** pays off. My PhD has been severely delayed. But due to this project and during this time a lot of **significant, positive** and **important** “side effects” emerged, which allowed me already to be proud and satisfied.
- I have so many good people around me, supporting me!
  - **This day would not be possible without you all** ❤️

Thank you  
Hartelijk bedankt  
Merci beaucoup  
Dziękuję bardzo