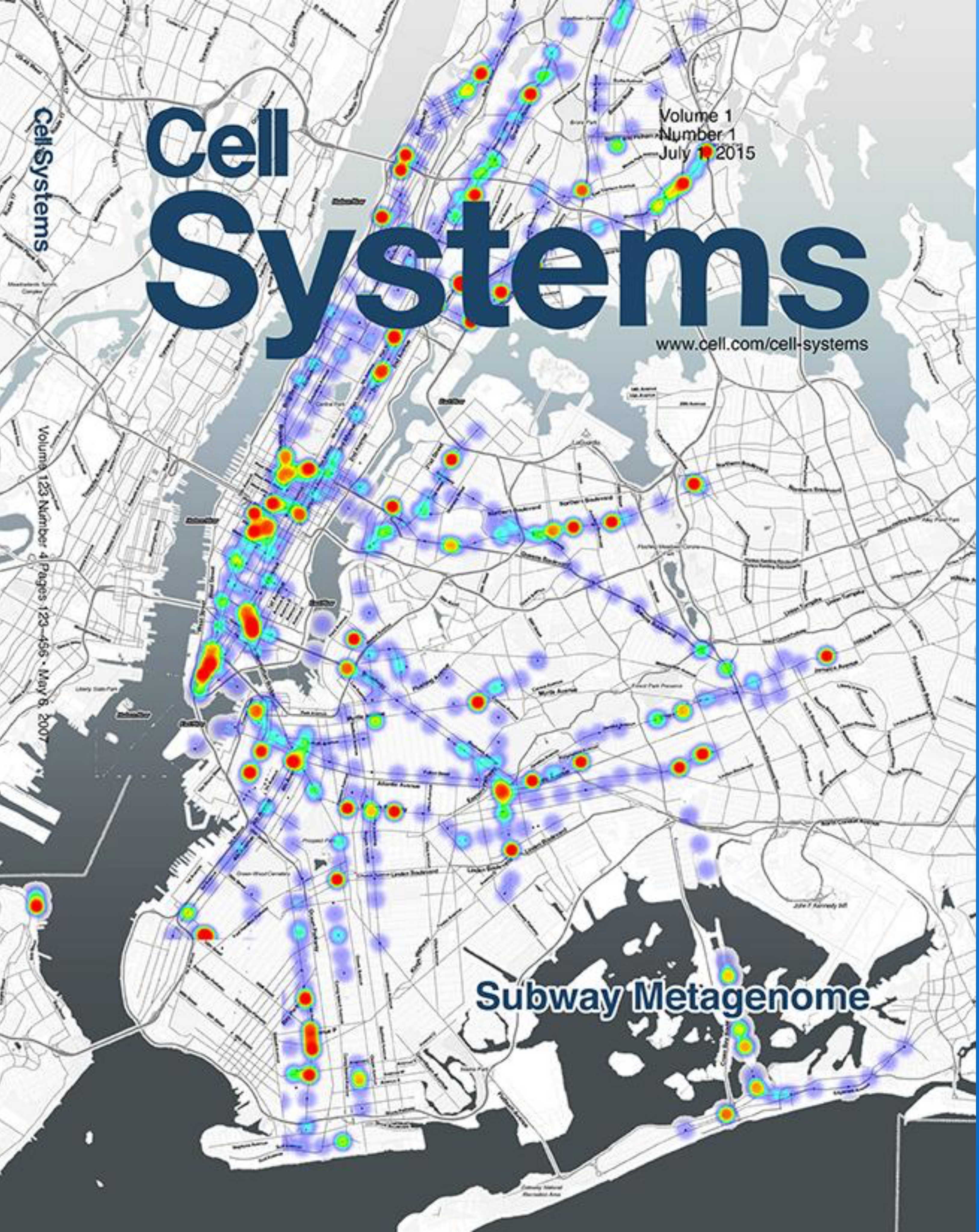
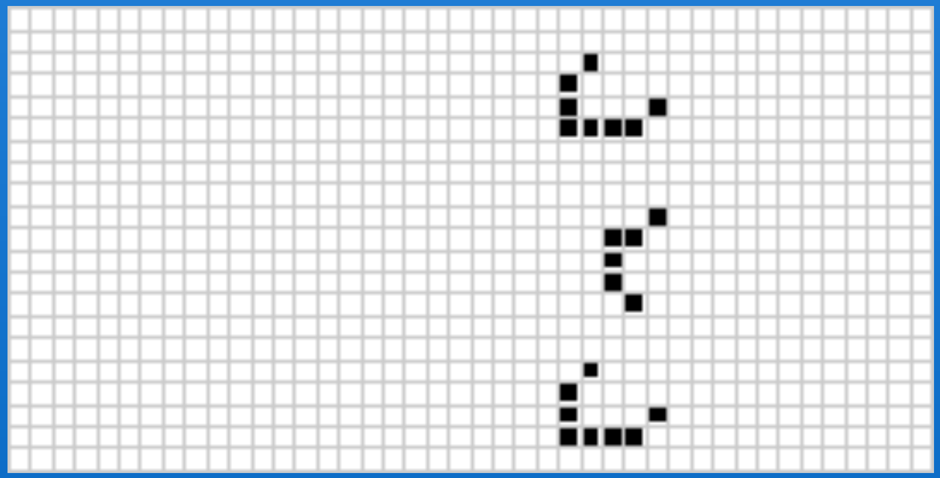
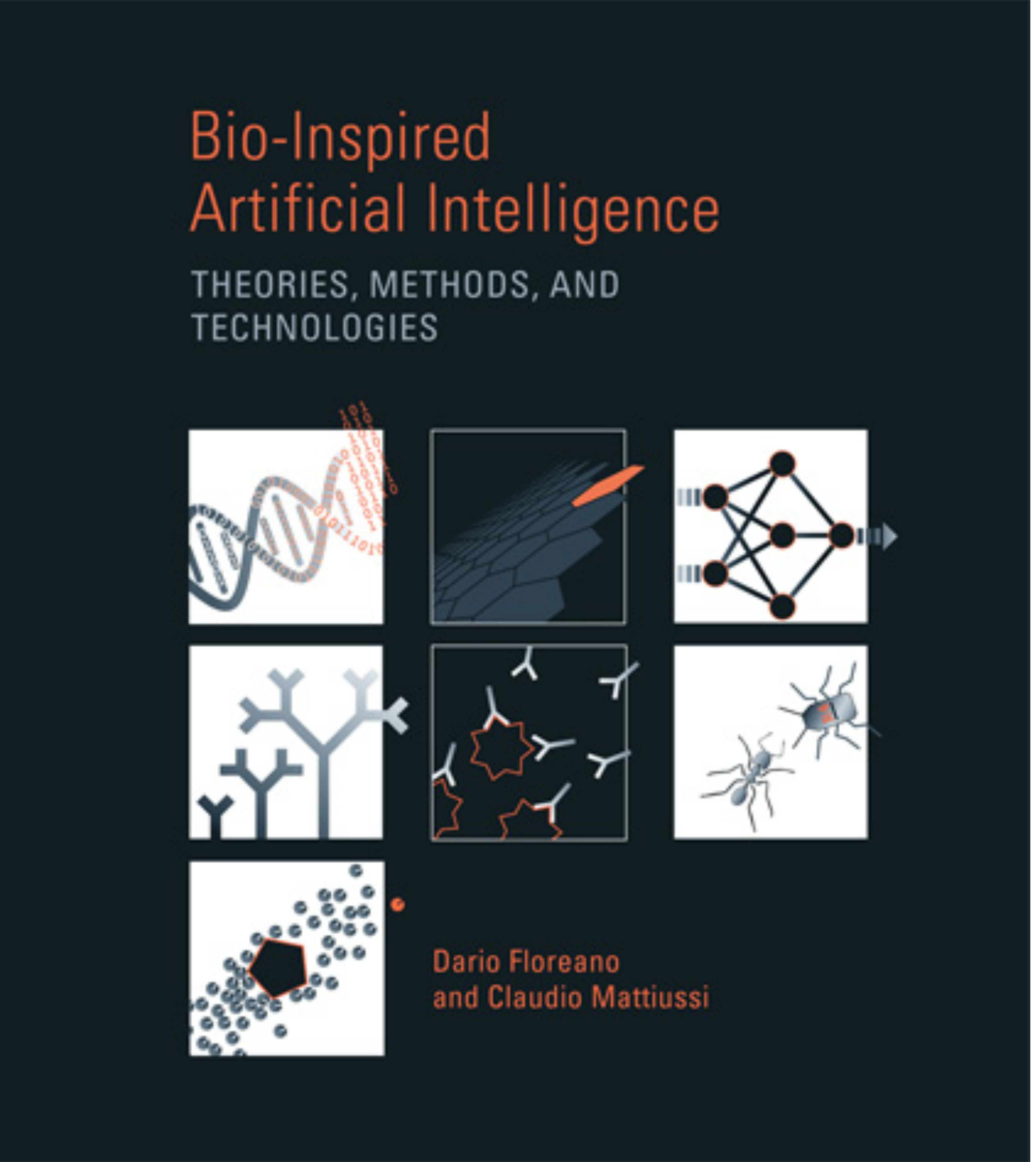


Cellular System

Lecture 2

Feb. 04, 2019

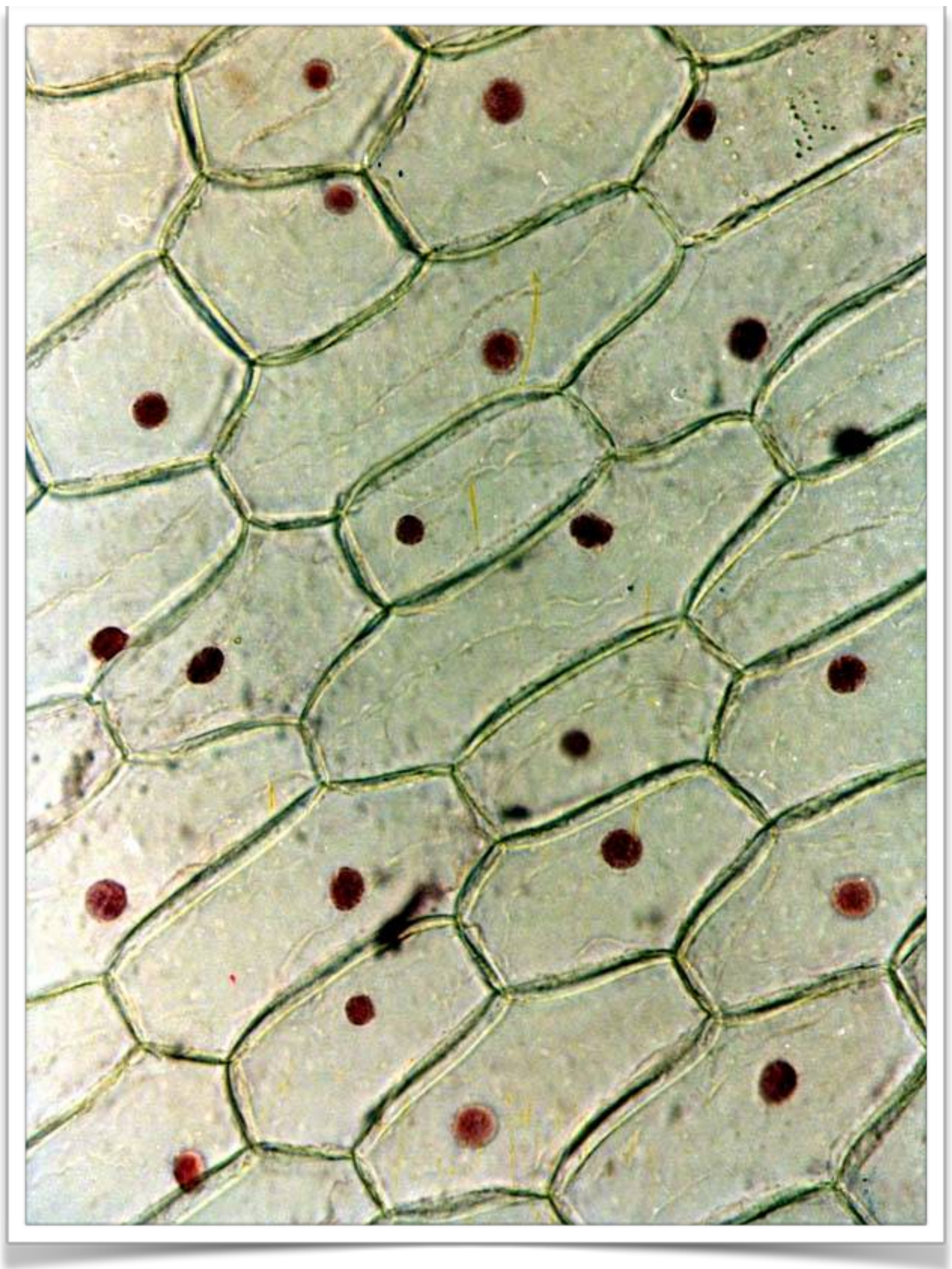




Slides in part based on the companion slides for the book "**Bio-inspired Artificial Intelligence: Theories Methods, and Technologies**", by Dario Floreano and Claudio Mattiussi, MIT Press

Cellular Systems

Evolution has rediscovered several times **multicellularity** as a way to **build complex** living systems



- Multicellular systems are composed by many copies of a unique **fundamental unit** - the cell
- The **local interaction** between cells ("communication") to build something bigger
- influences the **fate** and the **behaviour** of each cell
- The result is an **heterogeneous system** composed by **differentiated cells** that act as specialised units, even if they all contain the **same genetic material** and have essentially the **same structure**

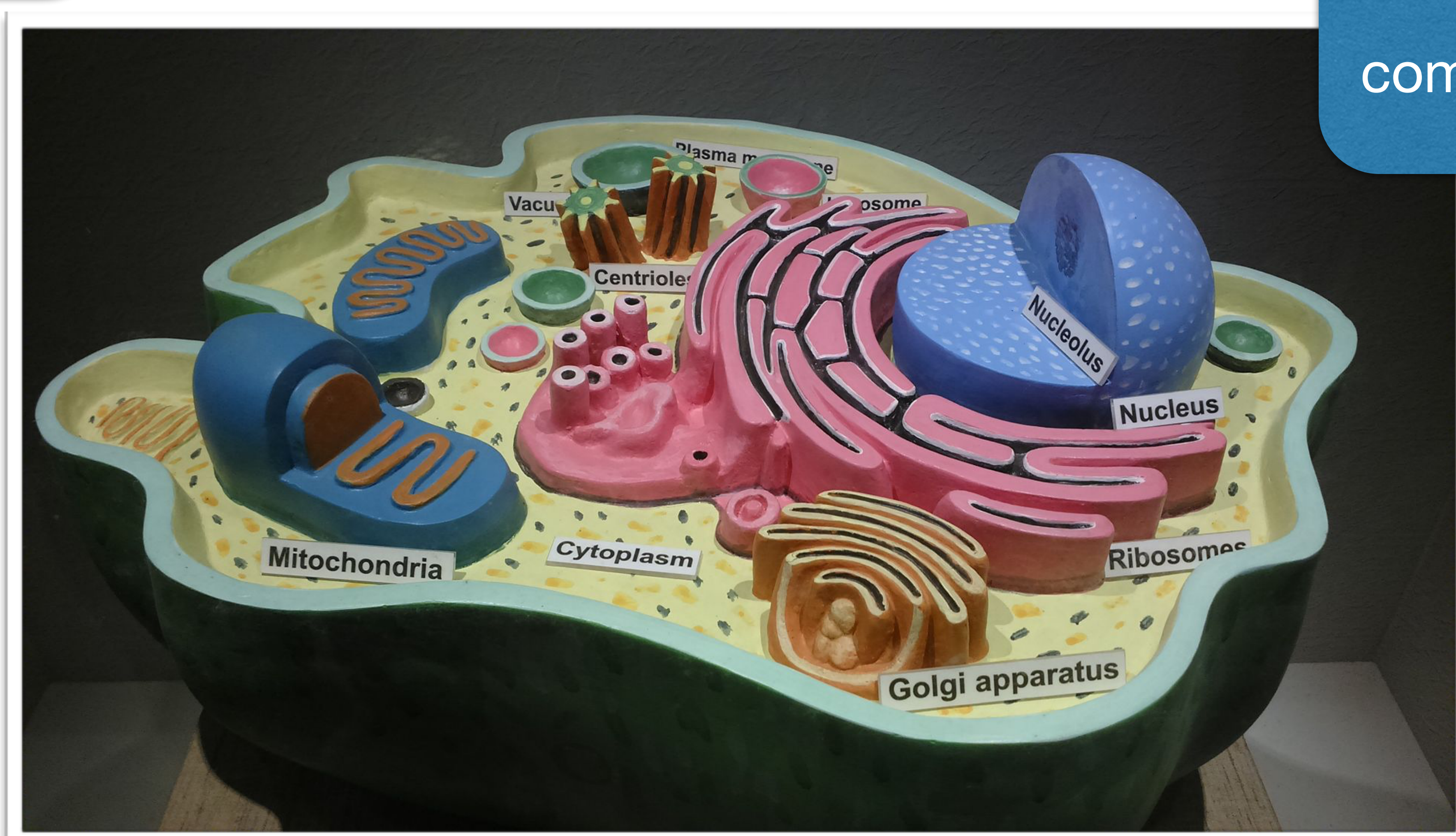
The Cell – A Closer Look

- cells are very complex (eukaryotic)
- 10 trillion cells (10^13) in our body
- basic building blocks (basic structures)
- about 200 different cell types in humans
- sensitive about their environment (cell signalling)

same basic building blocks

different states

local communication



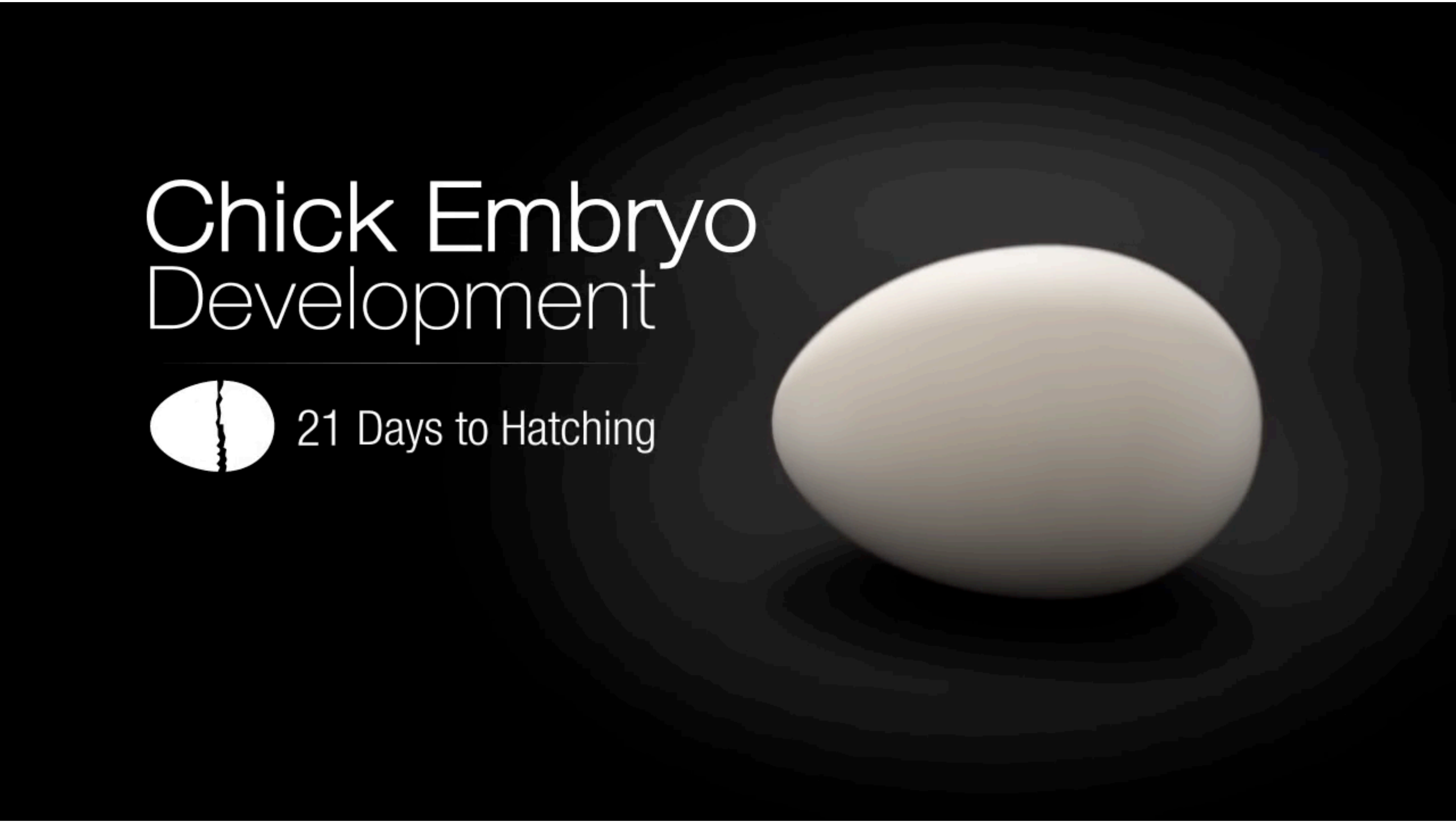
Emergence of complexity

Many **complex phenomena** are the result of the collective dynamics of a **very large number of parts obeying simple rules.**

Development of chicken embryo

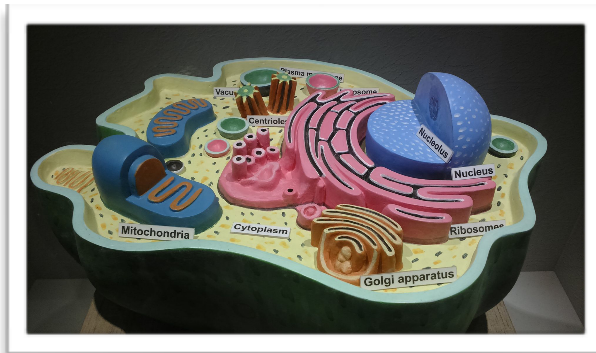
Unexpected global behaviours and patterns can emerge from the interaction of many simple systems that "communicate" only locally.

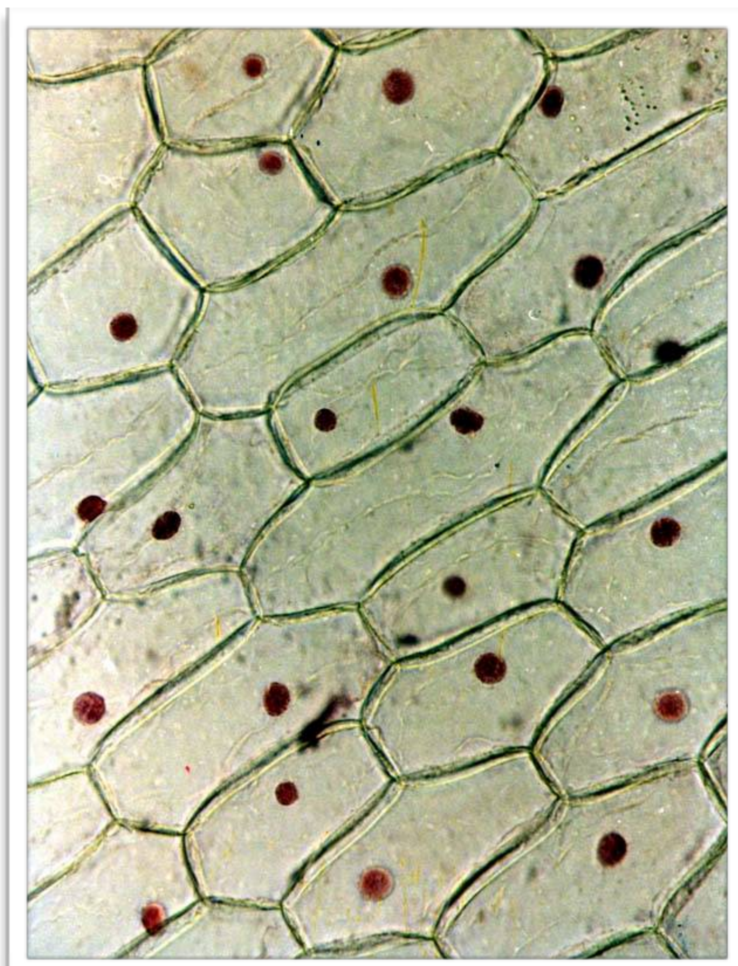
Emergence of complexity



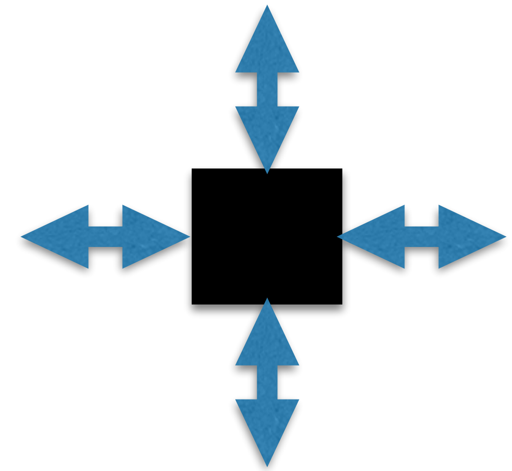
Another video of a salamander: https://www.youtube.com/watch?v=ABGZ-FKjC6c

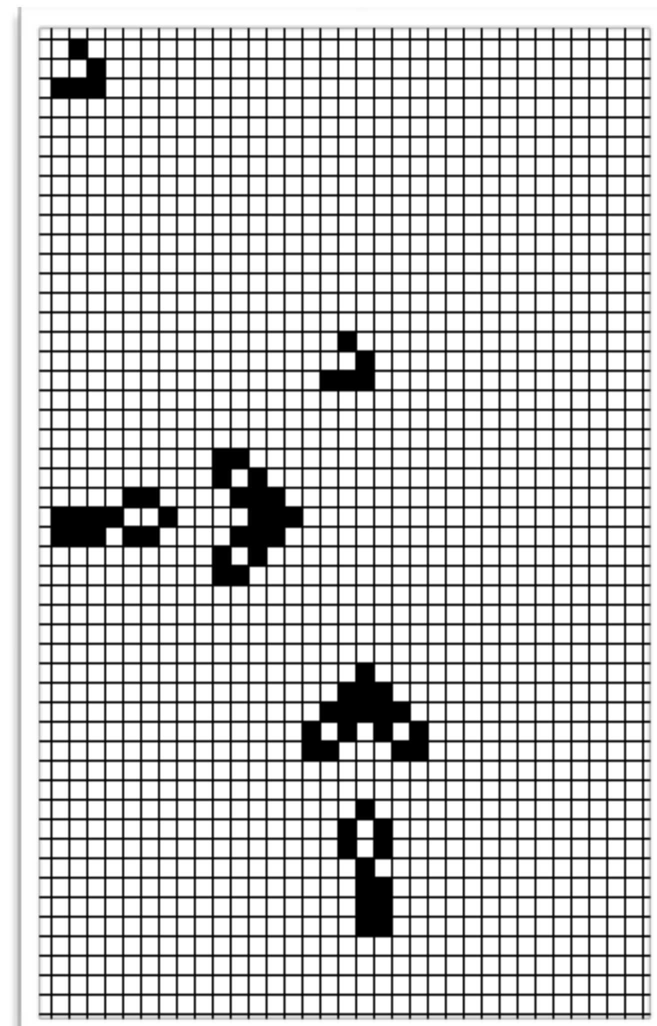
Abstraction





Abstraction





Understanding cell systems

“Behaving” like cell systems

Applications to engineering

Underlying concept
"many simple systems with (geometrically structured) local interaction"

Artificial Life and Evolutionary Experiments, where it allows the definition of arbitrary "synthetic universes" – Theoretical Biology.

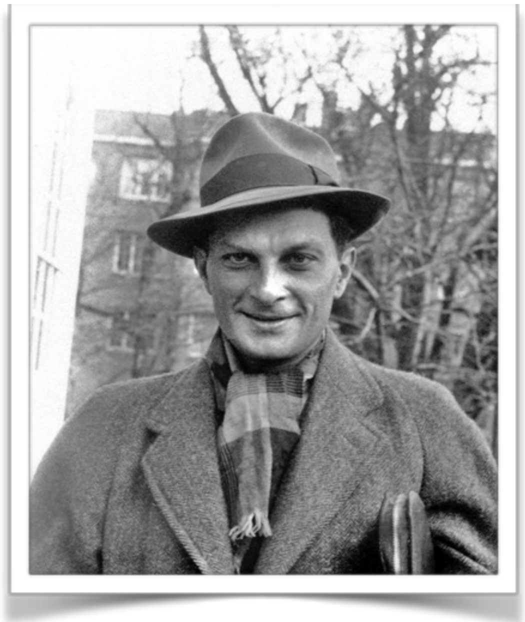
Computer Science and Technology for the implementation of parallel computing engines and the study of the rules of emergent computation.

Physics for the modelling and simulation of complex physical systems and phenomena, and research on the rules of structure and pattern formation.

- More generally, the study of complex systems, i.e., systems composed by many simple units that interact non-linearly

Modelling Social Phenomena voting behaviour, rumours, opinion changes, emerging city structures, etc.

Very short history



Stanislaw Ulam

John von Neumann

Los Alamos National Laboratory
in the 1940s

- **Motivation:** Building self-replicating systems (machine replication, artificial life)
- Lattice network (Ulam) and 2D grid (Neumann)
- von Neumann universal constructor (self-replicating system)
- Later in the 80's Stephen Wolfram (Elementary cellular automata)

Building blocks of cellular systems

We want to define the **simplest nontrivial model of a cellular system**.
We base our model on the following concepts:

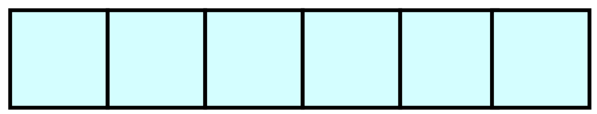
1. **Cell and cellular space**
2. **Neighbourhood (local interaction)**
3. **Cell state**
4. **Transition rule**

We do **not model all the details** and characteristics of biological multicellular organisms but we obtain **simple models** where many interesting phenomena can still be observed

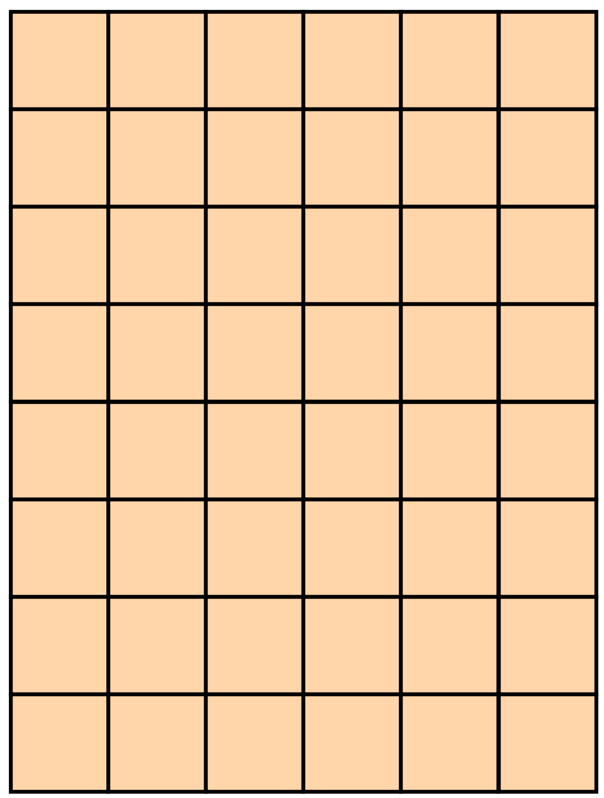
- There are many kinds of cellular system models based on these concepts
- The simplest model is called **Cellular Automaton (CA)**

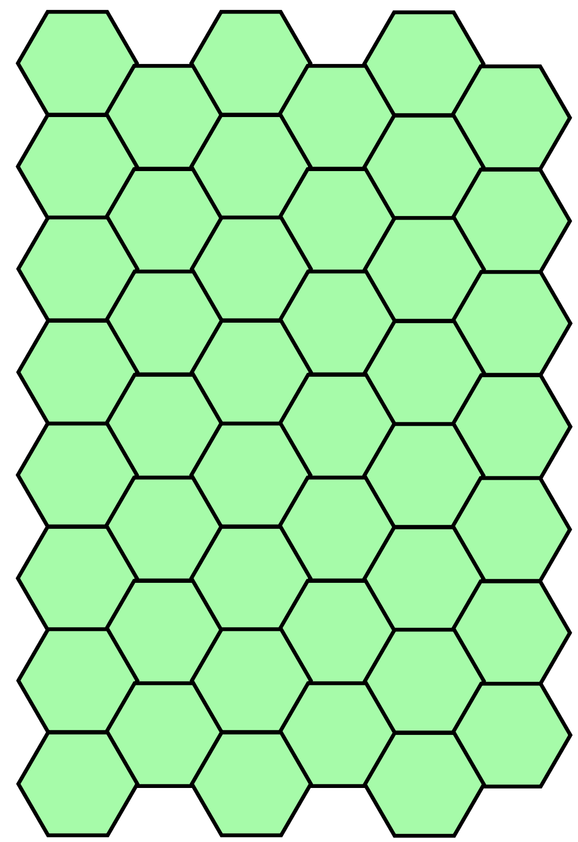
Cellular space

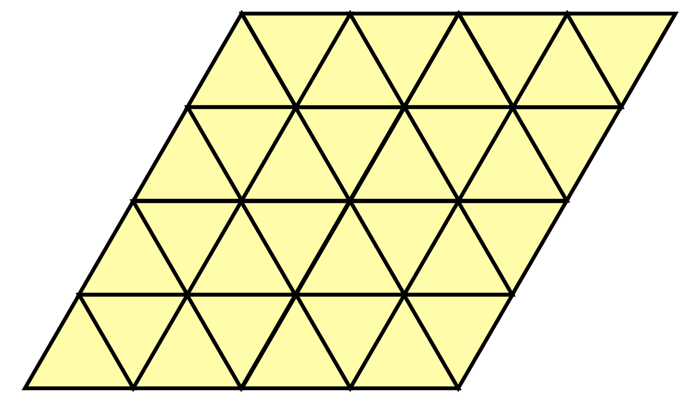
1D



2D

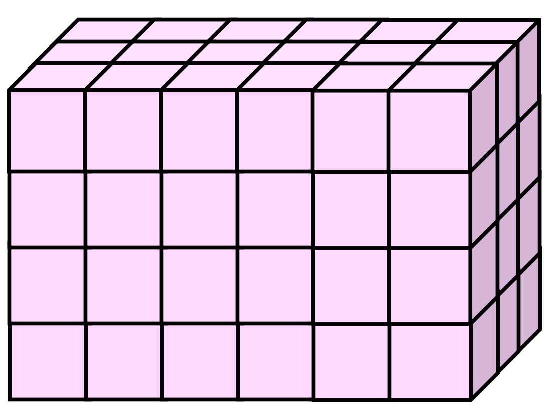






...

3D

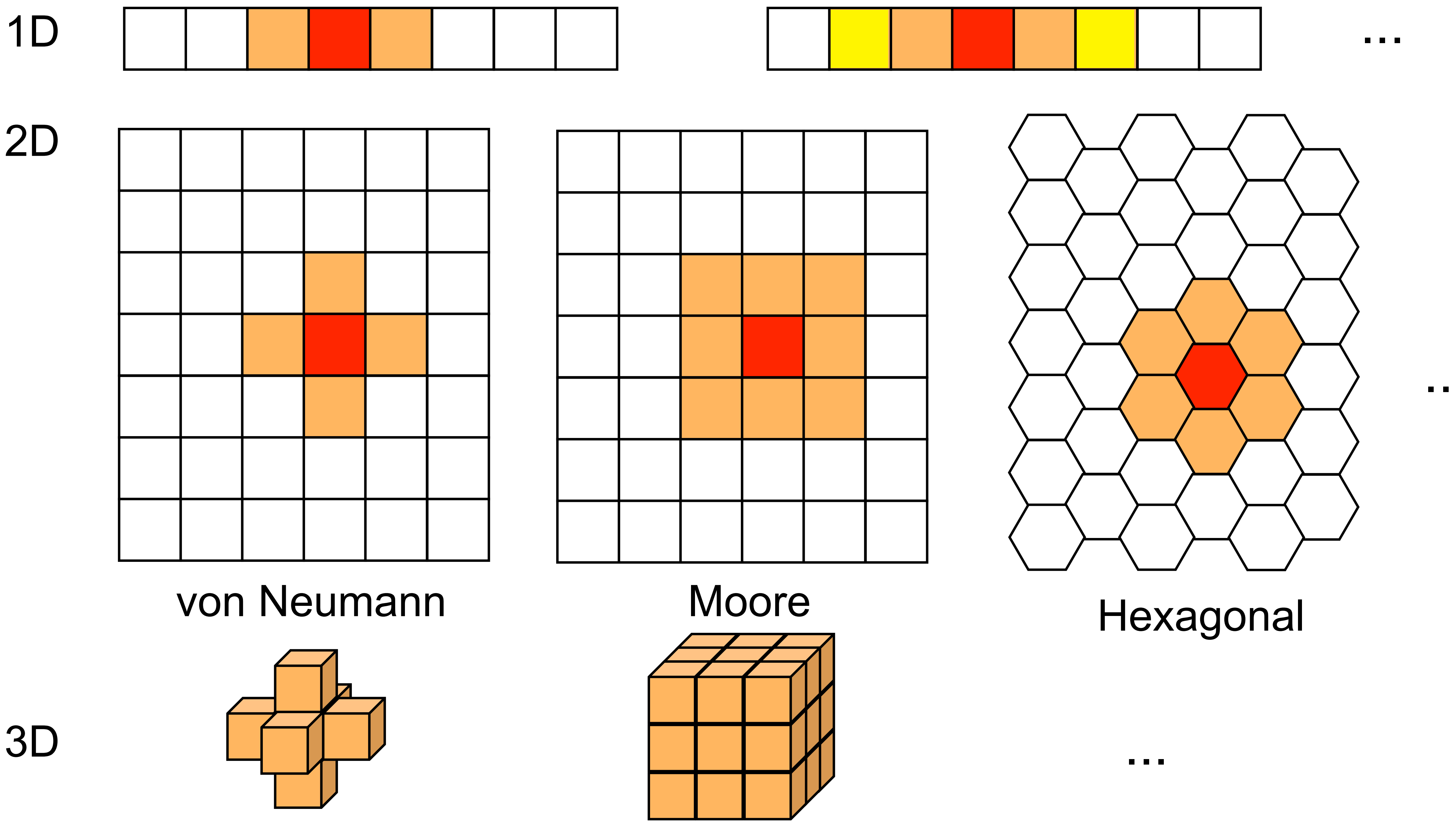


...

and beyond...

Neighbourhood

Informally, it is the set of cells that can influence *directly* a given cell
In *homogeneous* cellular models it has the same shape for all cells



State set and transition rule

Each cell has a state, often represented a a number. The value of the state is often represented by cell colours. There can be a special **quiescent state** s_0.   kwɪˈɛs(ə)nt

$$S = \{s_0, \ldots, s_{k-1}\}$$
$$= \{0, \ldots, k-1\}$$
$$= \{\bullet, \ldots, \bullet\}$$

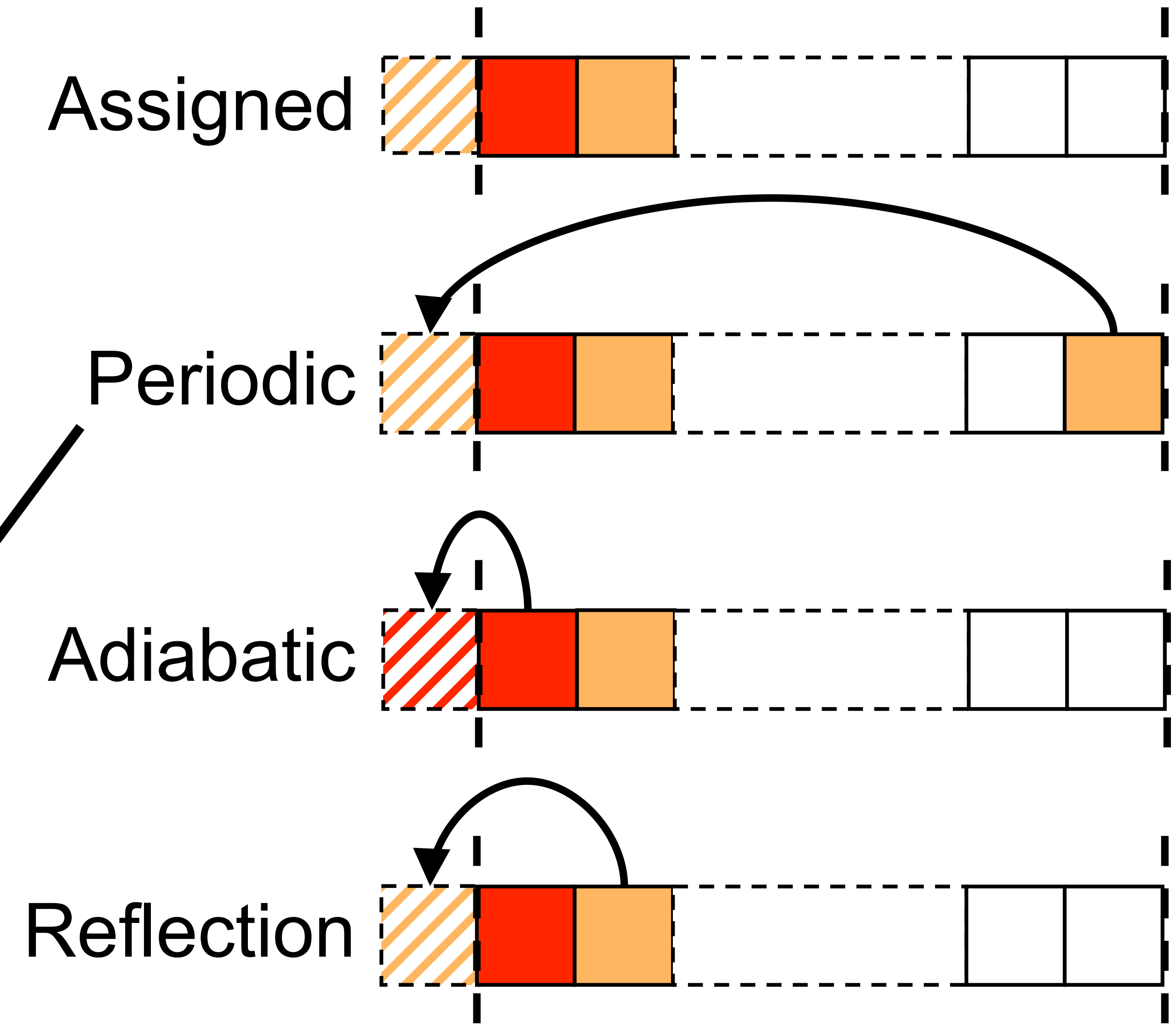
The **transition rule** is the fundamental element of the CA. It must specify the new state corresponding to each possible configuration of states of the cells in the neighbourhood.

The transition rule can be represented as a **transition table**, although this becomes rapidly impractical.

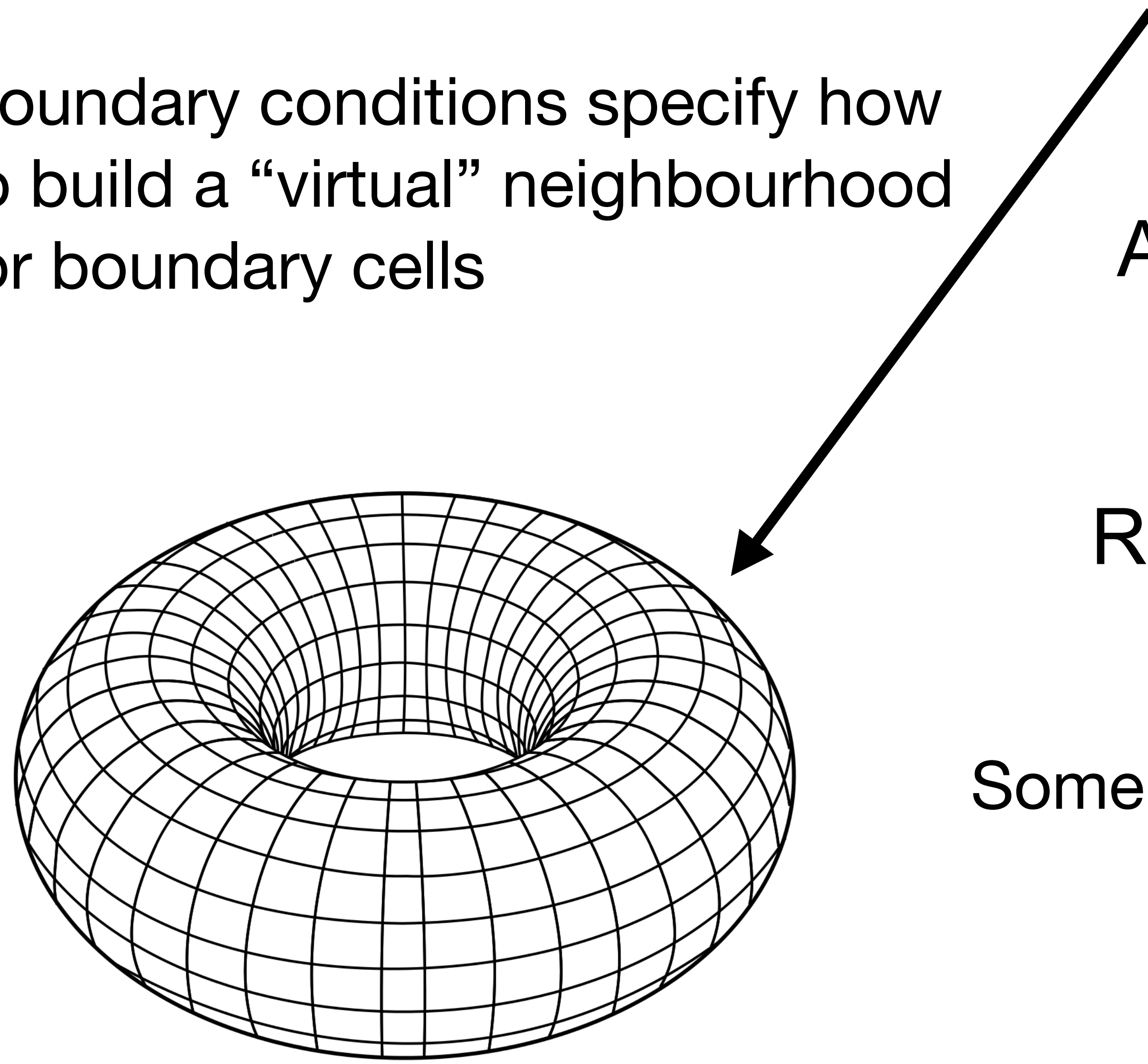
Boundary conditions

If the cellular space has a boundary, cells on the boundary may lack the cells required to form the prescribed neighbourhood

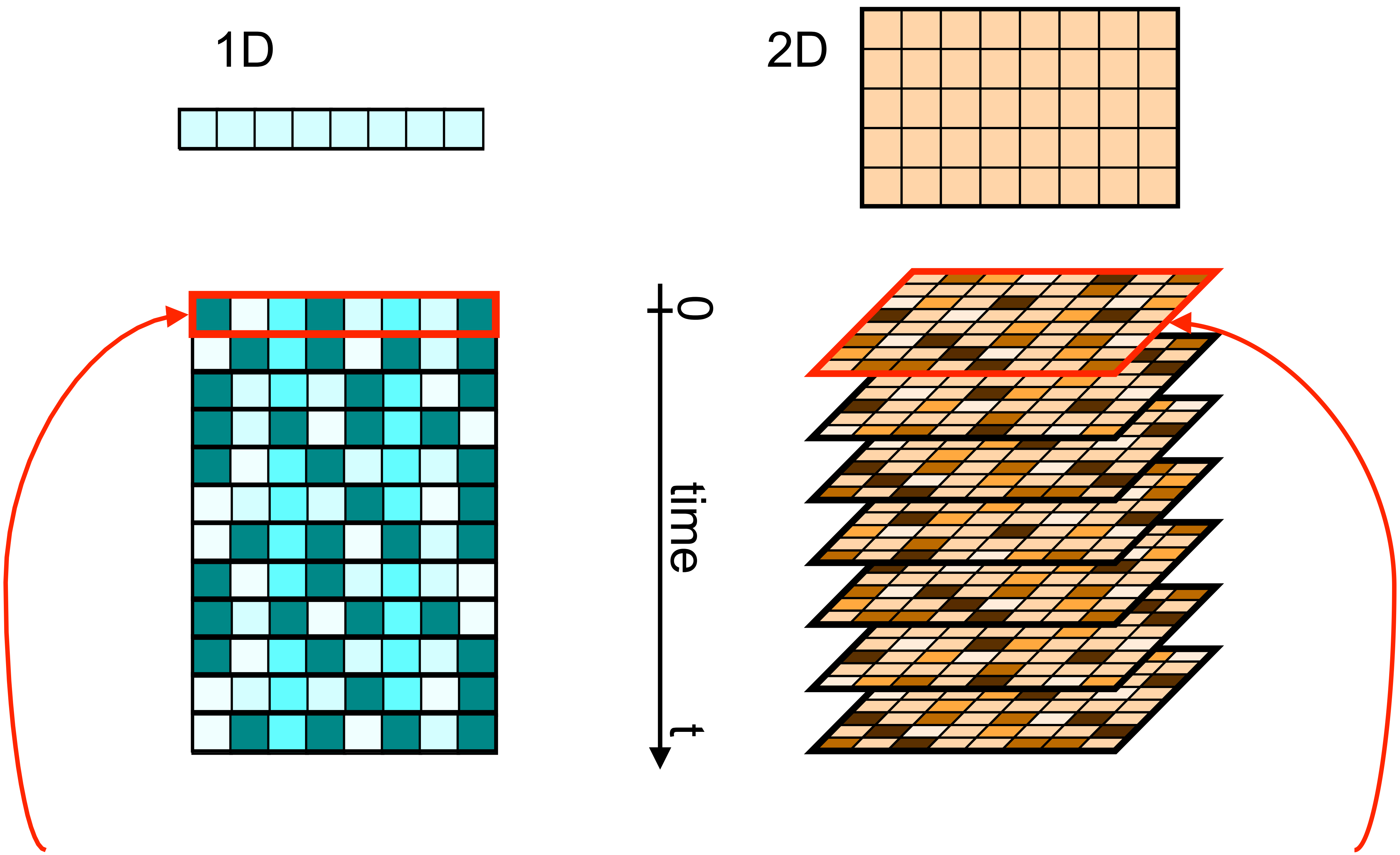
Boundary conditions specify how to build a “virtual” neighbourhood for boundary cells



Some common kinds of boundary conditions



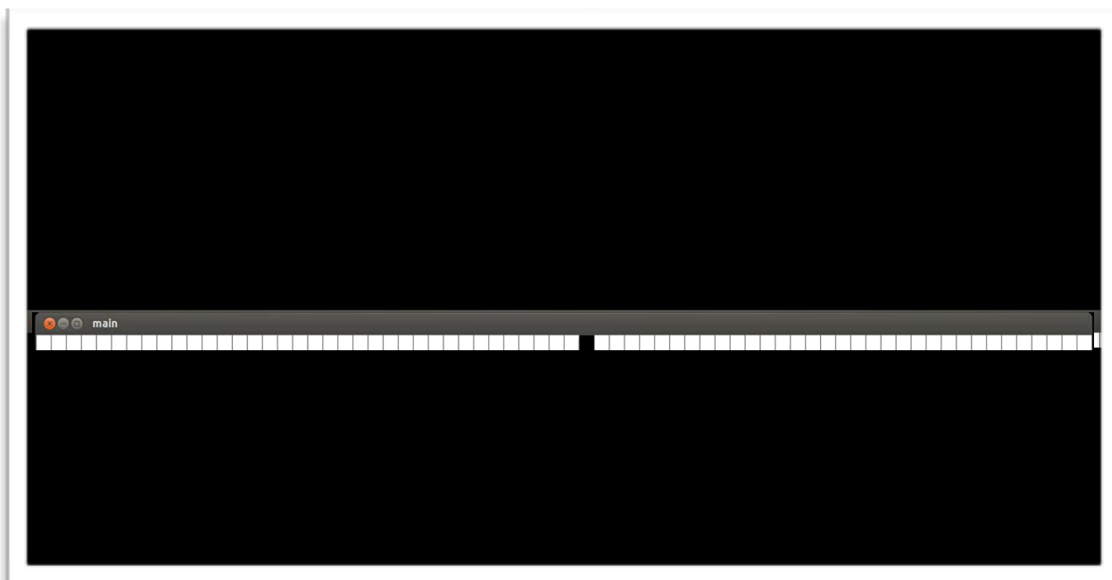
Initial conditions



In order to start with the updating of the cells of the CA we must specify the initial state of the cells (initial conditions or seed)

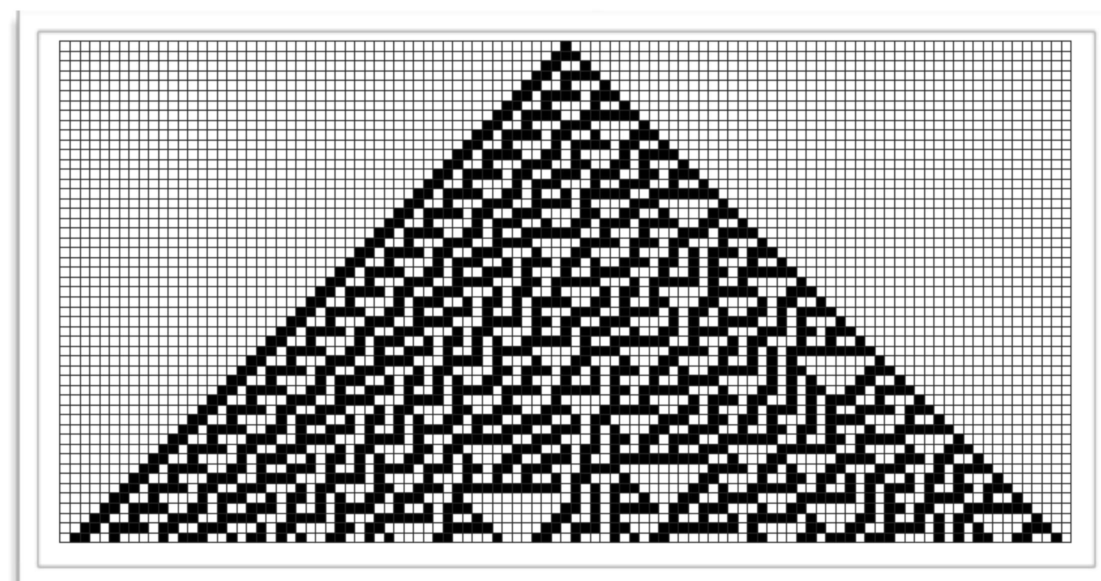
Displaying CA dynamics

1D
Space-time animation

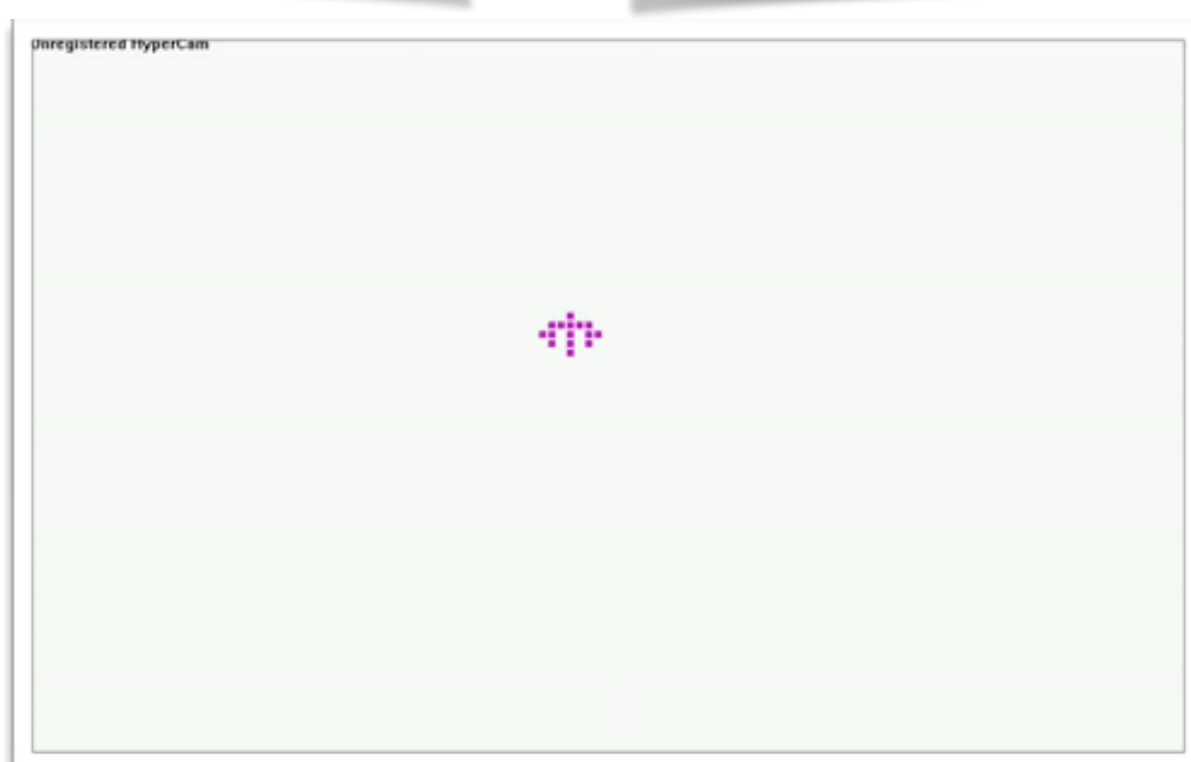


1D
as a static plot

time



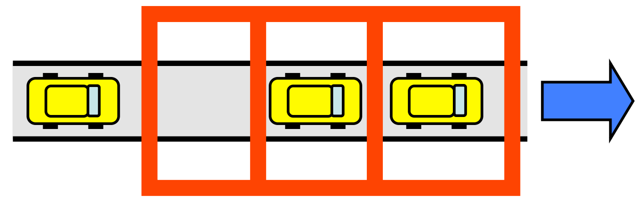
2D
animation of spatial plot



Example: Traffic Modeling

$S = \{s_0, \dots, s_{k-1}\}$
$= \{0, \dots, k-1\}$
$= \{\bullet, \dots, \bullet\}$

8 cases

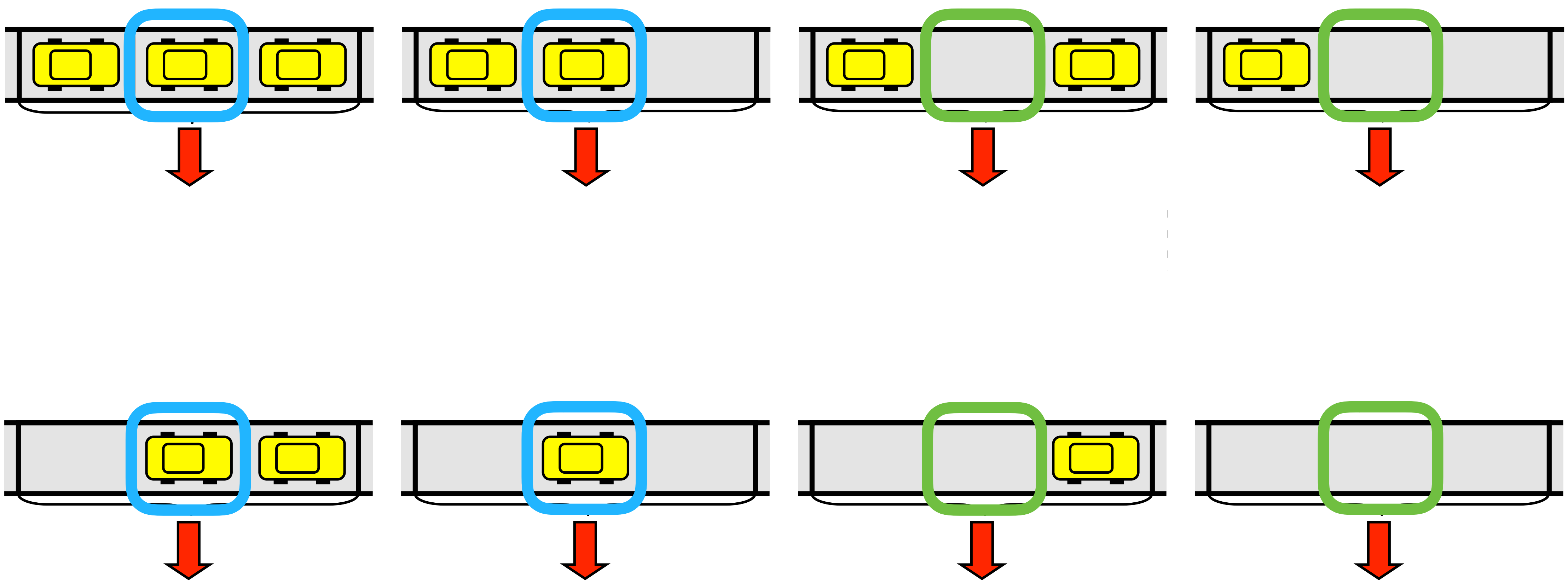


k states

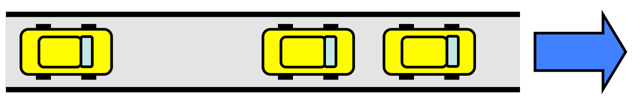
n cells in the neighborhood

k^n

We construct an elementary model of car motion in a single lane, based only on the **local** traffic conditions. The cars advance at **discrete time steps** and at **discrete space intervals**. A car can advance (and must advance) only if the destination interval is free.

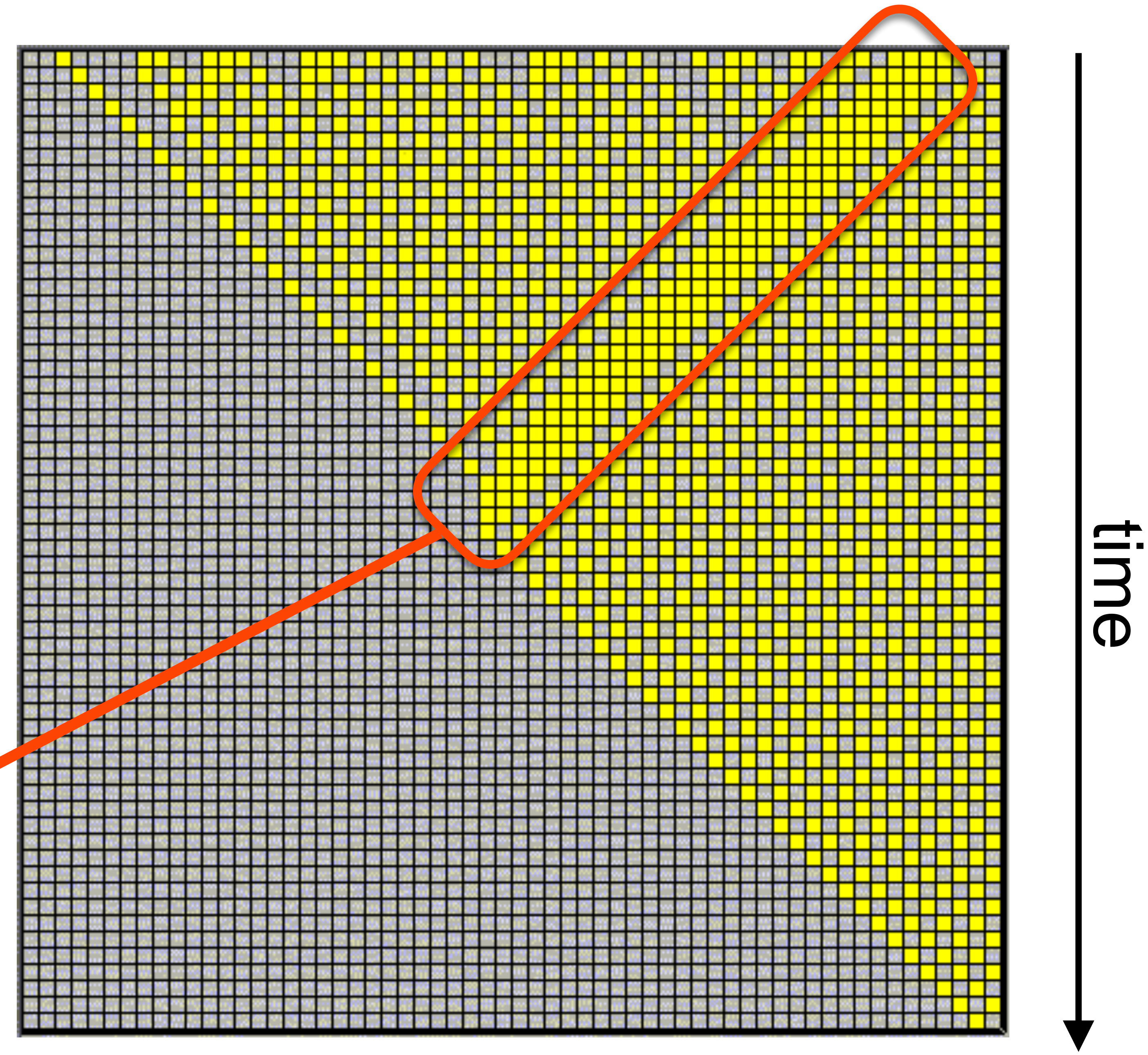


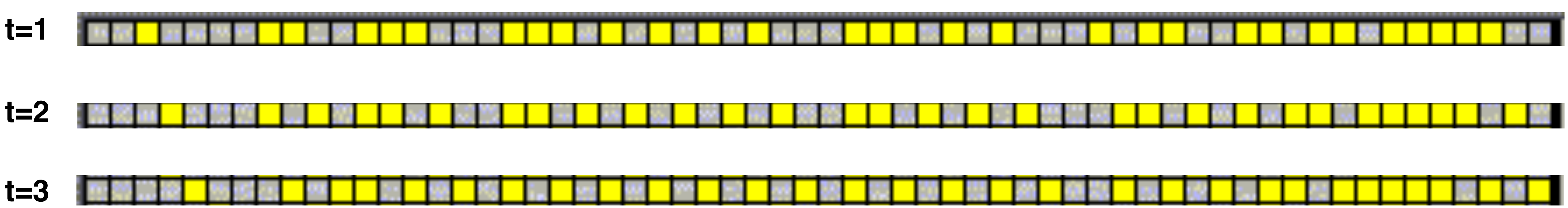
Example: Traffic Jam



Start the *traffic CA* with a **high-density random initial distribution** of cars

we observe a phenomenon of **backward propagation** of a region of extreme traffic congestion (traffic jam).

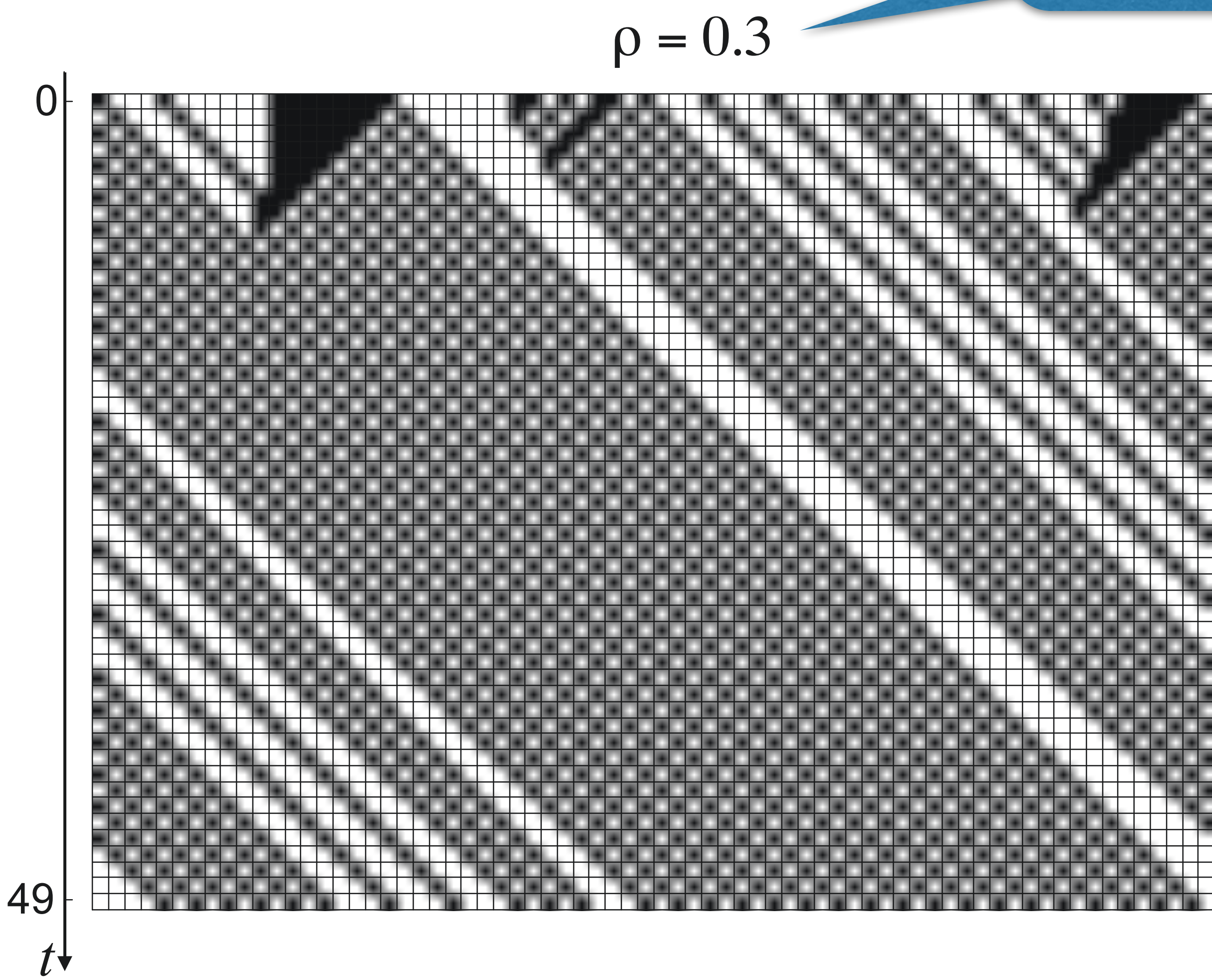


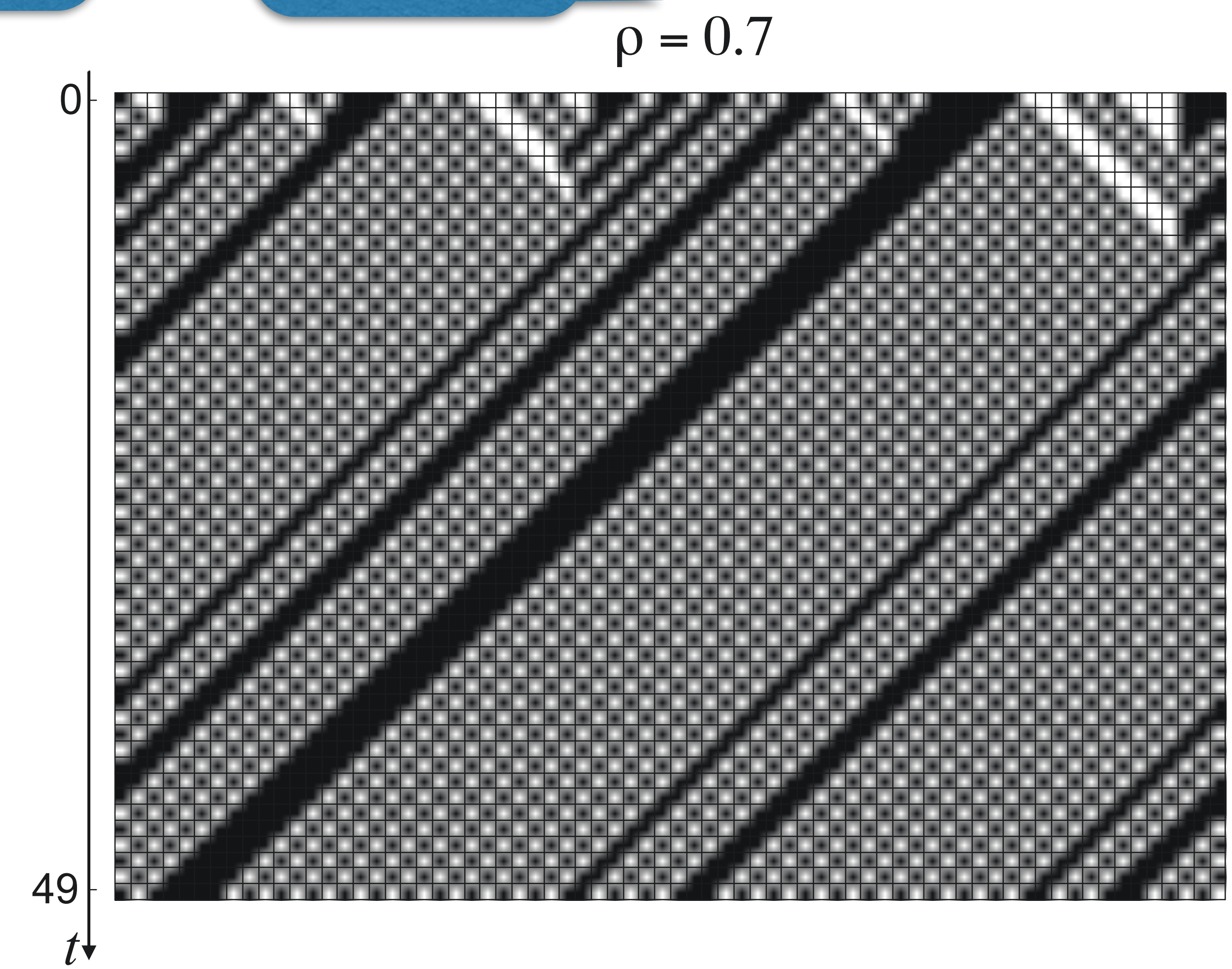


Example: Traffic Jam

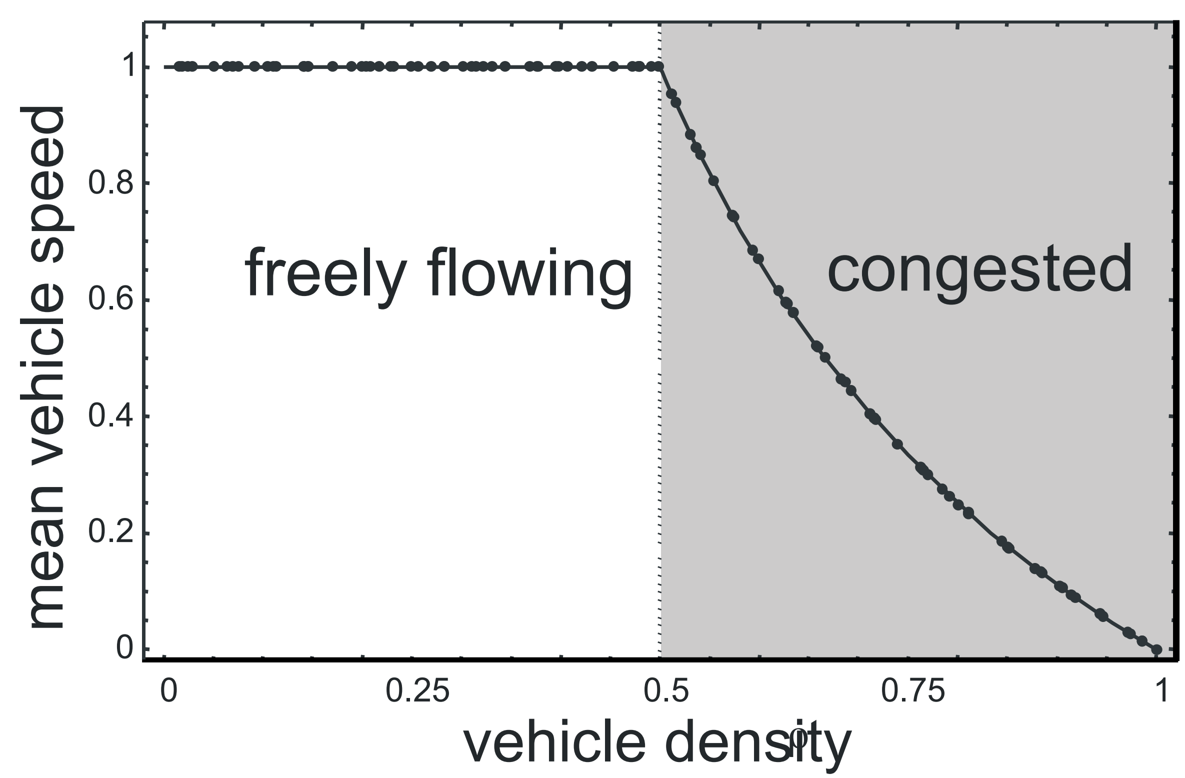
low
density

high
density

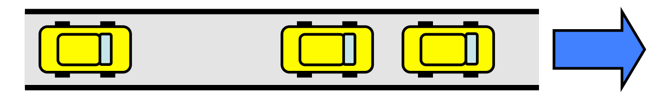


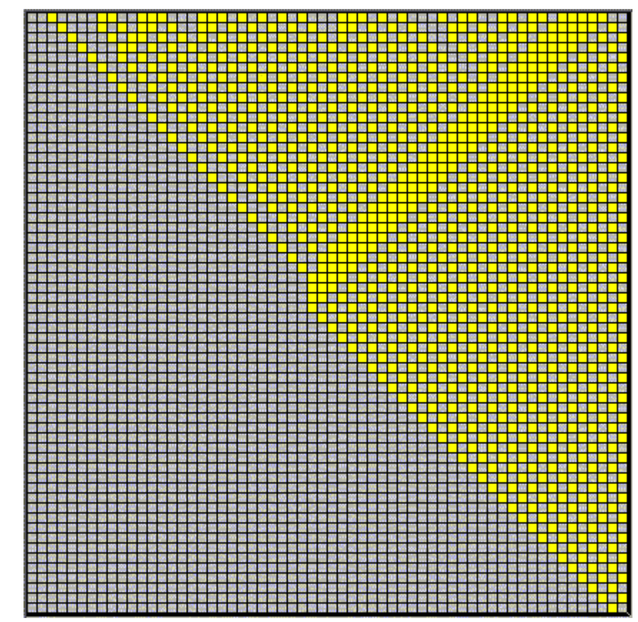


There is a qualitative change of behaviour for $\rho = 0.5$. In the language of physics - there is a *phase transition* between the two regimes at the ***critical*** **density ρ = 0.5**



How to run a CA



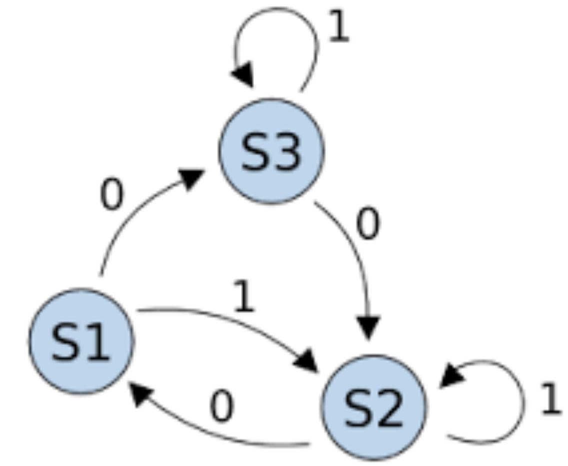


Steps in practice are:

1. Assign the geometry of the CA space
2. Assign the geometry of the neighbourhood
3. Define the set of states of the cells
4. Assign the transition rule
5. Assign the boundary conditions
6. Assign the initial conditions of the CA
7. Repeatedly update all the cells of the CA, until some stopping condition is met (for example, a pre-assigned number of steps is attained, or the CA is in a quiescent state, or cycles in a loop,...).

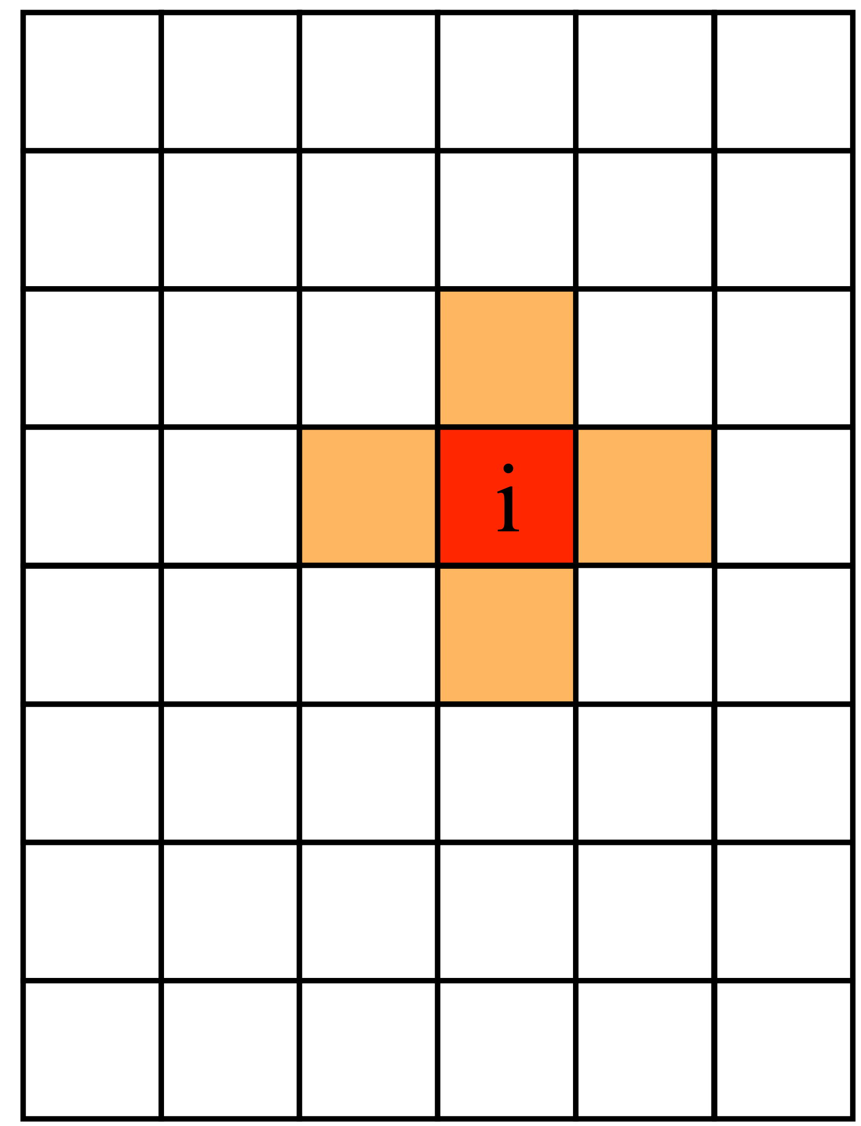

Let’s have a break!

Formal definition of CA



A Cellular Automaton is:

- an **n-dimensional lattice** of
- **identical** and **synchronous** finite state machines
- whose state *s* is updated (synchronously) following a **transition function** (or transition rule) ϕ
- that takes into account the state of the machines belonging to a **neighbourhood** *N* of the machine, and whose geometry is the same for all machines



$$s_i(t+1) = \phi(\, s_j(t) \,;\, j \in N_i \,)$$

Special rules

The transition table of a generic CA can have an enormous number of entries. Special rules can have **more compact definitions**.

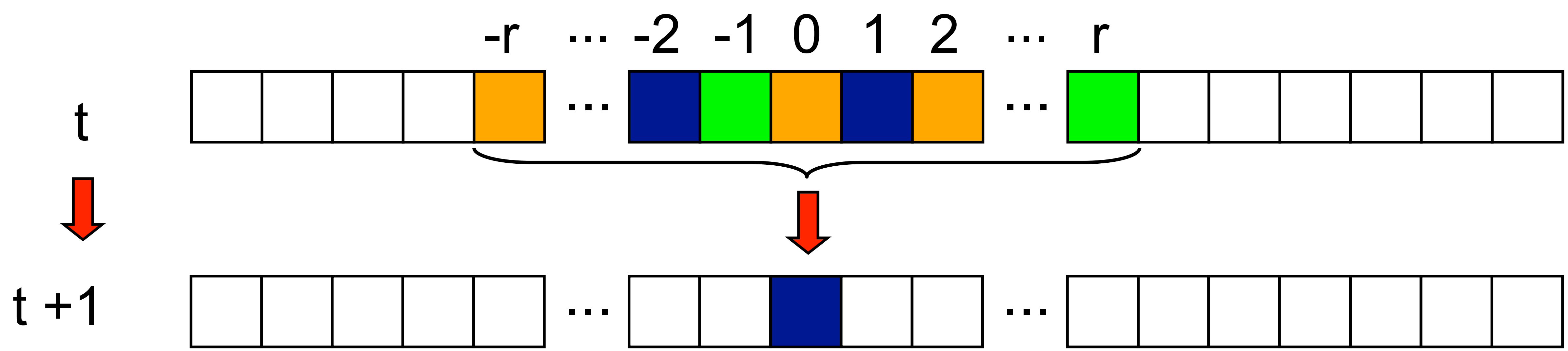
A rule is **totalistic** if the new value of the state depends only on the **sum** of the values of the states of the cells in the neighbourhood.

$$s_i(t+1) = \phi(\Sigma_j s_j(t) ; j \in N_i)$$

A rule is **outer totalistic** if the new value of the state depends on the value of the state of the updated cell **and** on the sum of the values of the states of the other cells in the neighbourhood

$$s_i(t+1) = \phi(s_i(t) , \Sigma_j s_j(t) ; j \in N_i , j \neq i)$$

Rules for 1D CA



k states (colors • , • , • , ...), range (or radius) r

Simplest possible CA?

e.g.: k=2 , r=1→ 256

k=3 , r=1→ ≈ $8 \cdot 10^{12}$

e.g.: k=2 , r=1→ 16 totalistic

k=3 , r=1→ 2187 totalistic

The number of possible rules **grows very rapidly** with k and r.

Rule code for elementary CA

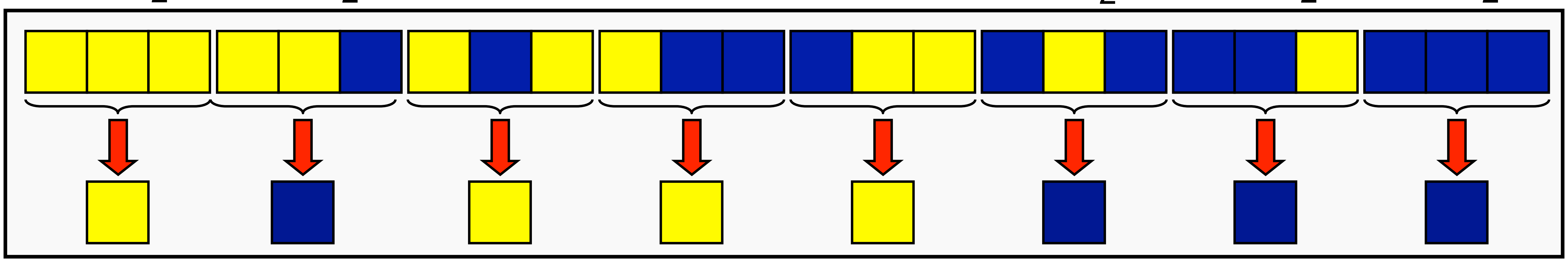
8 transitions

Elementary CA

1D binary CA (k=2) with minimal range (r=1)

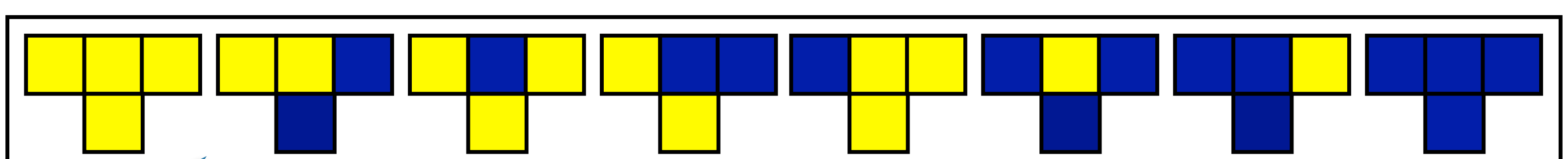
Wolfram's Rule Code (here, [blue square] = 0 , [yellow square] = 1)

111_2 110_2 ... ... 010_2 001_2 000_2



1 0 1 1 1 0 0 0

$$10111000_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + \ldots + 0 \cdot 2^0 = 184_{10} \Rightarrow \text{Rule 184}$$



R184

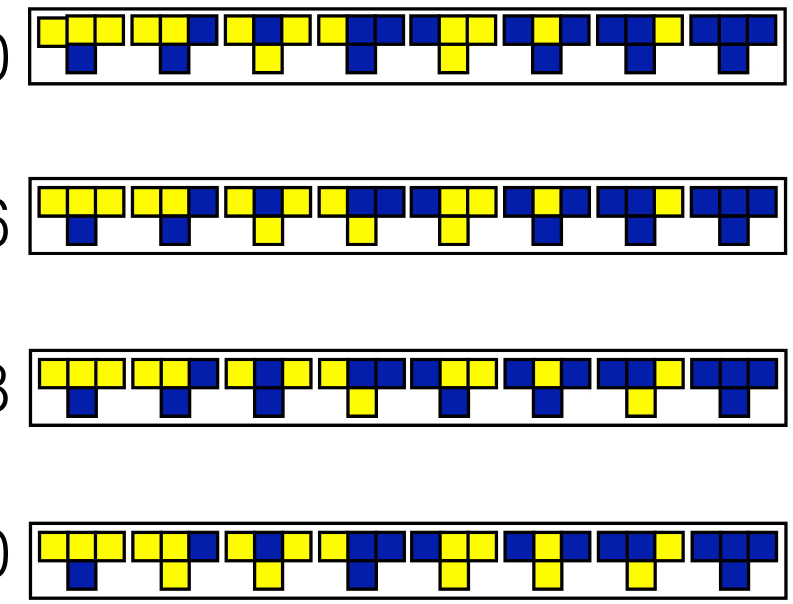
traffic rules

256 possible rules

Example of elementary CA

Download these Matlab Files and try rules 40, 56, and 18, and 110
http://www.mathworks.com/matlabcentral/fileexchange/26929-elementary-cellular-automata

40
56
18
110



The rules are described here:
http://mathworld.wolfram.com/ElementaryCellularAutomaton.html

```
>pattern=elementaryCellularAutomata(RULENUMBER,200,round(rand(1,200)));
>pattern( pattern==1 )=255;
>image(pattern)
```

There are four *qualitative* behavioural classes:

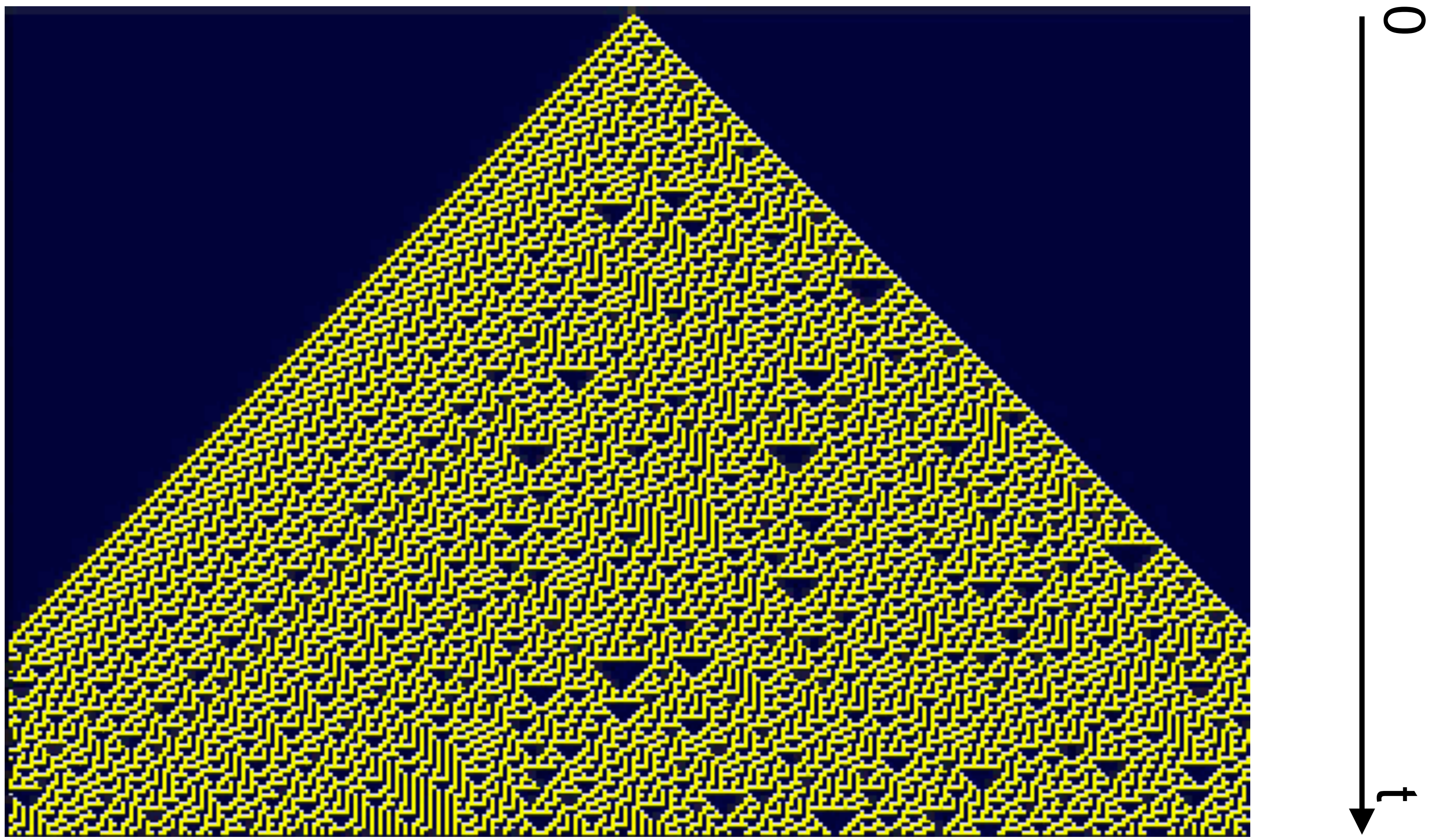
Uniform final state
Simple stable or periodic final state
Chaotic, random, nonperiodic patterns
Complex, localized, propagating structures

Which one of these rules has which behaviour?

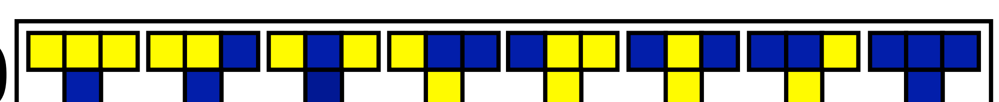
Example of elementary CA

Rule 30 is used by Mathematica as its Random Number Generator (RNG are ubiquitous in bio-inspired experiments).

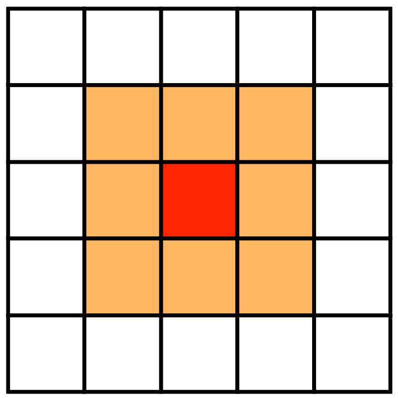
1, 3, 3, 6, 4, 9, 5, 12, 7, 12, 11, 14, 12, 19, 13, 22, 15, 19, ...



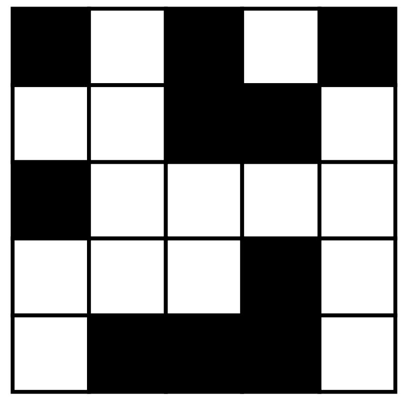
R30



Classical 2D CA: Life

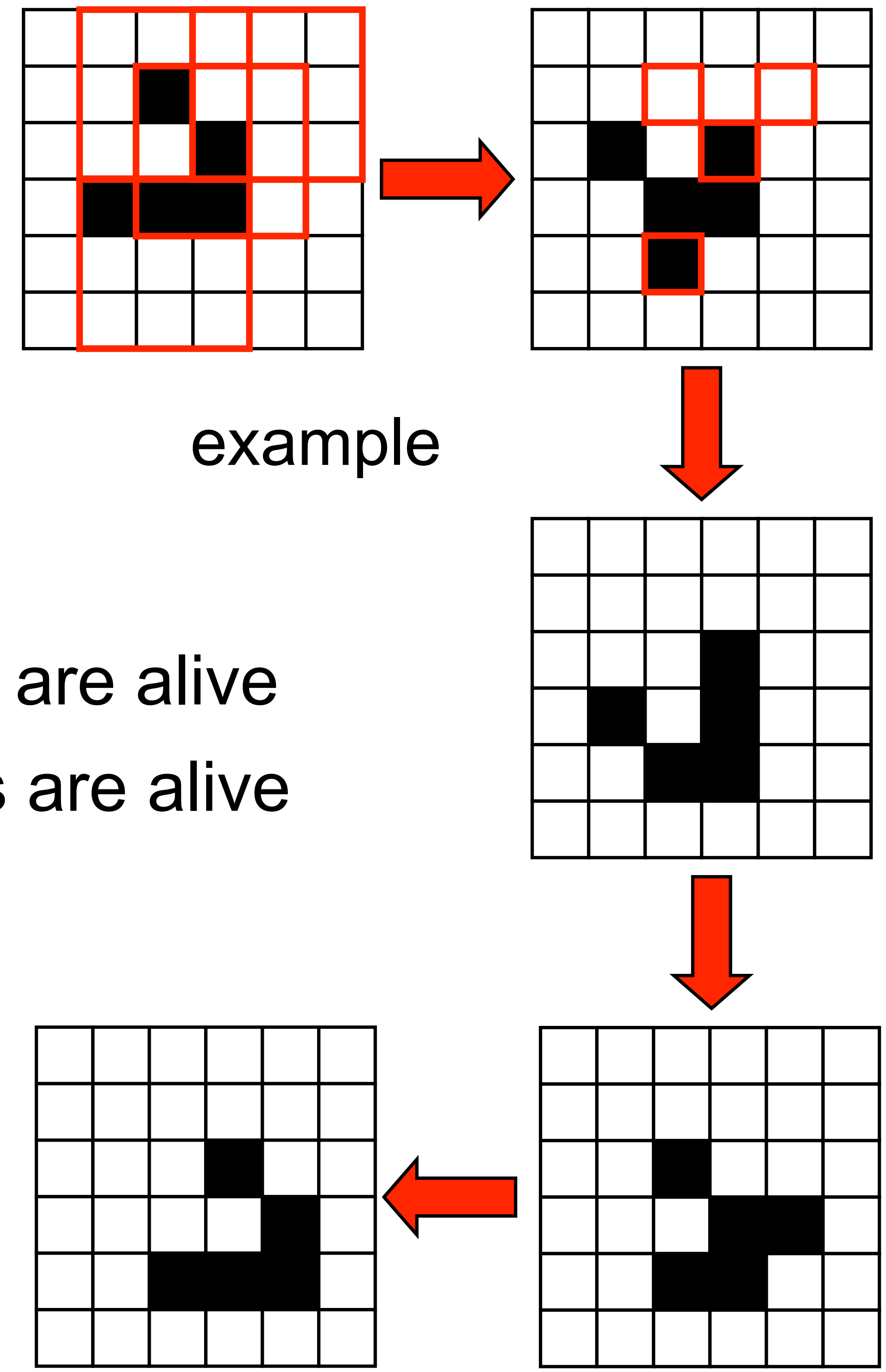


Moore neighborhood



two states

dead ☐ alive ■



example

Outer totalistic rule (John Conway)

- **Birth** ☐ → ■ if exactly 3 neighbors are alive
- **Survival** ■ → ■ if 2 or 3 neighbors are alive
- **Death** ■ → ☐

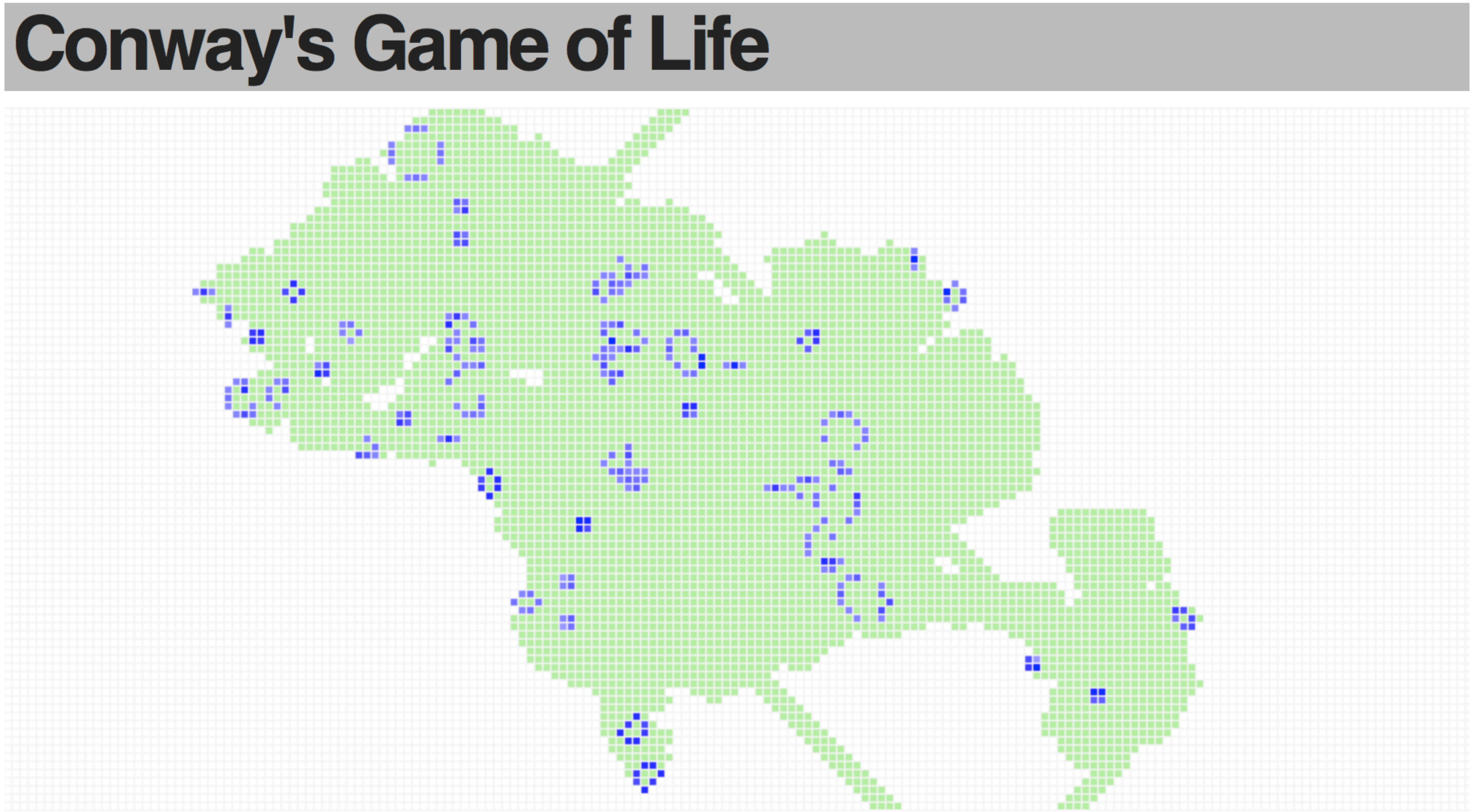
  from "isolation" if 0 or 1 n. a. a

  from "overcrowding" if more than 3 neighbors are alive

(often coded as "rule 23/3")

Life continued

pmav.eu/stuff/javascript-game-of-life-v3.1.1/



Epic video:

https://www.youtube.com/watch?v=C2vgICfQawE

Numberphile: Interview with John H. Conway

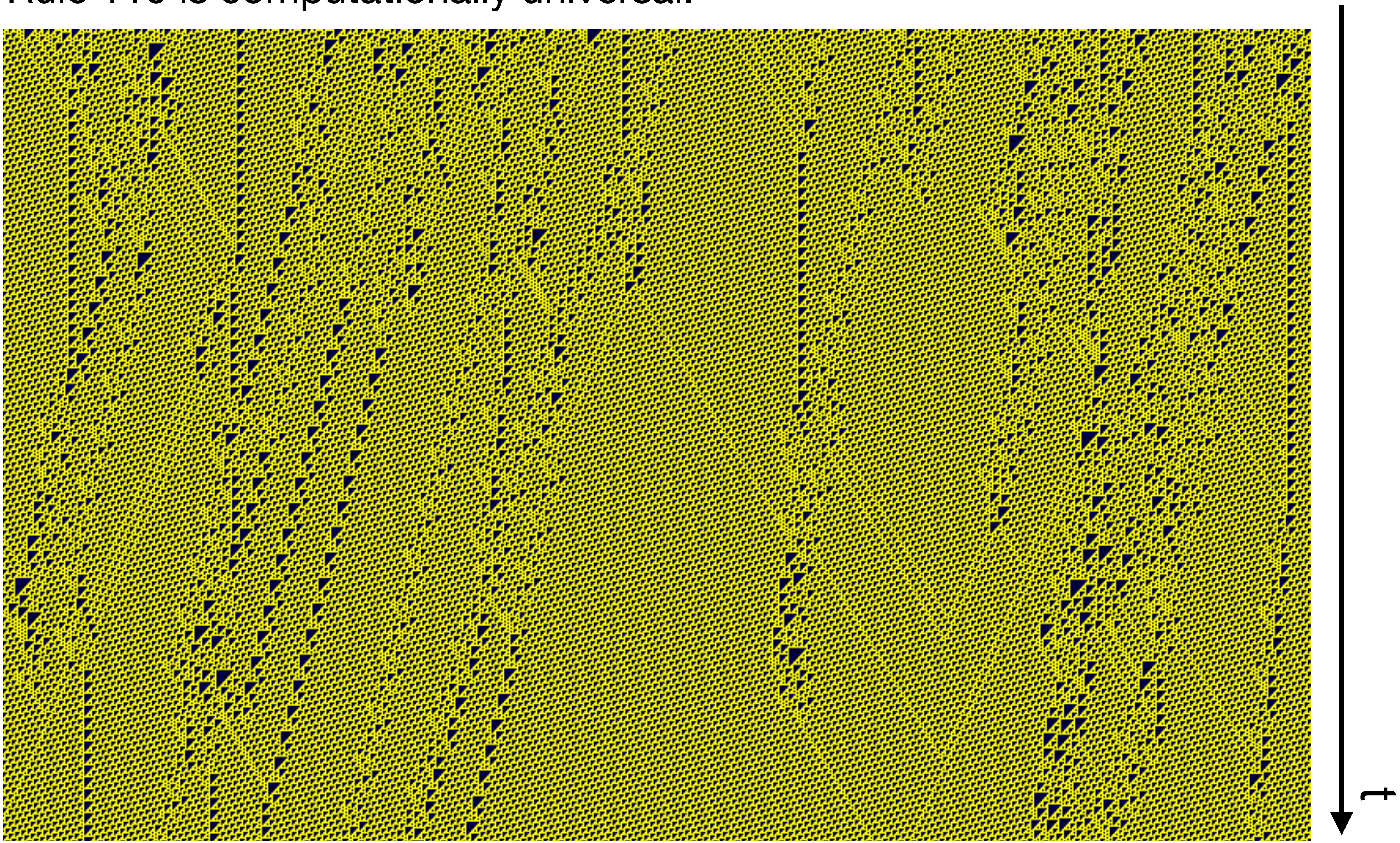
https://www.youtube.com/watch?v=R9Plq-D1gEk

Interesting facts

- In **Game of Life** we can define signals (as streams of gliders interpreted as bits), implement all logic gates (AND, NOT,...), implement delays, memory banks, signal duplicators, and so on.
- Hence, **Game of Life can emulate any computing machine**; we say that it is capable of **universal computation**.
- The theory of computation says that, in general, given an initial state for the automaton, there is no short-cut way to predict the result of Life's evolution. We must run it.
- We say that Game of Life is **computationally irreducible**.
- In simple words, this means that a very **simple CA** such as Game of Life can **produce highly nontrivial behaviours**, that cannot be predicted simply by observing the transition rule.
- The "universe" constituted by a CA can be an interesting backcloth for the emergence of complex phenomena.

Universality in 1D

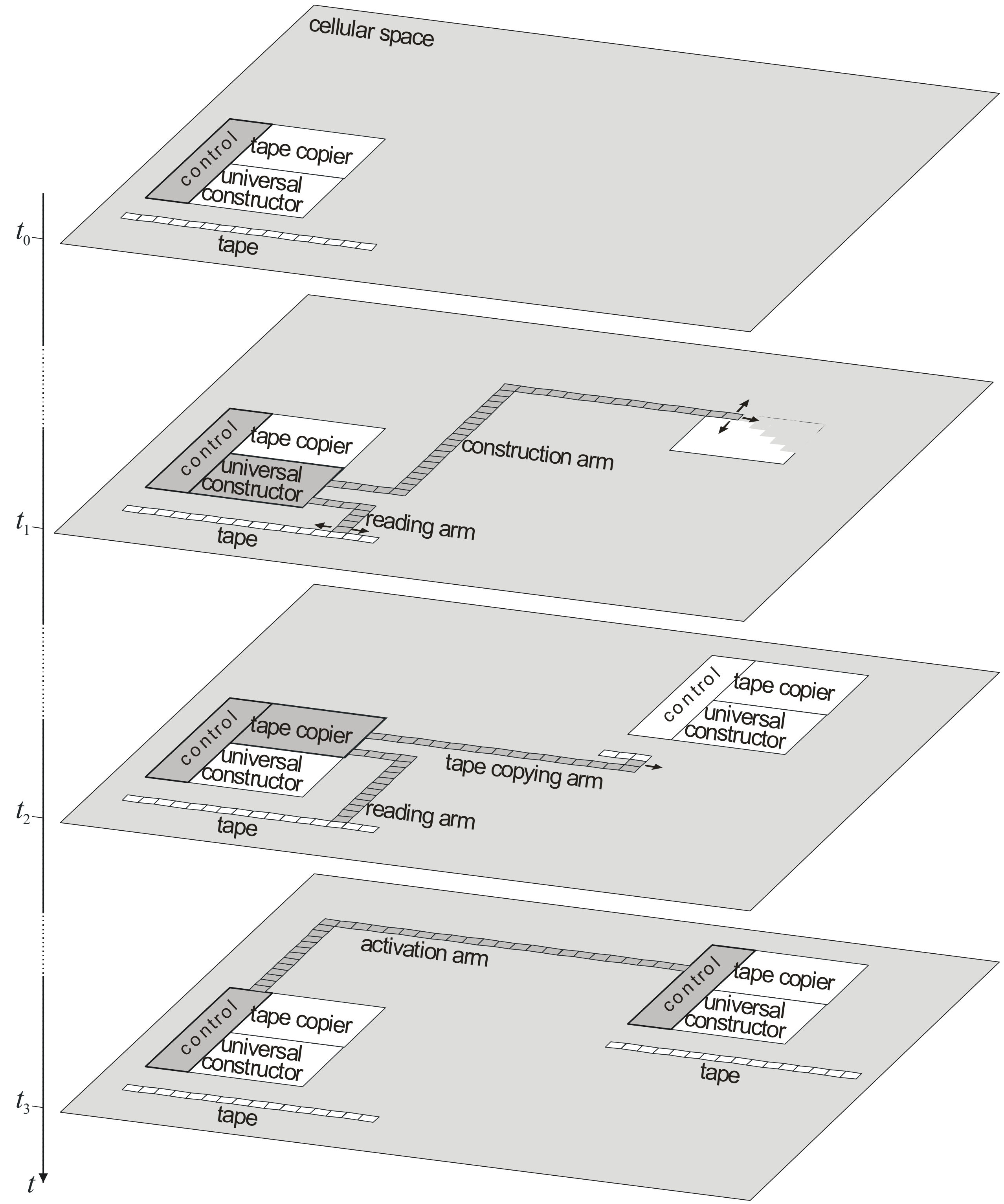
CA even simpler than *Life* display the same properties.
Rule 110 is computationally universal.



Self replication machine

von Neumann solved his problem by defining an automaton composed by a **universal constructor UC** and a description D(M) of the machine to be generated.

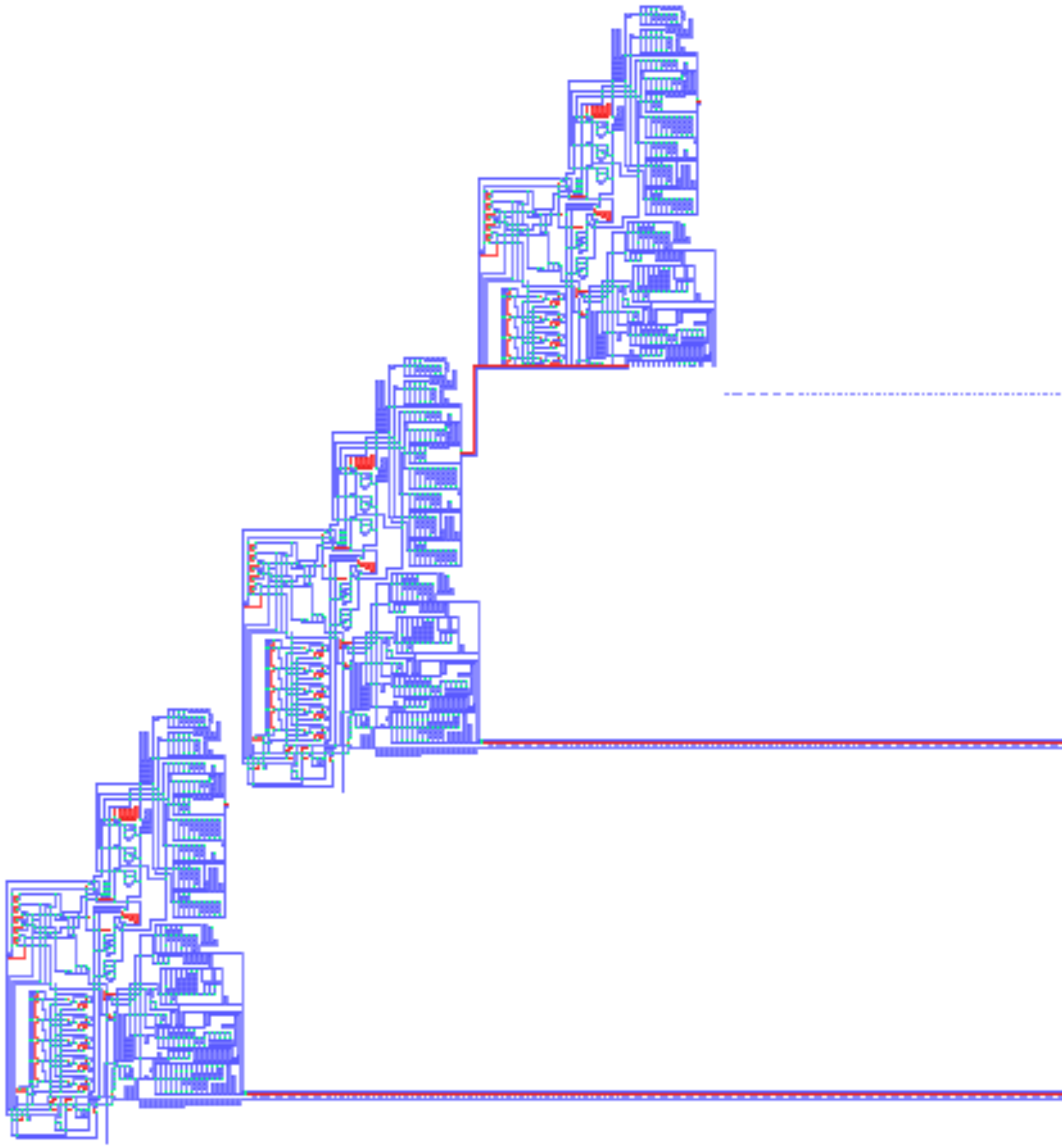
von Neumann automata is quite complex (**29 states** per cell, and about **200.000** active cells)



Self replication machine

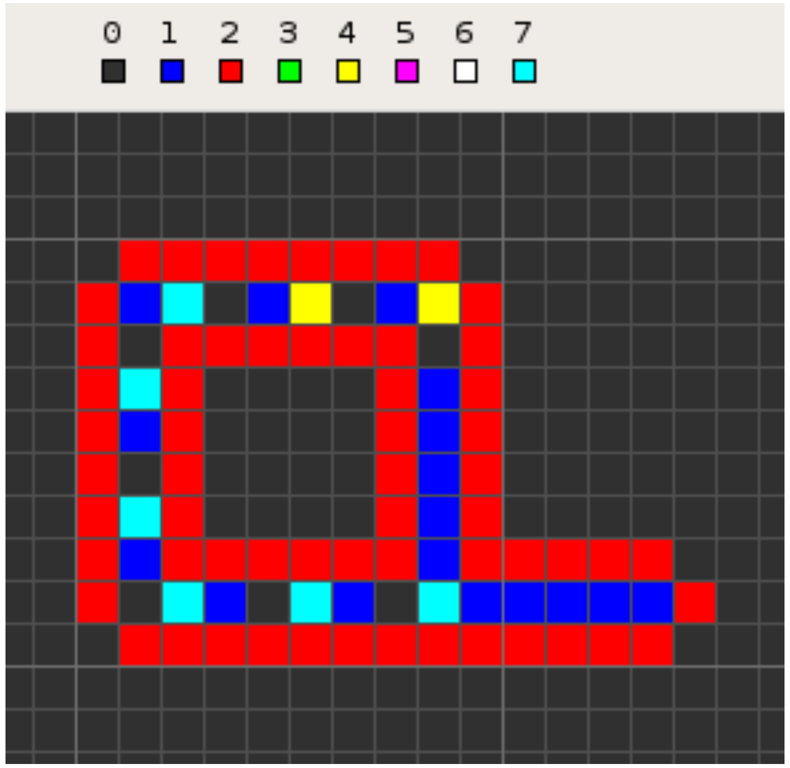
von Neumann solved his problem by defining an automaton composed by a **universal constructor UC** and a description D(M) of the machine to be generated.

von Neumann automata is quite complex (**29 states** per cell, and about **200.000** active cells)

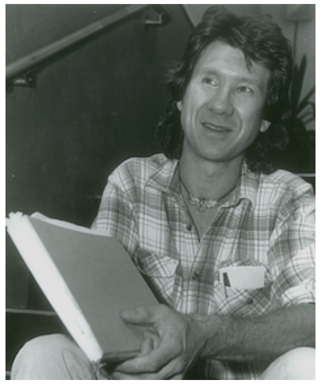


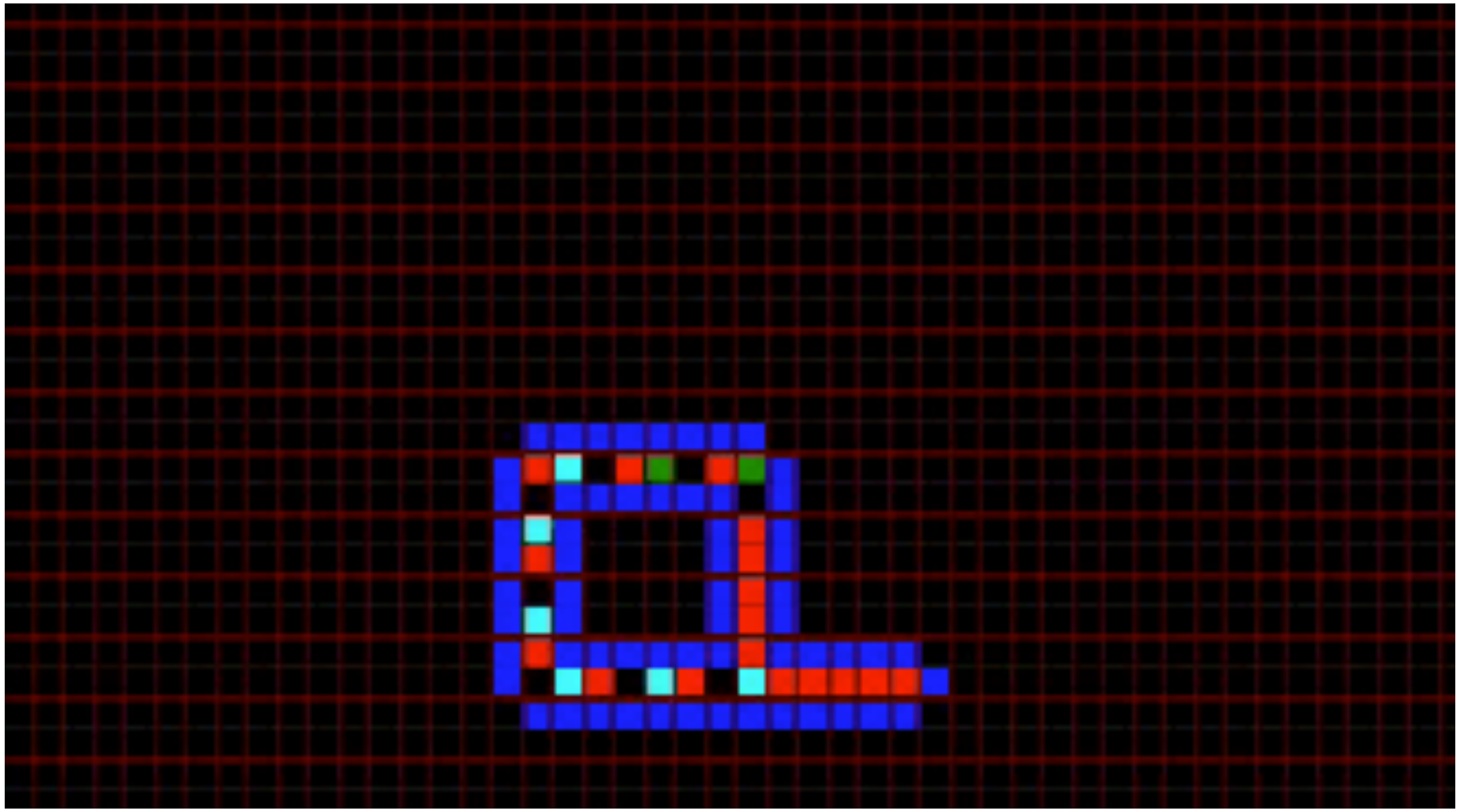
Self replication machine

Other scientists focused on the issue of self-reproduction and offered simpler solutions to this sub-problem (trivial self-reproduction)



Example: 1984 Langton's Loop (Artificial Life)

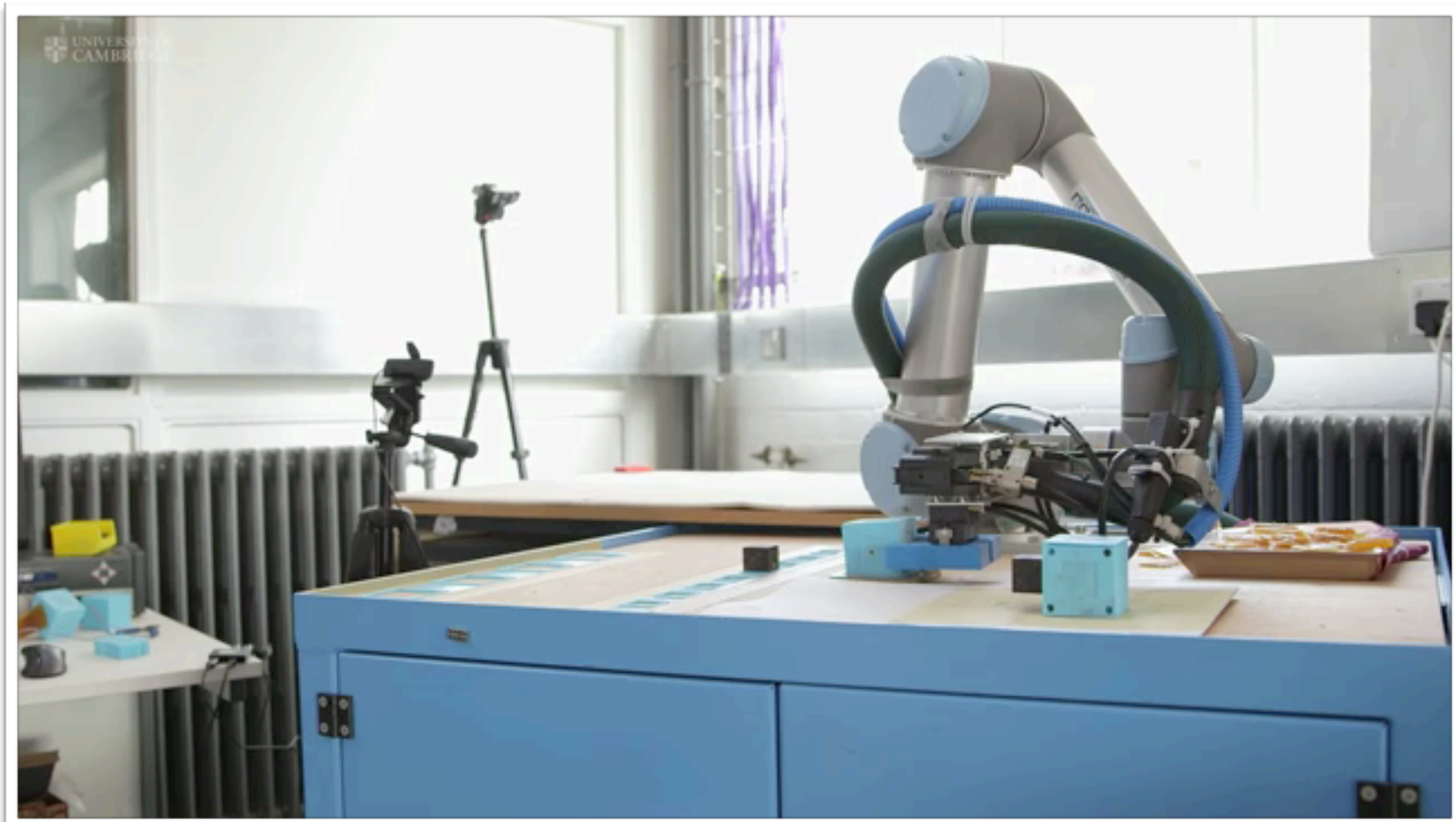




Cool self replicating robots (Cornell)

https://www.youtube.com/watch?v=gZwTcLeelAY

Cool evolutionary robot (Cambridge)



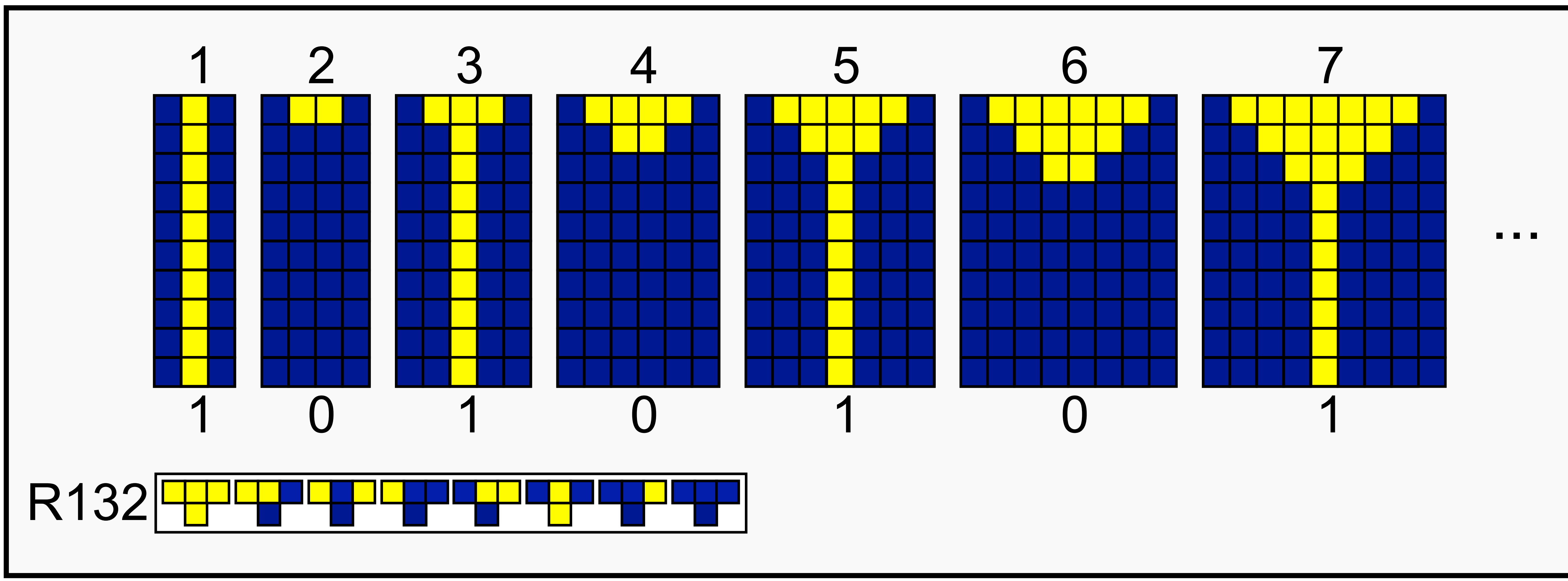
https://www.youtube.com/watch?v=PMHx5ePHjbs

https://www.youtube.com/watch?v=LJpe7myPb7U

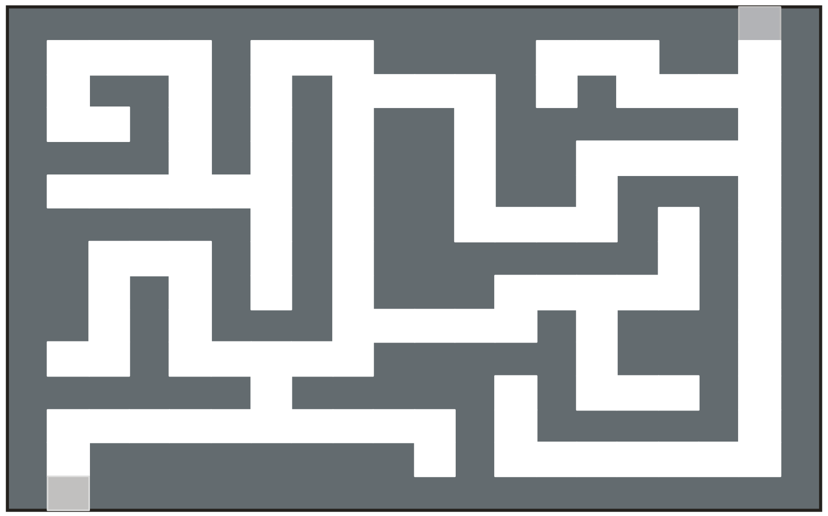
Computation with CA

CA used as input-output devices. The initial state is the input. The CA should go to a quiescent state (fixed point), which is the output.

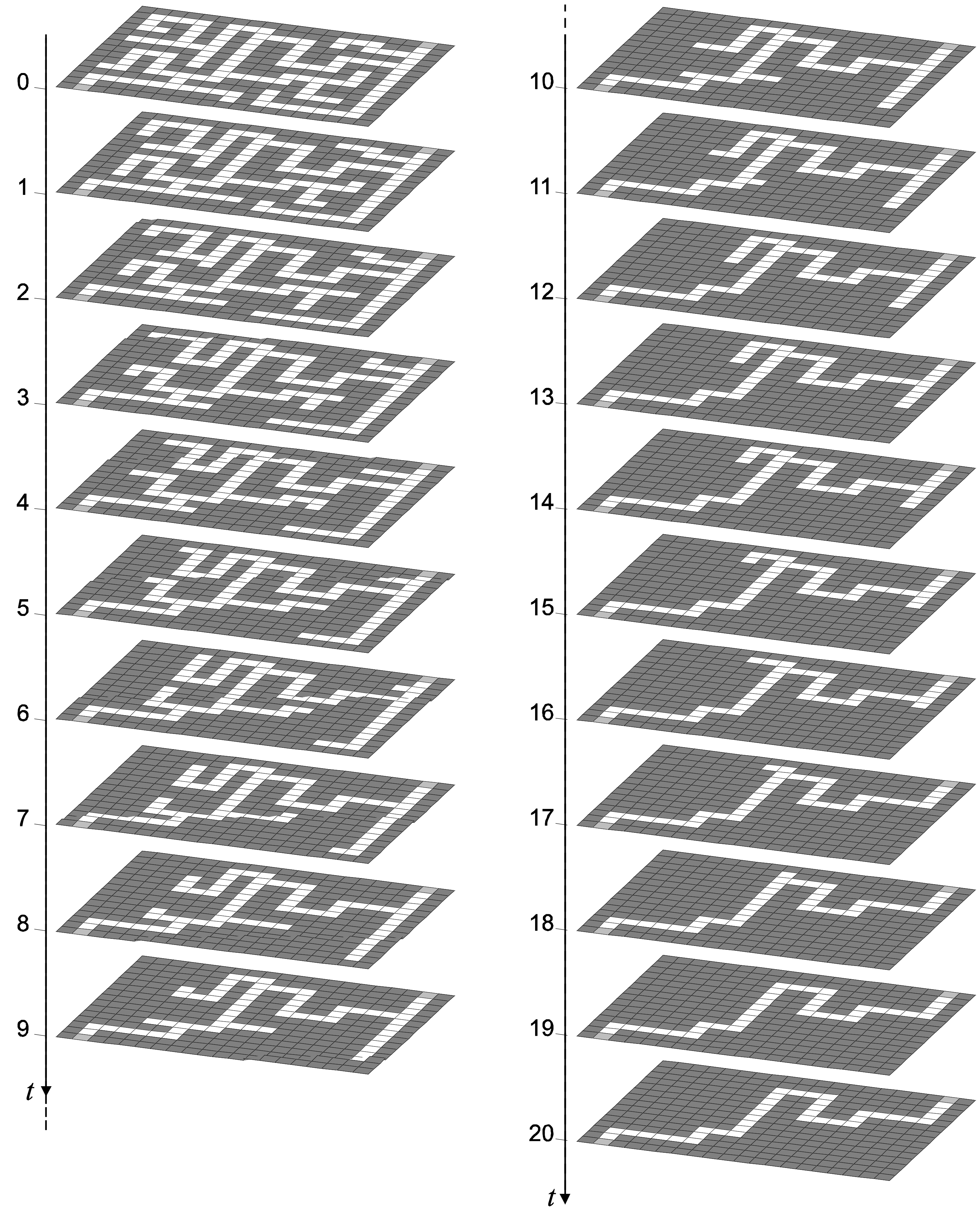
Example: Remainder after division by 2



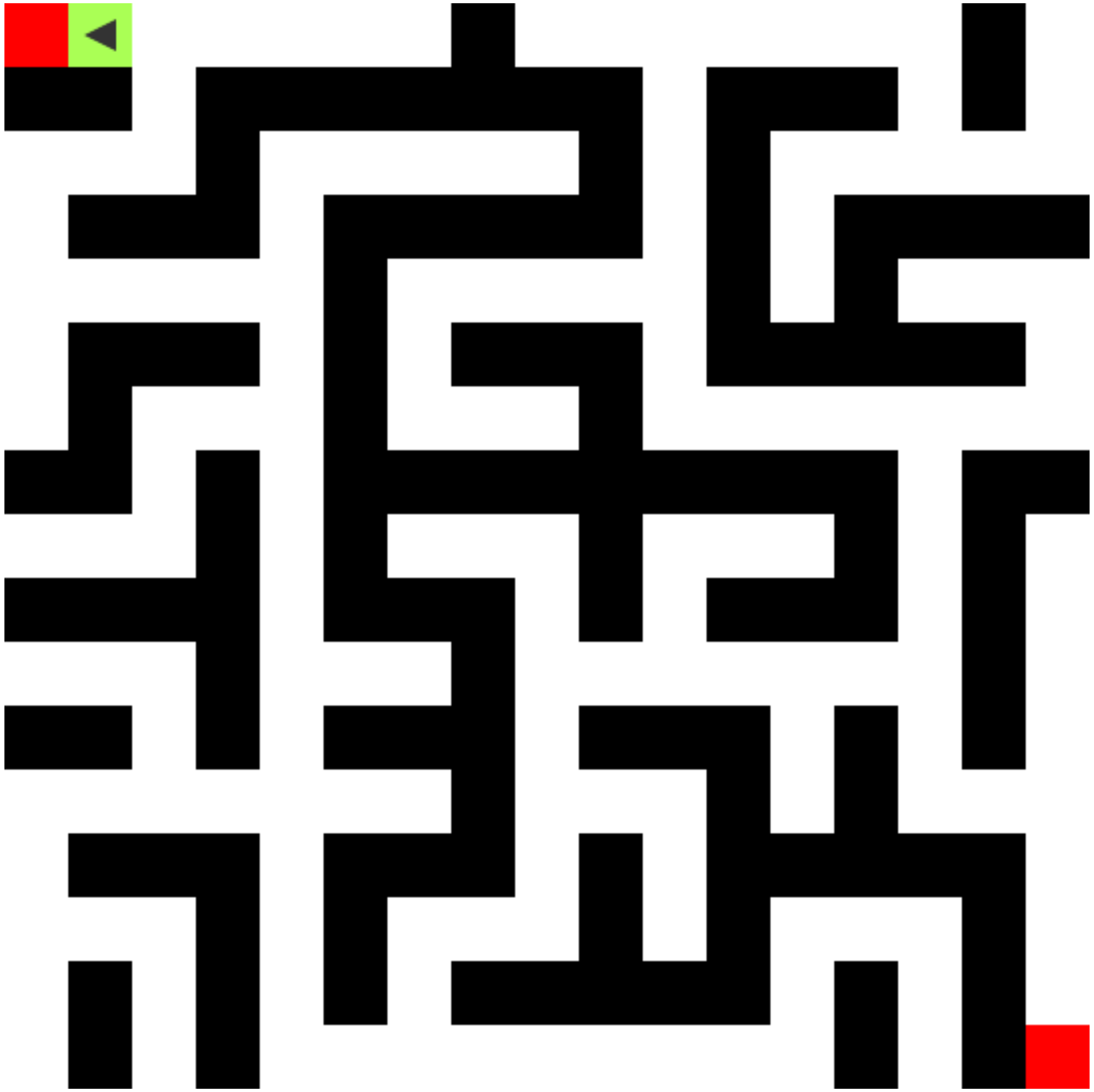
Example: CA maze solver



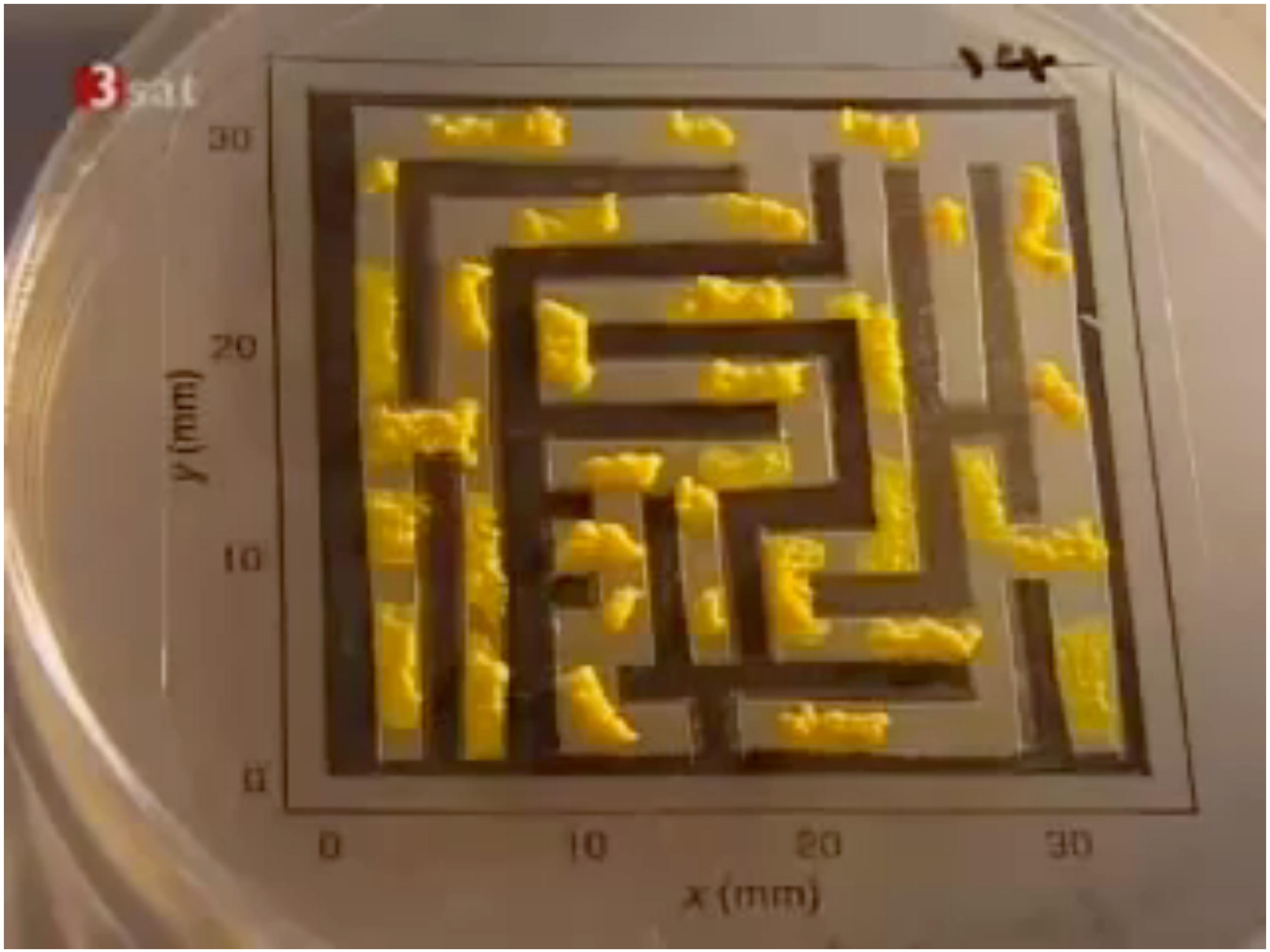
- Given a maze the problem consists in finding a path from the entrance to the exit.
- The conventional approach marks blind alleys sequentially
- The **CA solver** removes blind alleys **in parallel**



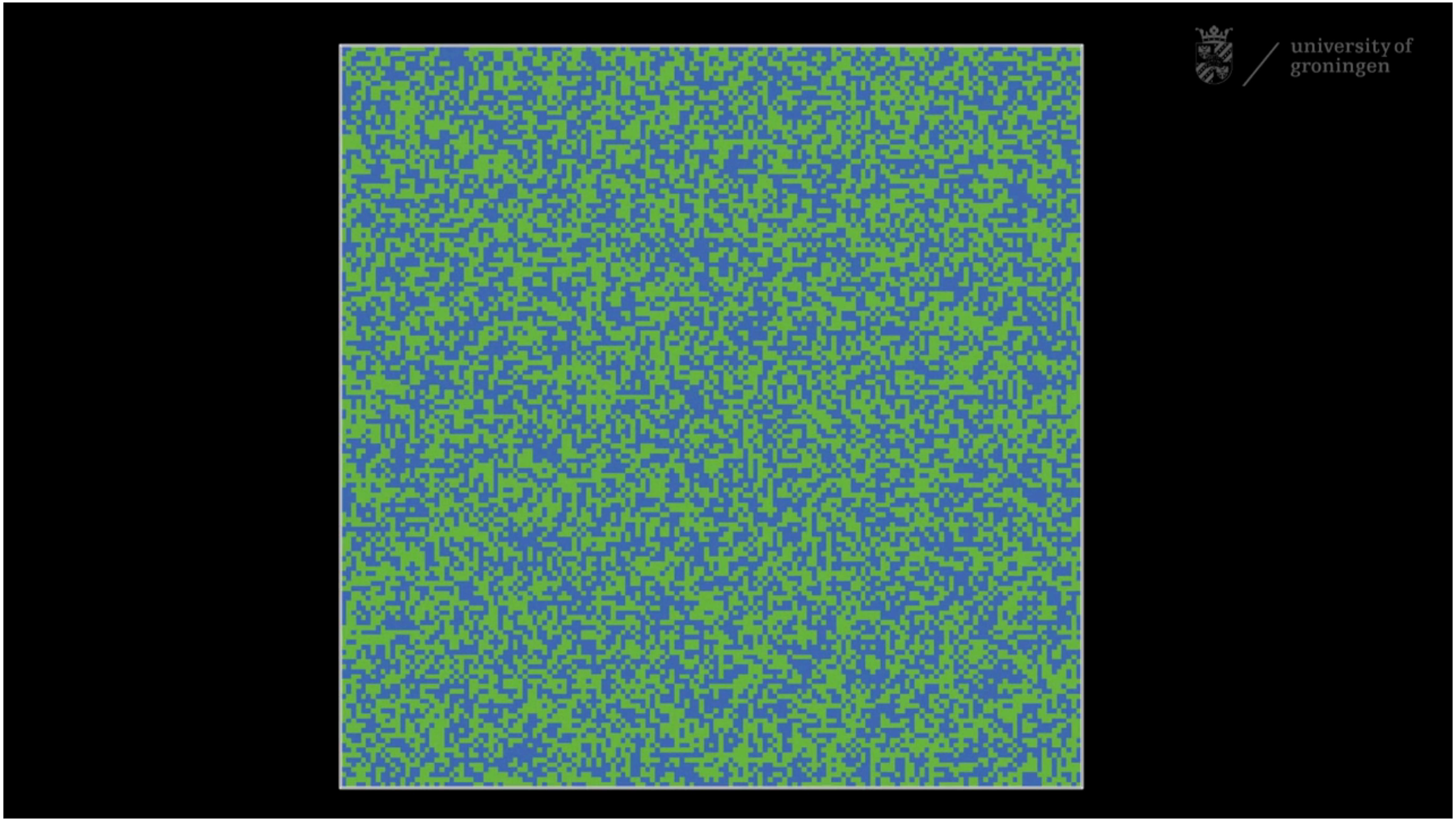
Example: CA maze solver



Maze Solve in Biology: Slime Mold



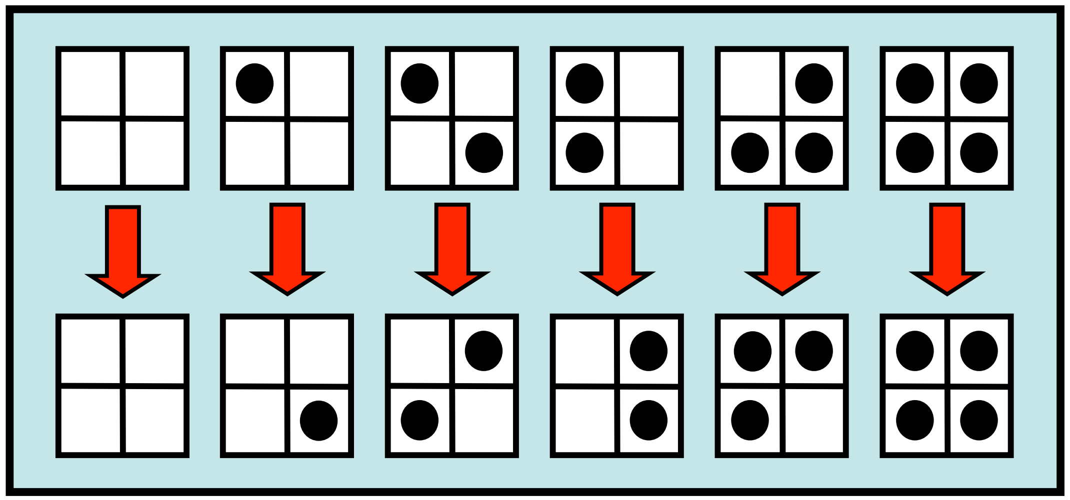
Example: Opinion clustering



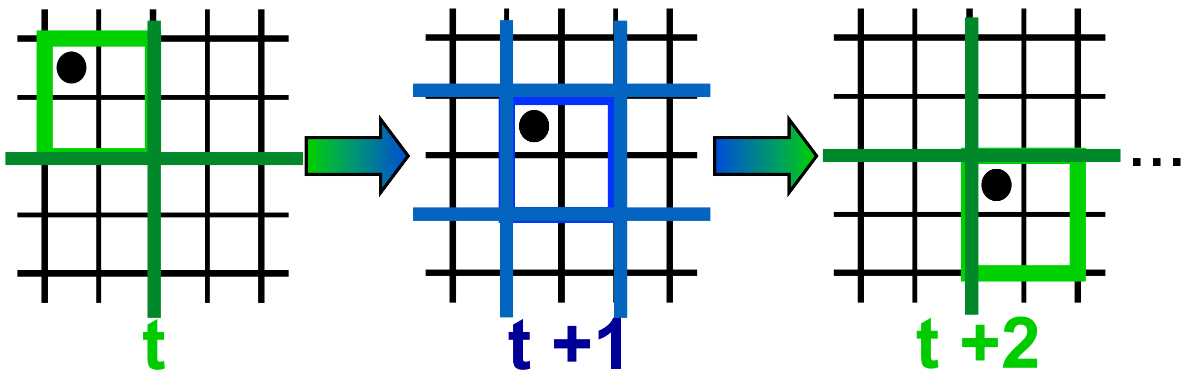
- 2 states: pro and con
- Transition: going with the majority
- 50/50 - then keep the opinion

Particle CA

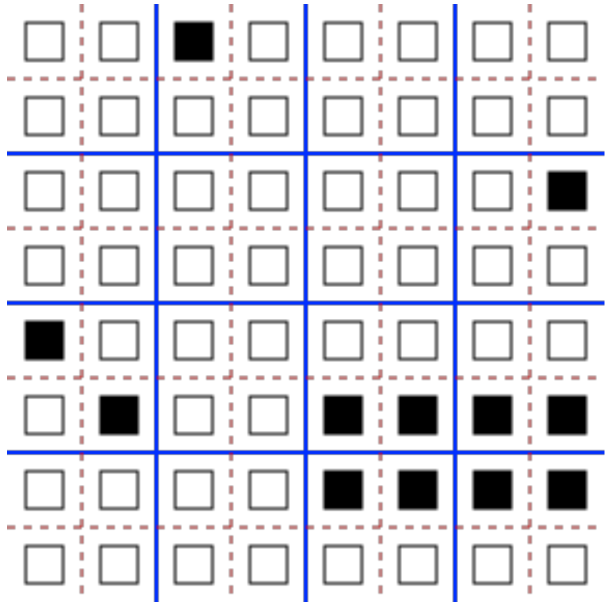
CA can be used to model phenomena that involve **particles.** The **transition rule** can be specified in terms of the **motion of particles** within blocks of two by two cells (**block rules**).



The automaton space is partitioned in **non-overlapping blocks** – (with **different partitions at different time steps**)



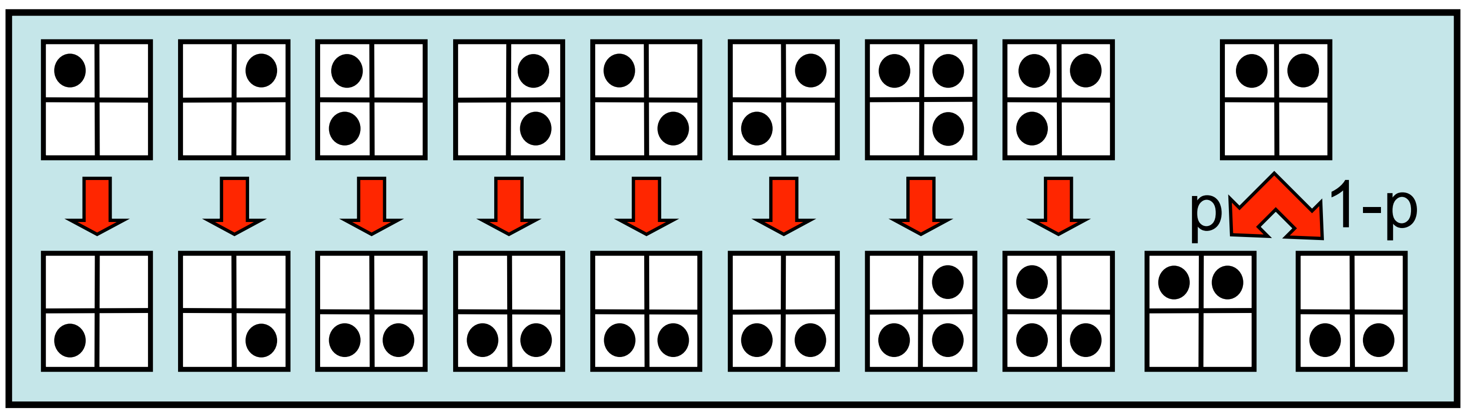
To allow the propagation of information the position of the blocks **alternates** between an odd and an even partition of the space (*Margolus* neighbourhood).



Complex Systems

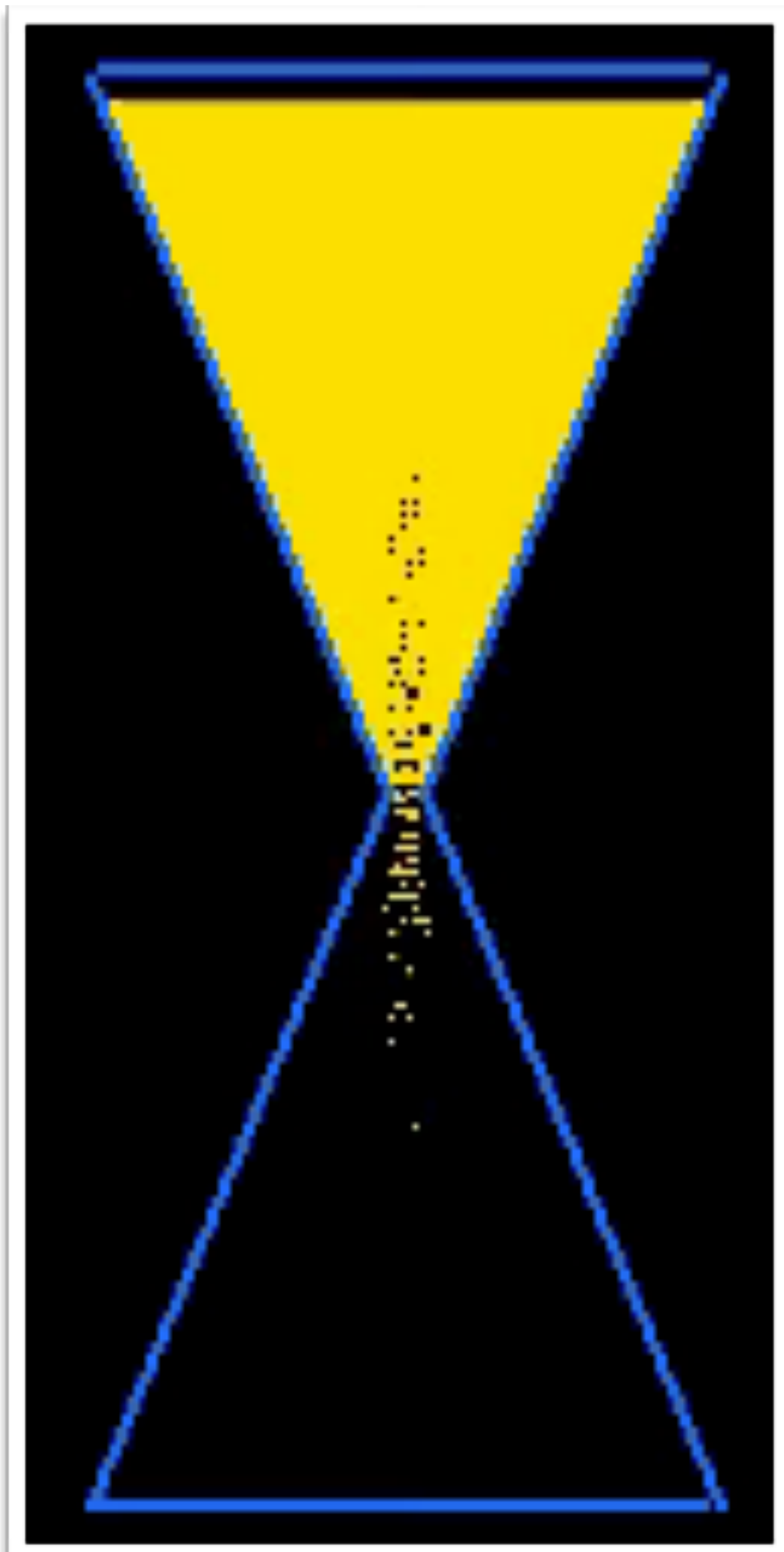
Cellular systems allow the modelling and simulation of phenomena that are difficult to describe with conventional mathematical techniques

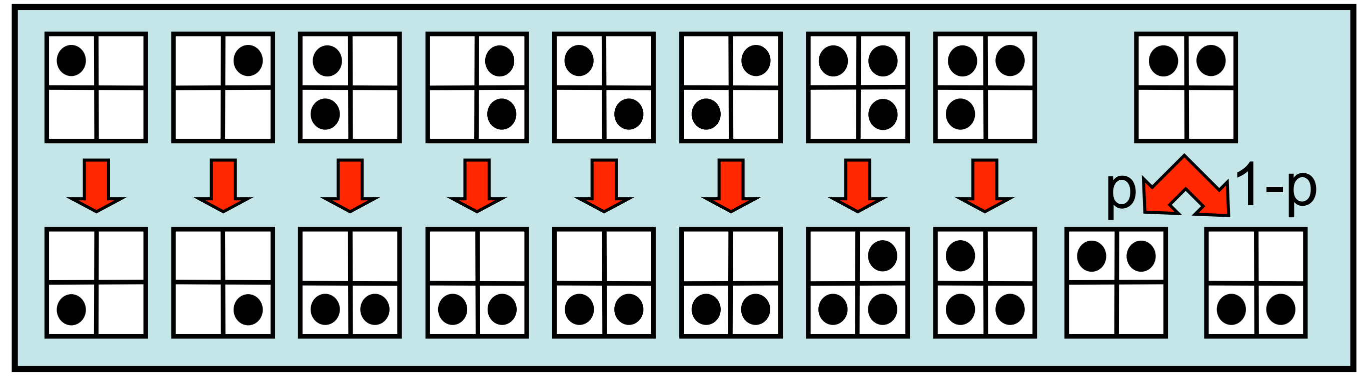
Example: The sand rule with friction

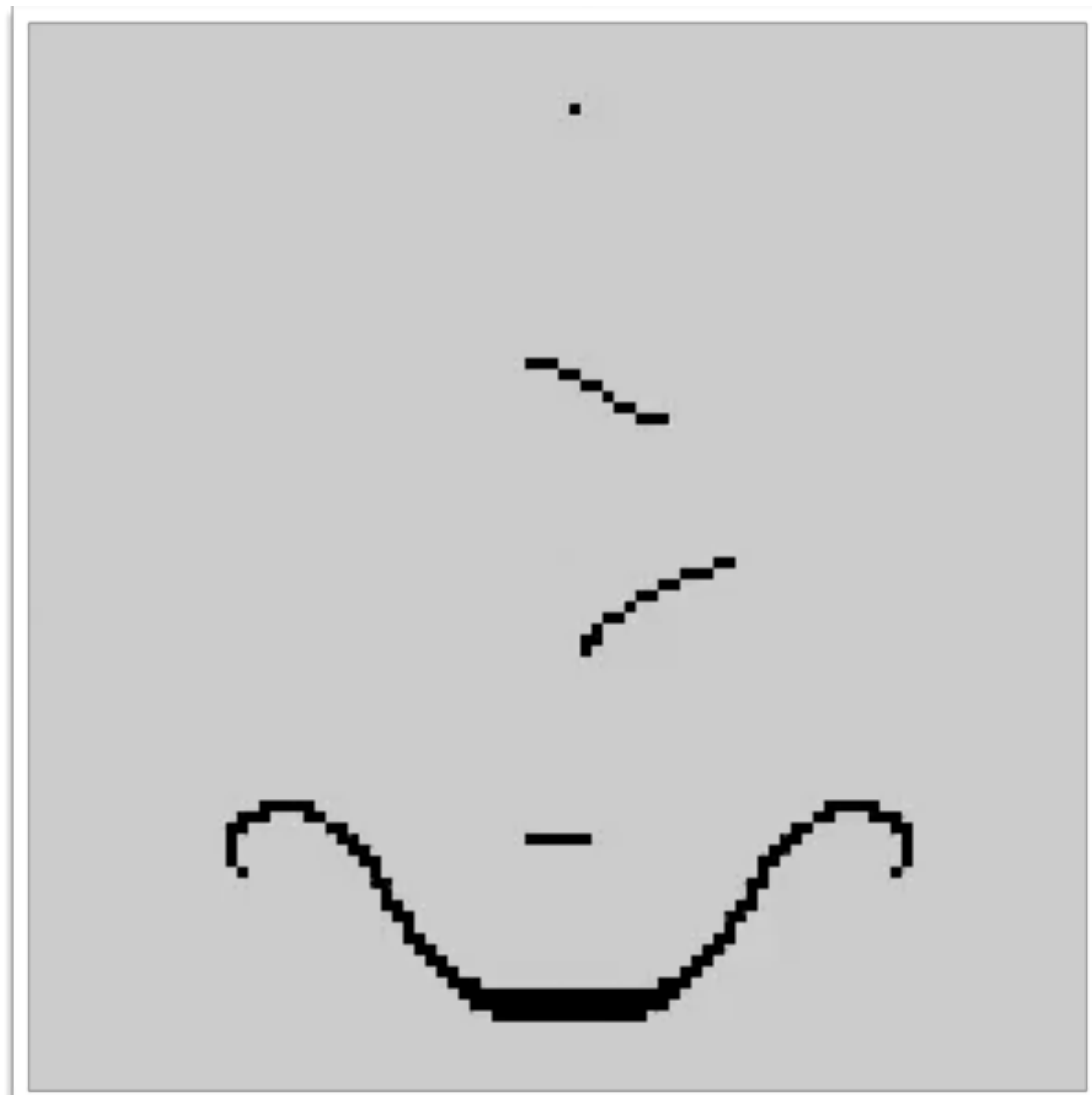


This kind of model permits the exploration of the behavior of *granular media*, which is difficult with conventional tools (e.g., PDEs)

Complex Systems



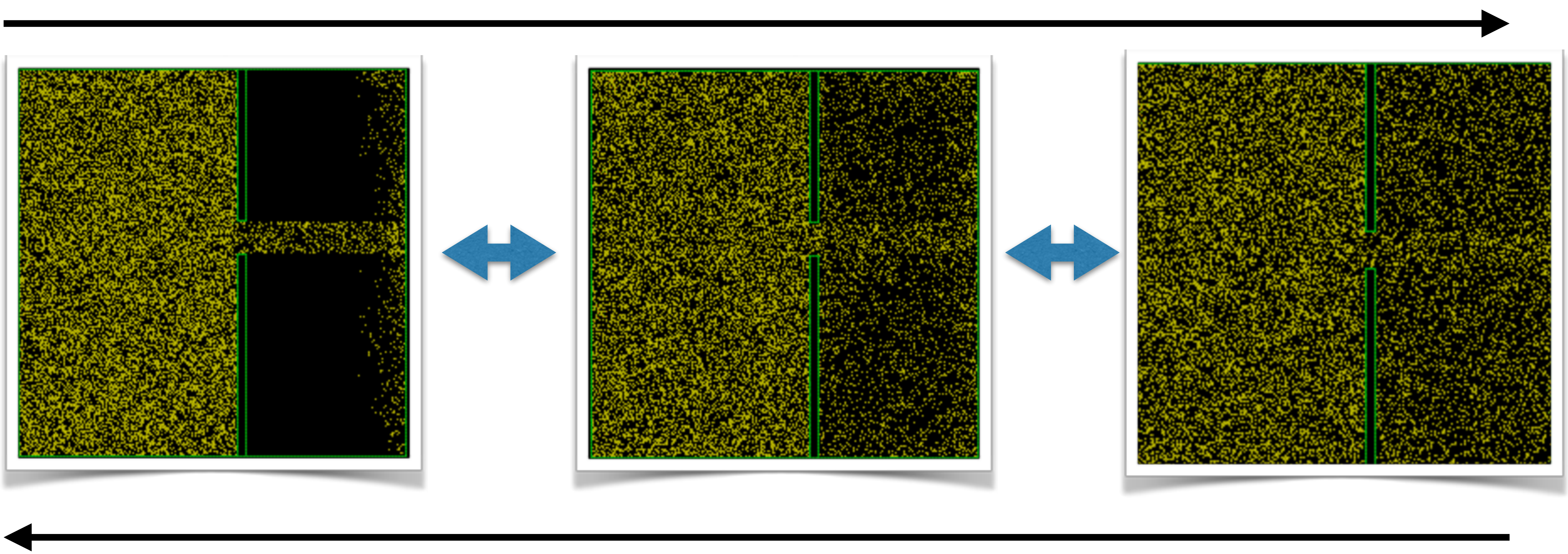




Reversibility

One of the interesting properties of CA is the possibility to display **exact reversibility.** Contrary to conventional numerical simulations, CA are not plagued by approximation errors.

At the **microscopic level** the **laws of physics are assumed as being reversible.** A particle CA can display invariance under time reversal. This means that no information is lost during the evolution of the CA. We can therefore observe very subtle effects.



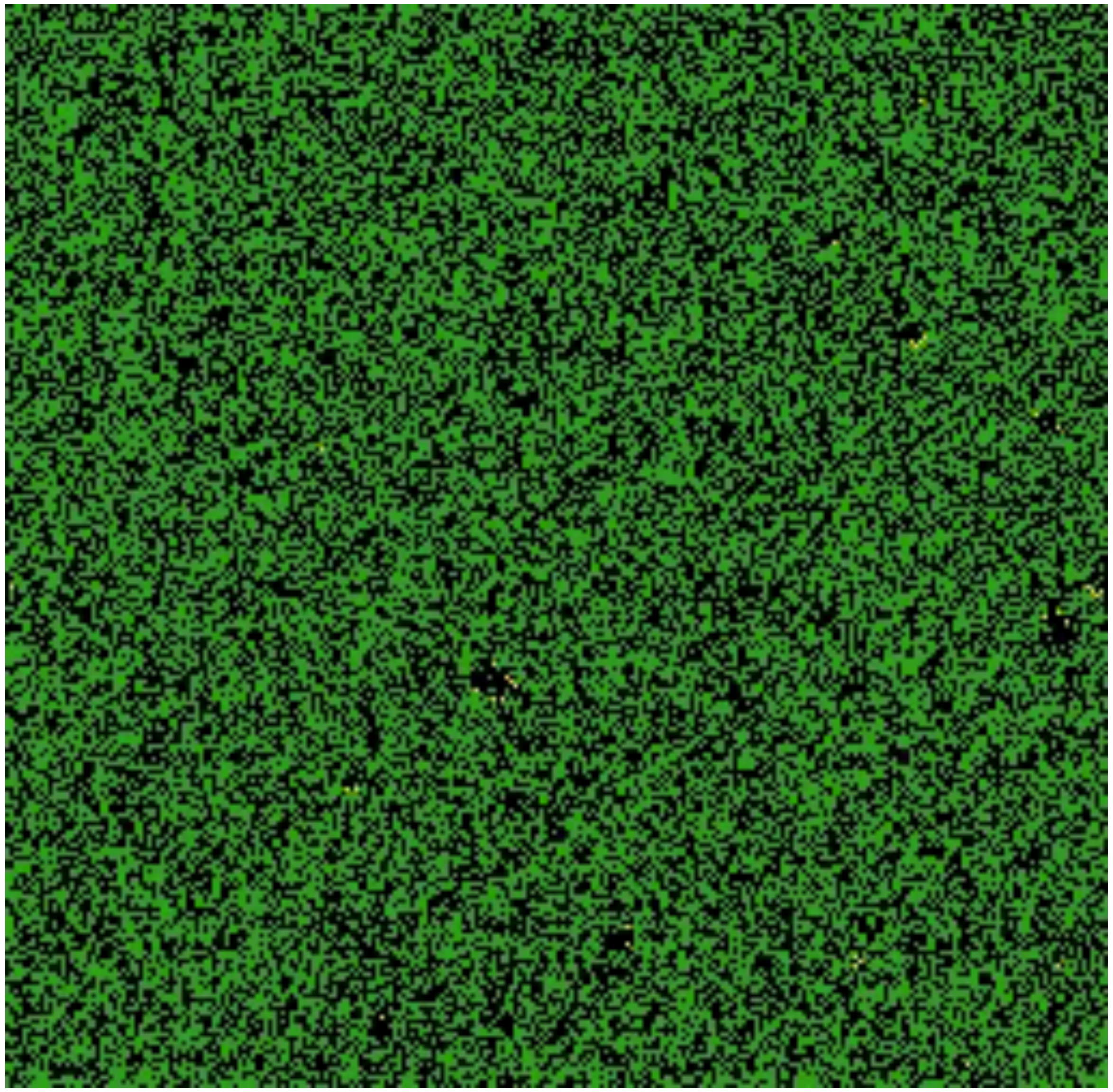
Probablistic CA

So far we have considered only **deterministic CA**.
To model many phenomena it is useful to **transition rules** that **depending** on some externally assigned **probability**

Example: The forest fire model:

- Each cell contains a green tree, a yellow burning tree, or is empty (black)
- A burning tree becomes an empty cell
- A green tree with at least a burning neighbour becomes a burning tree
- A green tree without burning neighbours becomes a burning tree with probability f (probability of lightning)
- An empty cell grows a green tree with probability g (probability of growth)

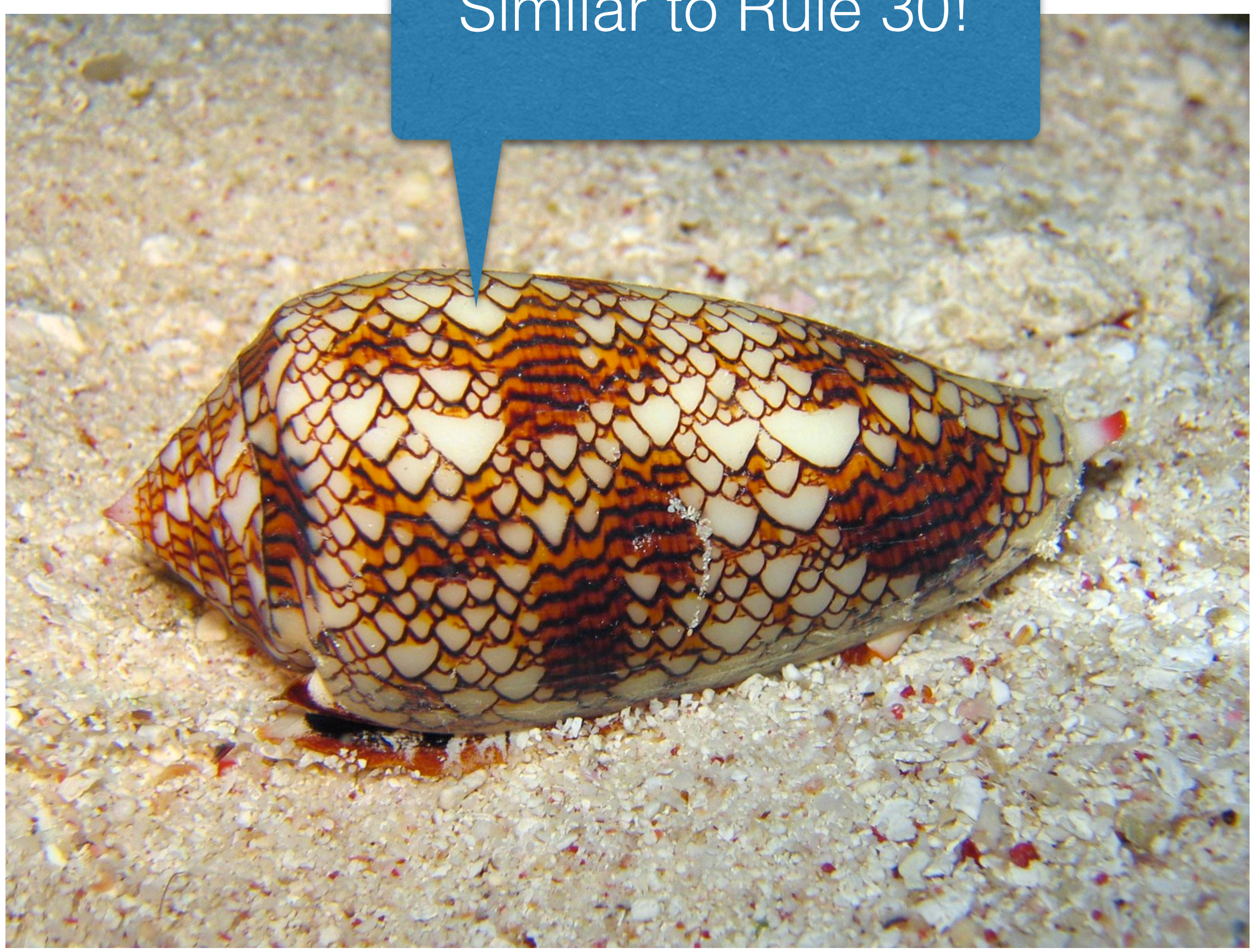
The parameters can be varied in a continuous range and introduce some "continuity" in the discrete world of CA models

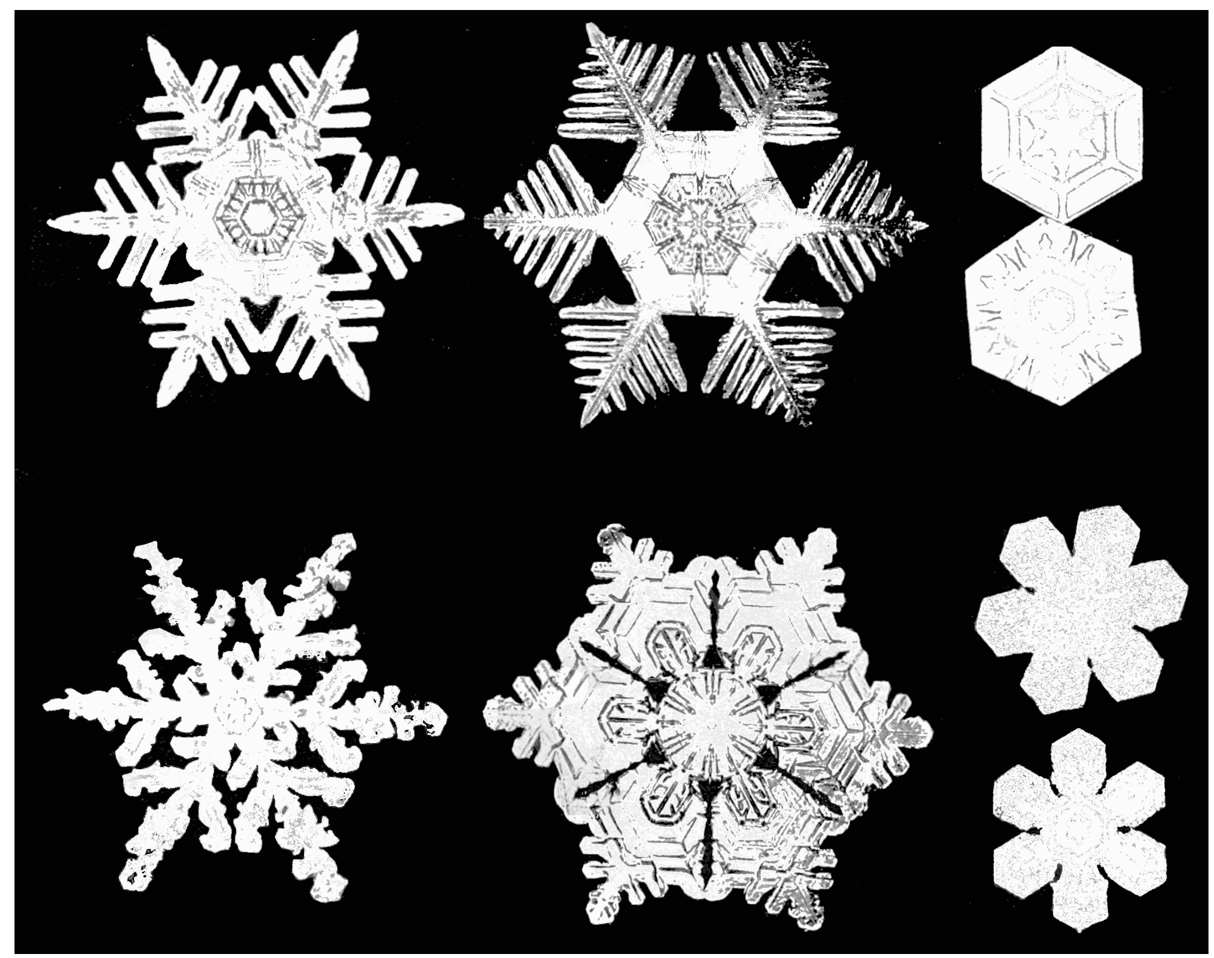


Structures and Patterns

One of the most fascinating aspects of biological and natural systems is the **emergence of complex spatial and temporal structures** and patterns from simple physical laws and interactions.

Cellular systems are an ideal tool for the analysis of the hypotheses about the local mechanisms of structure and pattern formation.



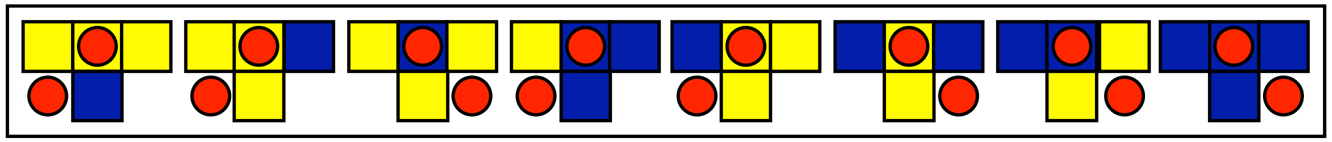


Variants and Extensions

The basic CA is **discrete** in space, time and state; updates all its cells **synchronously**; uses the **same neighbourhood geometry and transition rule** for all cells.

We can relinquish some of these prescriptions and obtain:

- **Asynchronous** CA (for example, mobile automata, where only one cell is active at each time step, and the transition rule specifies the fate of the activation)



- **Non-homogenous** (or non-uniform) CA
- Continuous-state CA (**Coupled Map Lattices**)
- Continuous-state and time CA (**Cellular Neural Networks**)
- ...

Questions

- What are the limitations of CAs?
- Are they good models for the real-world?
- Can everything be thought of as an automaton?
- Are there practical applications beyond the modelling world?
- Is the whole universe a CA?
- Ethics?