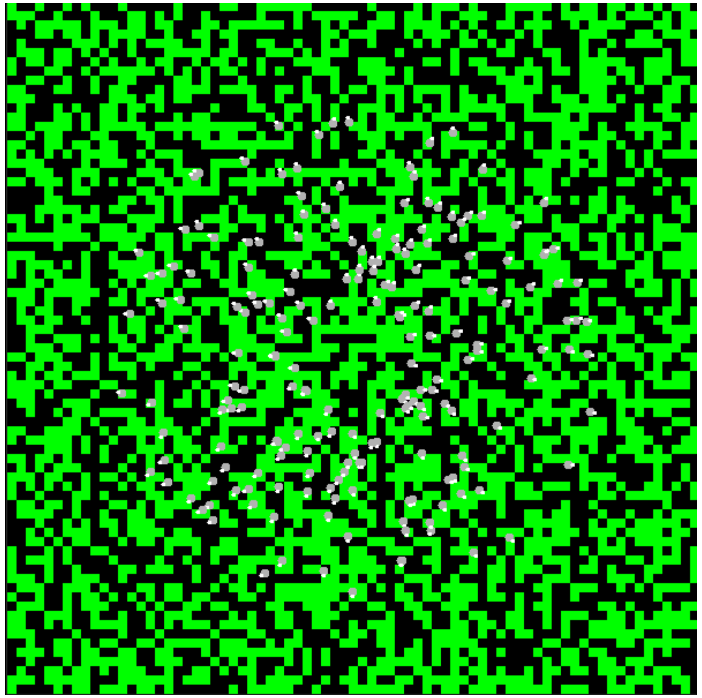
Introduction to Cellular Automata

BASIC CONCEPTS

Cellular Automata

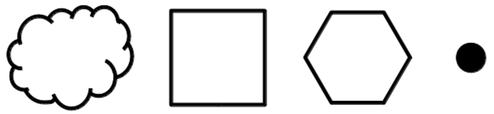
- Modelling using „cellular automata“, short CA, is a microscopic simulation method
- Cellular automata can be imagined as a coloured grid observed dynamically



Although this is a very simplified image of a CA, keep it in mind to understand the formal details of this concept

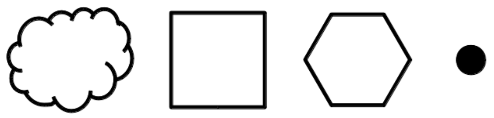
Components of a CA

- Cells



Components of a CA

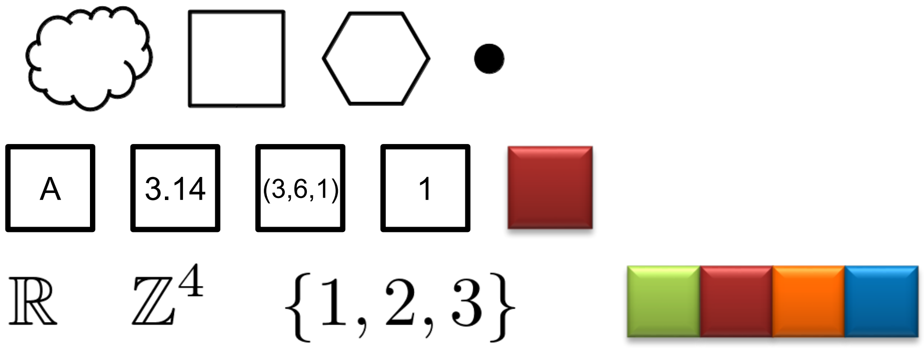
- Cells



  - Notations: cell, entity, node
  - Cells are passive: no internal dynamic, only container for some information
  - Each cell has some state.

Components of a CA

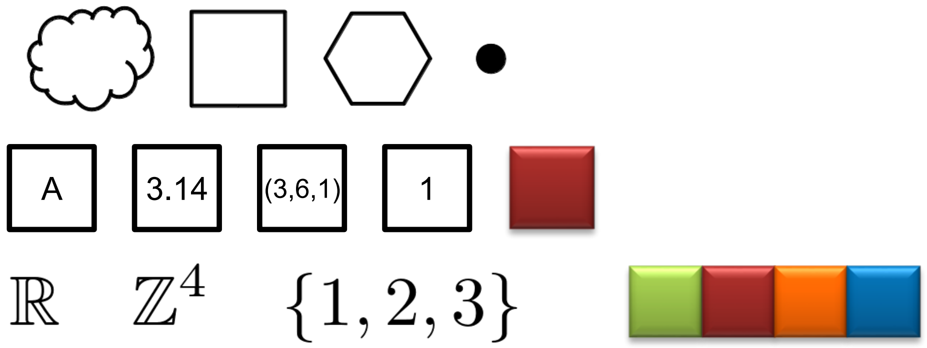
- Cells
- States
- State-space

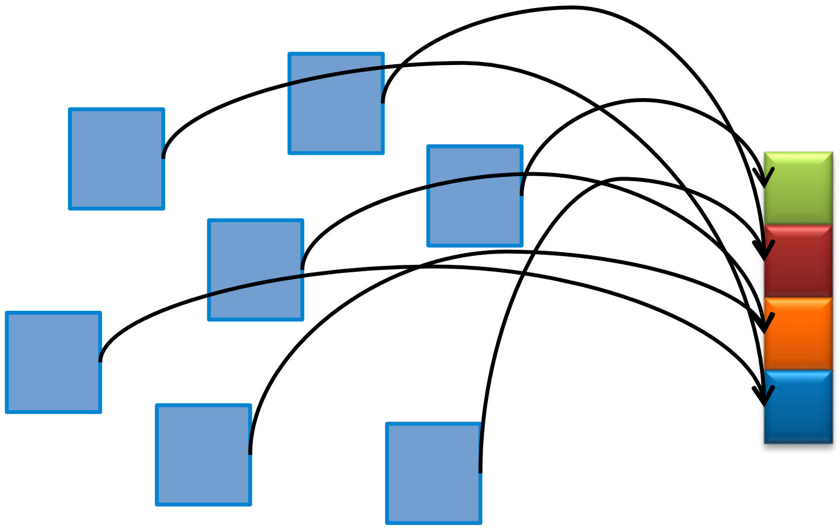


- Every Cell has a state
- There is always some space $\mathbb{S}$ that contains all possible states. It is usually called state-space.

Components of a CA

- Cells
- States
- State-space

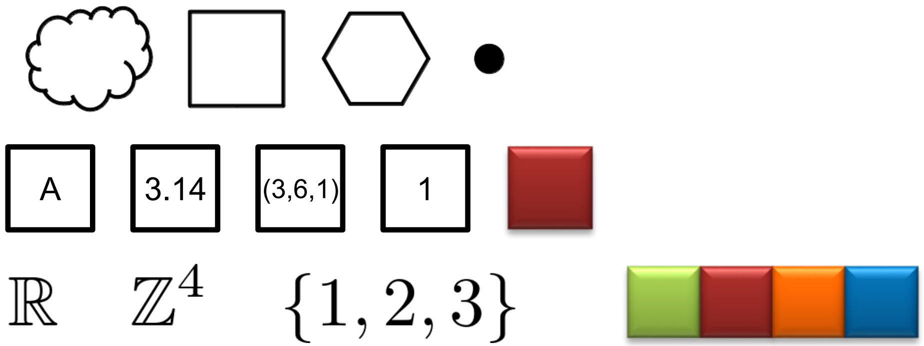


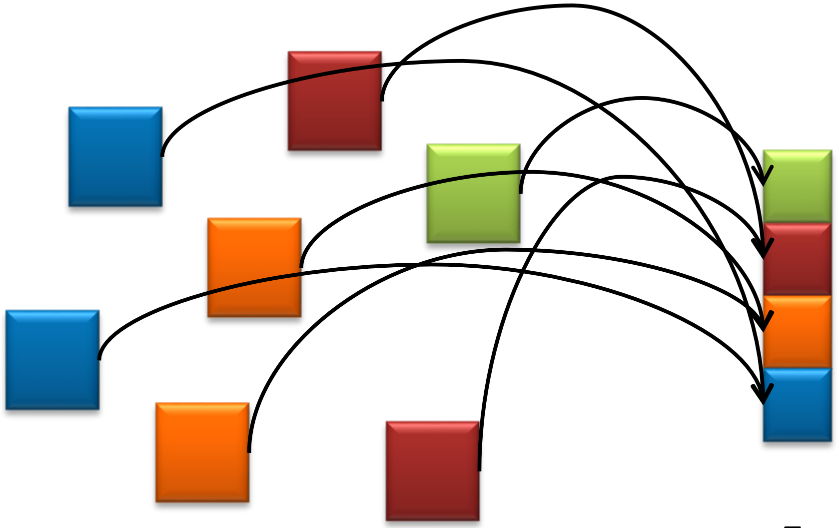


Every cell has a state from a common state-space

Components of a CA

- Cells
- States
- State-space

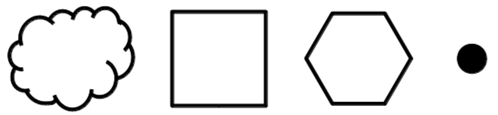




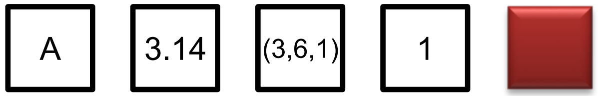
Every cell has a state from a common state-space

Components of a CA

- Cells



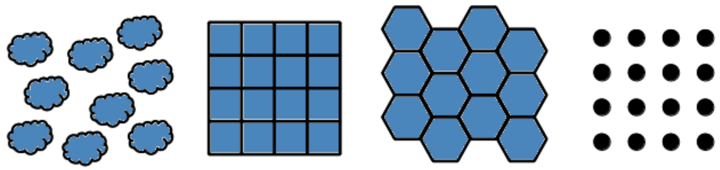
- States



- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

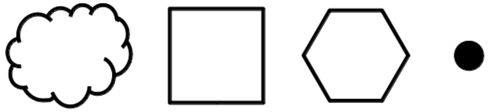
- Arrangement (Cell-space)



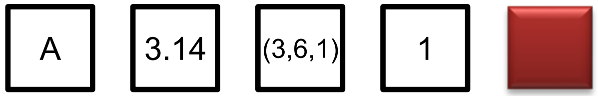
- All cells are arranged on some lattice structure: the „cell-space“ – in the simplest case, a rectangular grid.
- There is some index mapping that maps some subset of $\mathrm{I} \subset \mathbb{Z}^d$ onto each cell

Components of a CA

- Cells



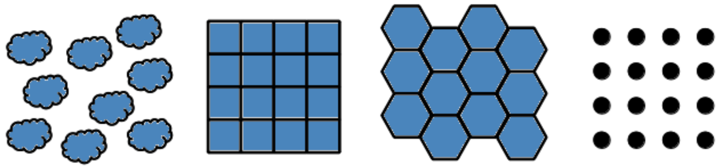
- States

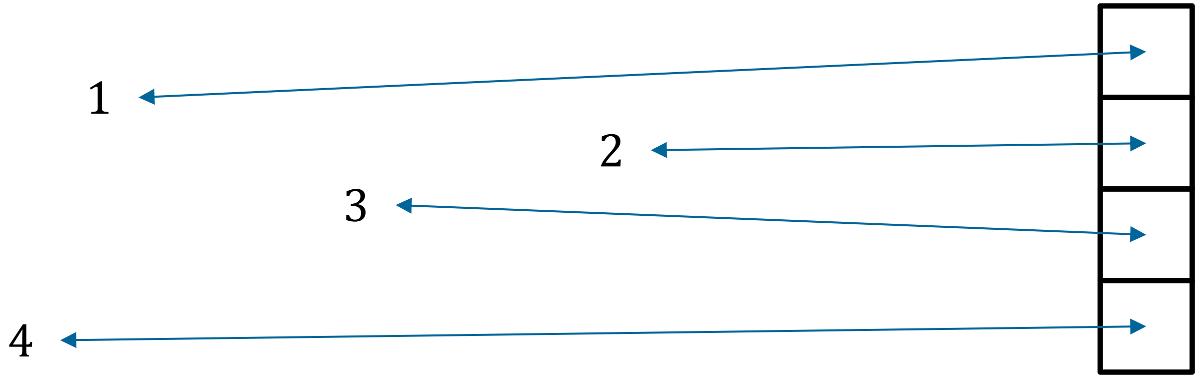


- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

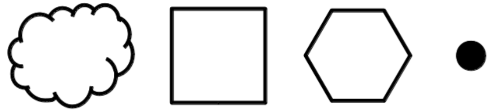
- Arrangement (Cell-space)



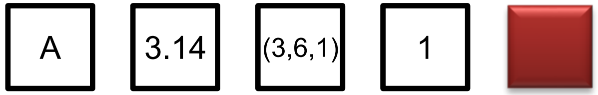


Components of a CA

- Cells



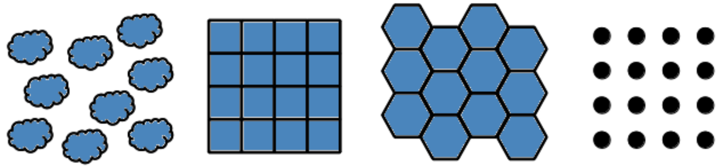
- States

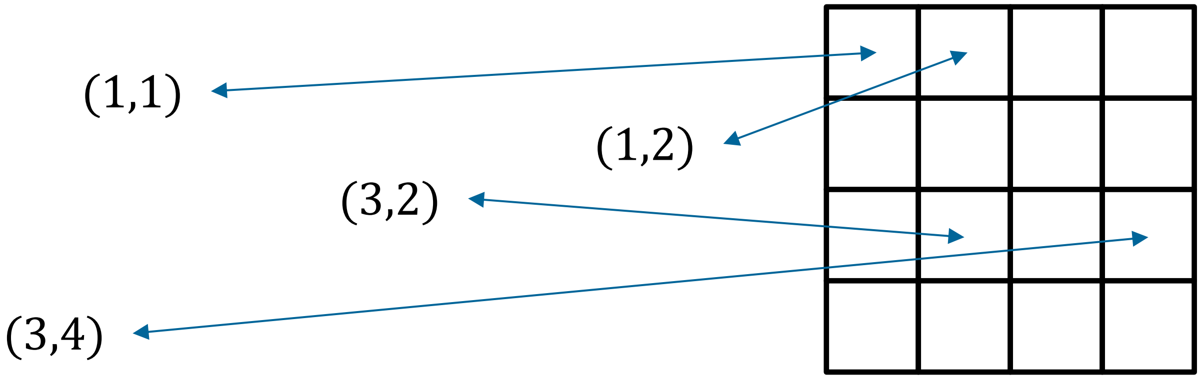


- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

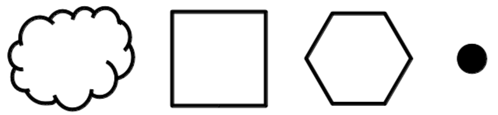
- Arrangement (Cell-space)

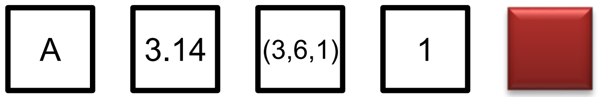




Components of a CA

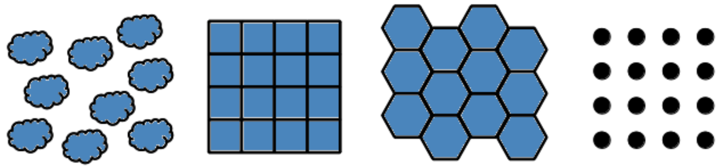
- Cells
- States
- State-space

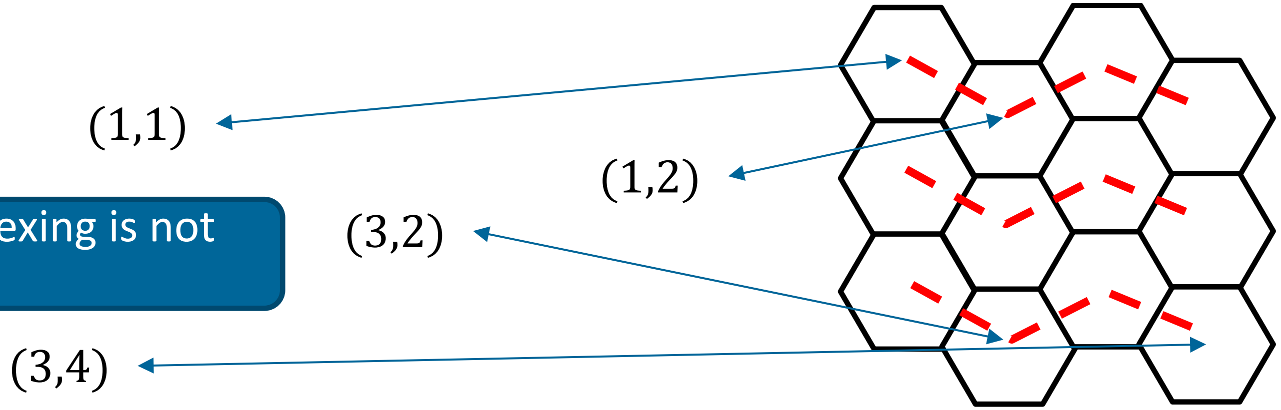




$\mathbb{R}$ $\mathbb{Z}^4$ $\{1,2,3\}$

- Arrangement (Cell-space)

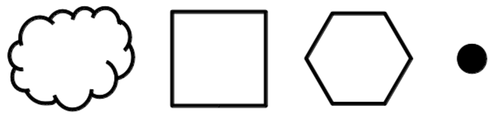




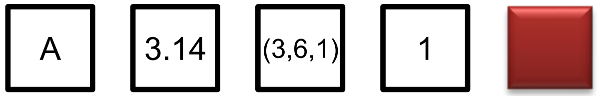
Sometimes indexing is not so trivial…

Components of a CA

- Cells



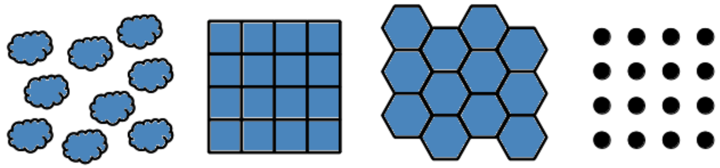
- States



- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

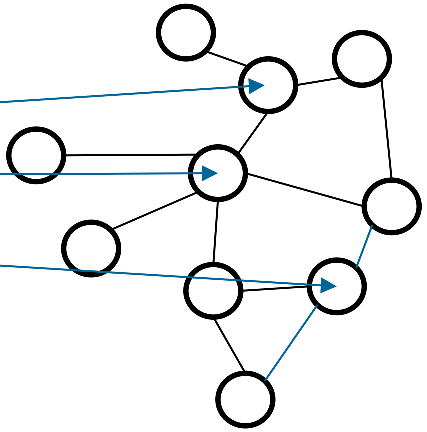
- Arrangement (Cell-space)



(1,1)

(1,2)

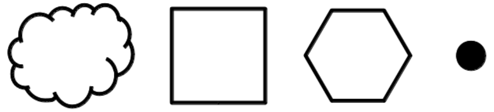
(3,2)



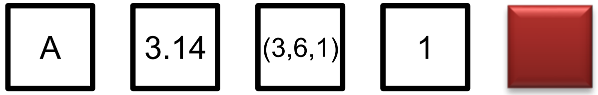
It often is, but does not necessarily have to be a natural attribute of the cell-space…

Components of a CA

- Cells



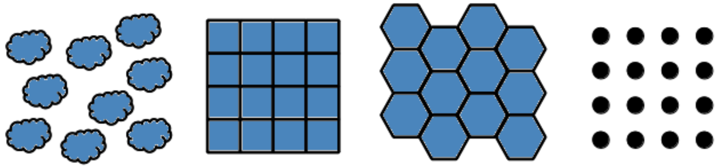
- States



- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1,2,3\}$

- Arrangement (Cell-space)

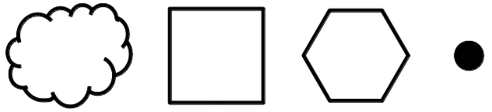


▪ Possible characteristics of the index set:
  - regular
  - finite or infinite
  - connected
  - multi-dimensional

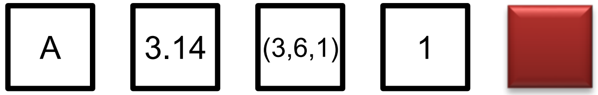
▪ Interpretation of the index set: discretisation of a space or spatial arrangement of entities

Components of a CA

- Cells



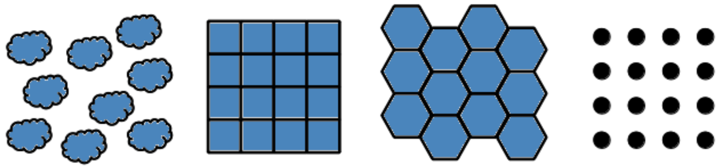
- States



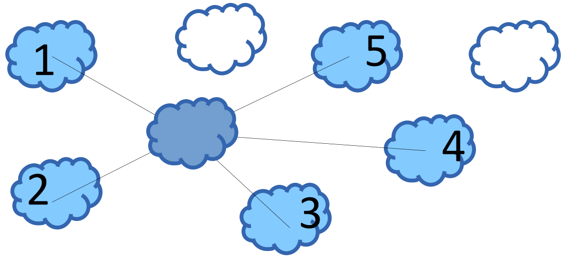
- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

- Arrangement (Cell-space)



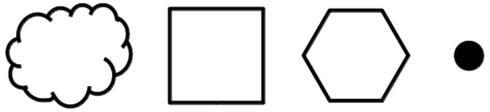
- Neighbourhood



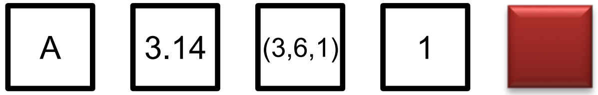
The neighborhood of a cell *z* is an ordered set of *n* other cells $(z_1, \ldots, z_n)$.

Components of a CA

- Cells



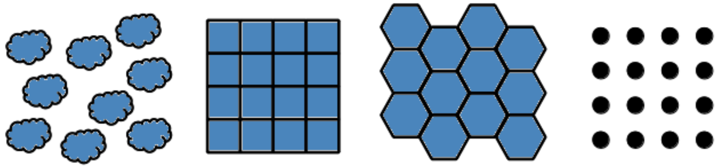
- States



- State-space

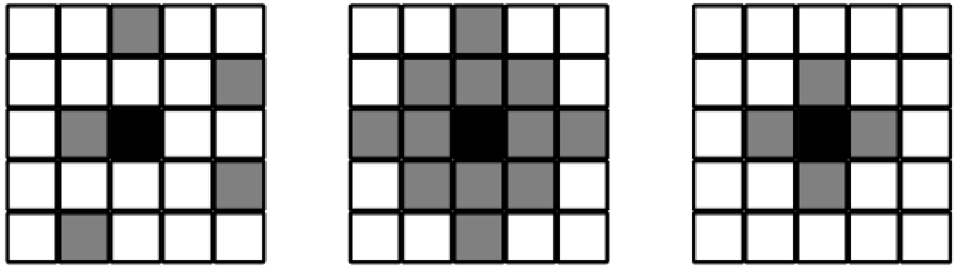
$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

- Arrangement (Cell-space)



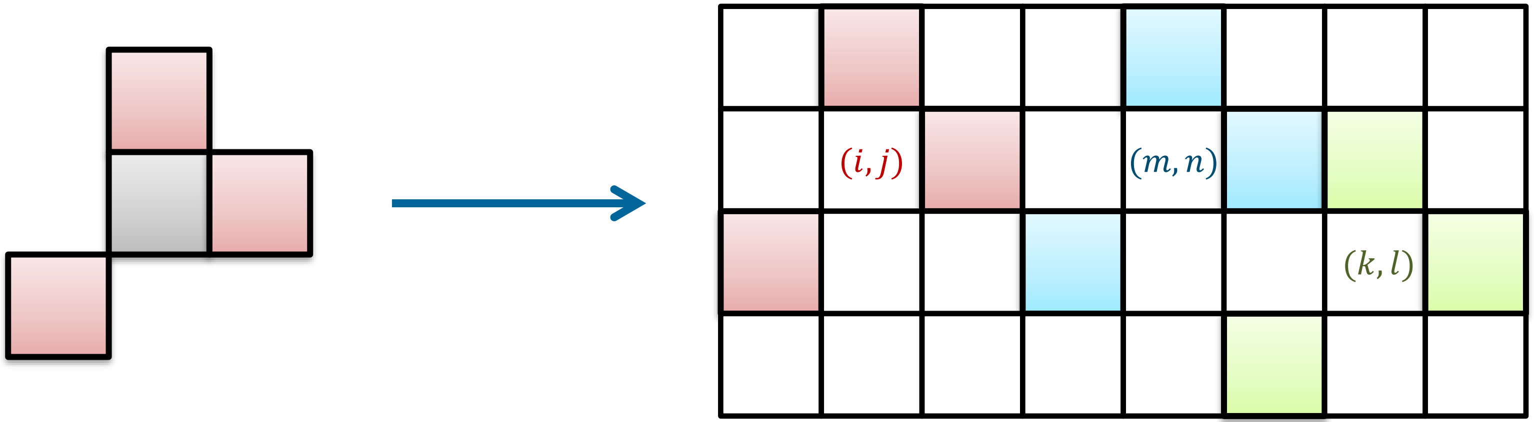
- Neighbourhood

Some examples:



Neighbourhood

- The neighbourhood mapping is relative to the cell's position (= index)



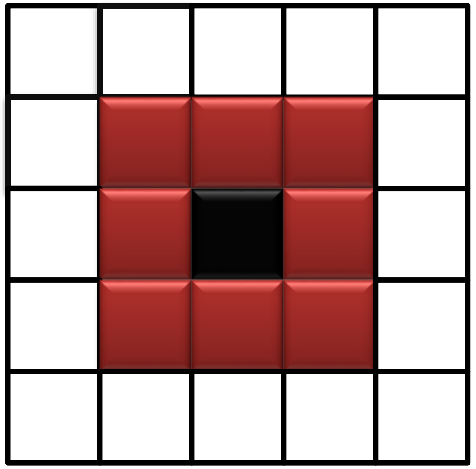
- Calculation of neighbouring cells by stencil: Index translations yield the positions (index) of *n* neighboring cells: $\vec{i} \mapsto (\vec{i} + \vec{t_1}, \dots \vec{i} + \vec{t_n})$

Neighbourhood

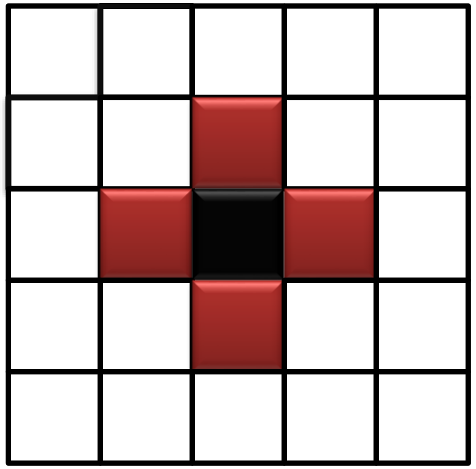
- Possible characteristics of neighbourhoods:
  - **local:** the neighbourhood consists of cells of neighboring points on the grid
  - **symmetric:** the neighborhood of cell A contains cell B if and only if the neighborhood of cell B contains cell A

Neighbourhood

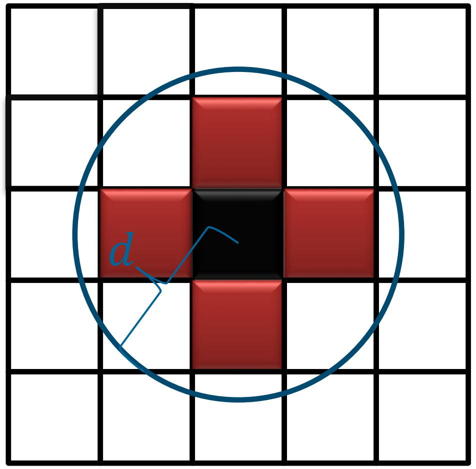
- Classic, popular neighborhoods



Moore
neighborhood



Von-Neumann
neighborhood

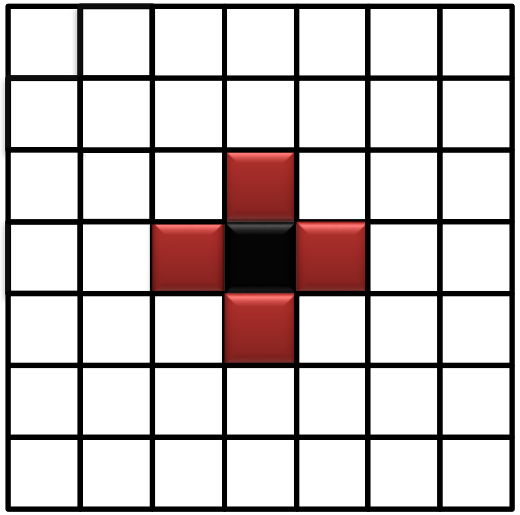


Neighbourhood by distance:
$\vec{i} \to \{\vec{j} : |\vec{i} - \vec{j}| < d\}$

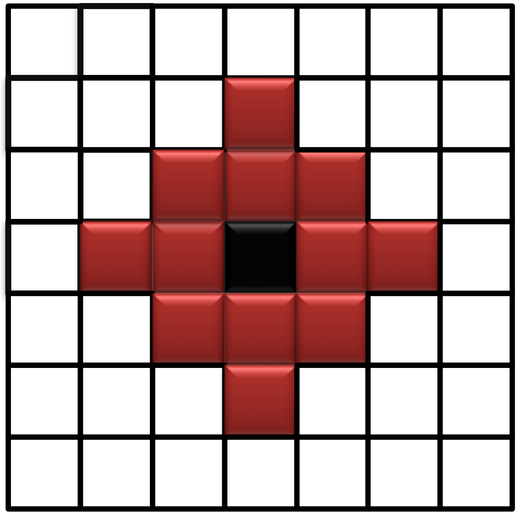
Neighbourhood

- Von Neumann/Moore Neighbourhood of higher order

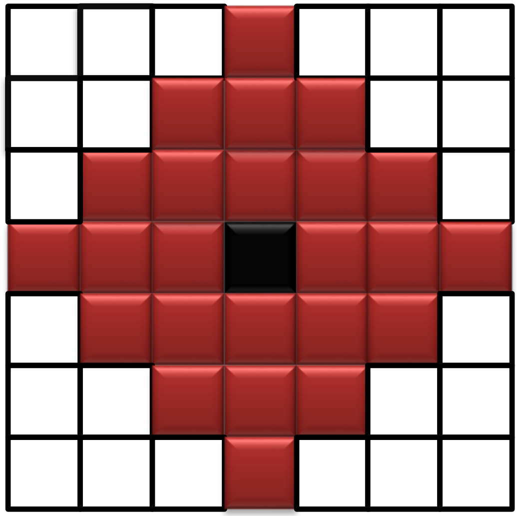
Von-Neumann neighborhood 1st order



Von-Neumann neighborhood 2nd order

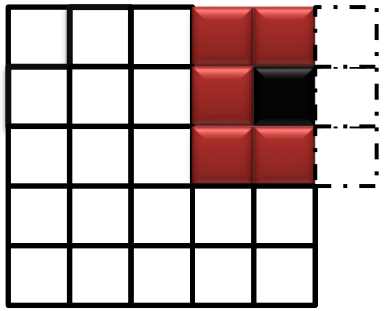


Von-Neumann neighborhood 3rd order

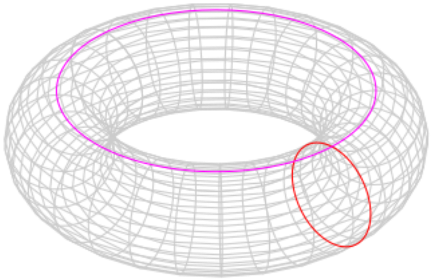


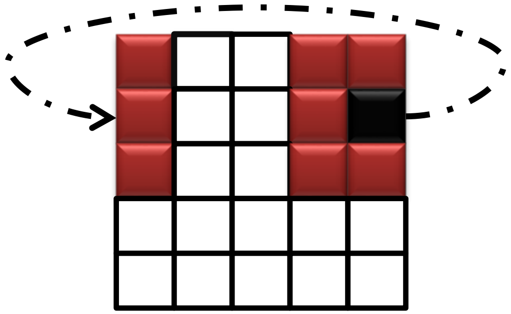
Neighbourhood

- The index set is limited → either incomplete neighborhoods for cells near the borders $(z_1, z_2, \emptyset, z_4 \ldots, z_n)$....



...or other compensation ideas.

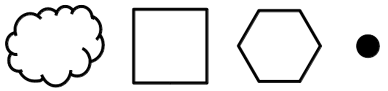




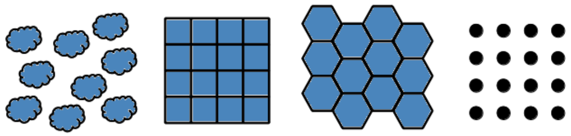
Periodic Boundary Conditions (Torus)

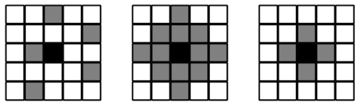
Components of a CA

- Cells
- States
- State-space
- Arrangement (Cell-space)
- Neighbourhood
- Update Rule



$\mathbb{R}$ $\mathbb{Z}^4$ $\{1,2,3\}$

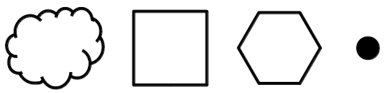




Some rule, that simultaneously updates all states of all cells of the CA.
Maps all states of a cell's neighbourhood to a new state for the cell.

Components of a CA

- Cells

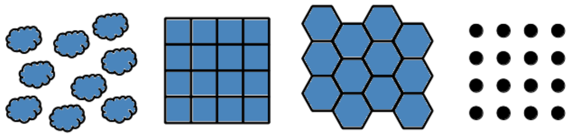


- States

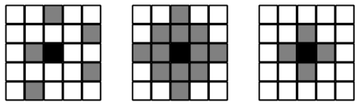
- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

- Arrangement (Cell-space)



- Neighbourhood



- Update Rule

$$f(s, s_1, \ldots, s_n) = s_{new}$$

state of the cell

state of the (ordered) neighbors

new state of the cell

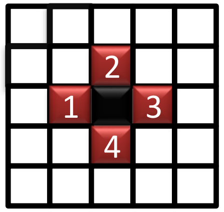
Stochastic CAs have stochastic updates!

Update Rule

- Example:

Neighbourhood = Von Neumann

$$f(s, s_1, s_2, s_3, s_4) = \sum s \;\; (mod\ 4)$$



Old state of the CA

1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	0
2	0	1	1	1	3	1
2	2	0	2	3	1	2

New state of the CA

Update Rule

- Example:

Neighbourhood = Von Neumann

$$f(s, s_1, s_2, s_3, s_4) = \sum s \ (mod\ 4)$$

		2		
	1		3	
		4		

Old state of the CA

1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	0
2	0	1	1	1	3	1
2	2	0	2	3	1	2

$$f(s, s_1, s_2, s_3, s_4) =$$
$$= 1 + 1 + 2 + 1 + 0\ (mod 4) =$$
$$= 5\ (mod\ 4) = 1$$

New state of the CA

			1			

Update Rule

- Example:

Neighbourhood = Von Neumann

$$f(s, s_1, s_2, s_3, s_4) = \sum s \ (mod\ 4)$$

		2		
	1		3	
		4		

Old state of the CA

1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	0
2	0	1	1	1	3	1
2	2	0	2	3	1	2

$$f(s, s_1, s_2, s_3, s_4) =$$
$$= 1 + 1 + 2 + 3 + 3\ (mod 4) =$$
$$= 10\ (mod\ 4) = 2$$

New state of the CA

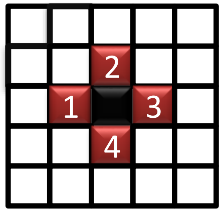
			1			
					2	

Update Rule

- Example:

Neighbourhood = Von Neumann

$$f(s, s_1, s_2, s_3, s_4) = \sum s \ (mod\ 4)$$



Old state of the CA

1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	[illegible]
2	0	1	1	1	3	[illegible]
2	2	0	2	3	1	[illegible]

$$f(s, s_1, s_2, \emptyset, s_4) =$$
$$= 1 + 1 + 1 + 1\ (mod 4) =$$
$$= 4\ (mod\ 4) = 0$$

New state of the CA

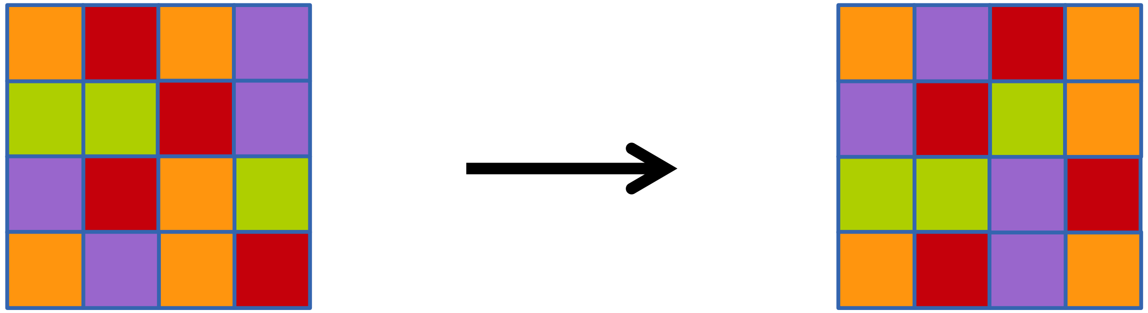
						0
			1			
					2	

The update function needs to be capable to deal with incomplete neighbourhoods as well

Updates

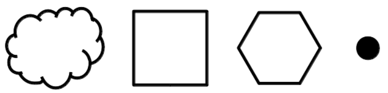
- Updates happen for all cells simultaneously.

  Why? ighborhoods are all computed from the same system state
  - Update order of cells is irrelevant



Components of a CA

- Cells

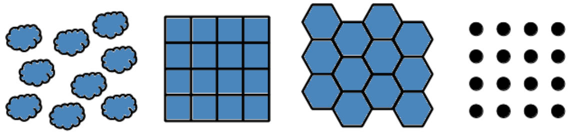


- States

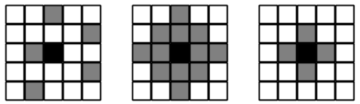
- State-space

$\mathbb{R}$ $\mathbb{Z}^4$ $\{1, 2, 3\}$

- Arrangement (Cell-space)



- Neighbourhood

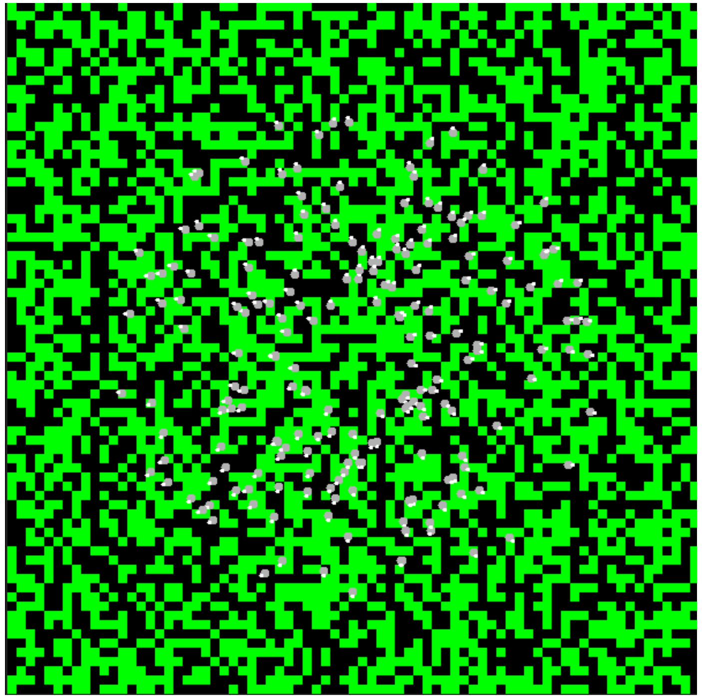


- Update Rule

$$f(s, s_1, \dots, s_n) = s_{new}$$

- Iteration

Iteratively apply the update rule on the complete CA



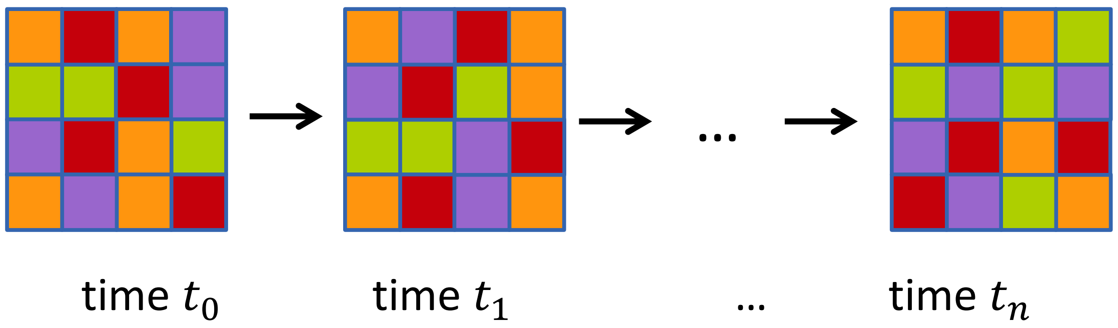
Iterations

- Define discrete, equidistant time points (all time steps between time points are of the same length): $t_0, t_1, \ldots, t_n$
- Every update of states brings the model to the next time point

➢ Cellular Automaton (CA)

Iterations

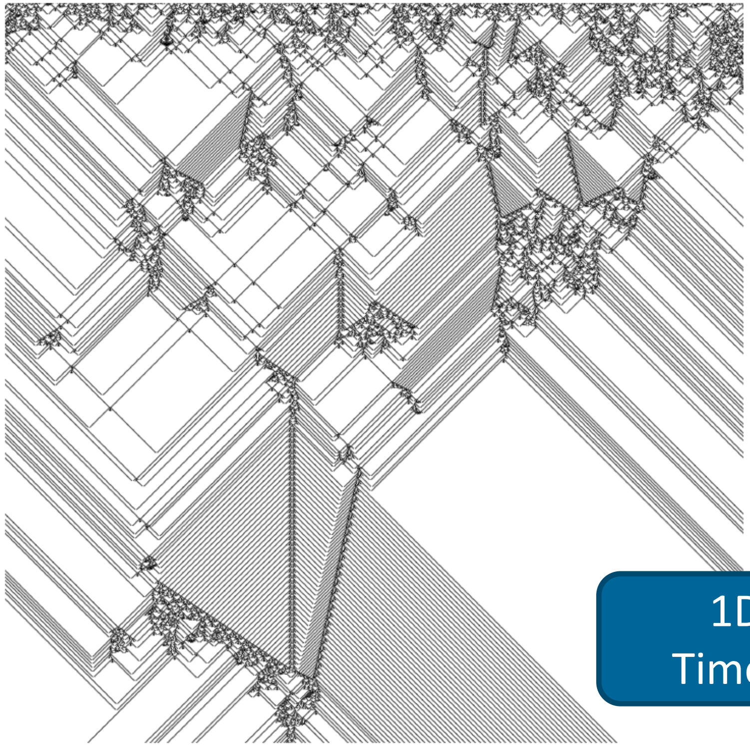
- Tasks for one iterations
  - Compute the neighbors of all cells
  - Determine states of all cells, and states of all neighbours of all cells
  - Compute state updates for all cells and store them
  - Apply the updated states for all cells



Properties of CA models

Cellular Automata are microscopic simulation models that are capable of producing almost arbitrarily complex, up to chaotic, behaviour.

They are, hence, not only a very powerful, but also a very dangerous modelling approach with respect to validity.



1D deterministic CA!
Time is plotted vertically

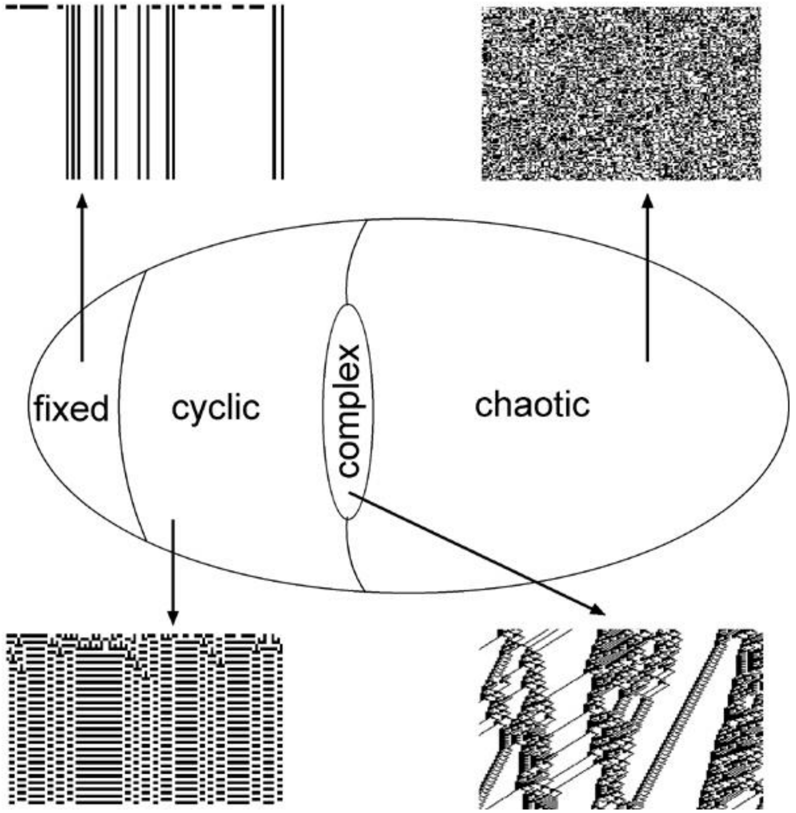
Properties of CA models

Cellular Automata are microscopic simulation models that are capable of producing almost arbitrarily complex, up to chaotic, behaviour.

Stephen Wolfram
(A New Kind of Science, 2002)
stated that CAs may have one of the four types of behavour:

fixed, cyclic, complex, chaotic

Chris Langton developed the schematic to the right.

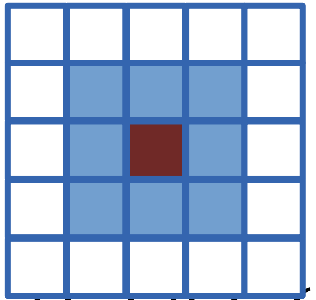


Example:

CONWAY’S GAME OF LIFE

Conway‘s Game of Life

- Cells on a 2-dimensional, rectangular or infinite lattice: $I = (1,2, \dots a) \times (1,2, \dots, b)$ or on $I = \mathbb{Z}^2$.
- Set of states: $\mathbb{S} = (alive \quad , dead \quad)$
- Moore neighborhood

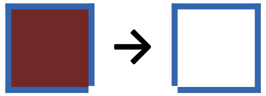


Index translations:

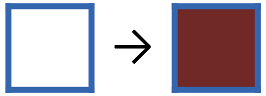
$$\left(\binom{1}{0}, \binom{1}{1}, \binom{0}{1}, \binom{-1}{1}, \binom{-1}{0}, \binom{-1}{-1}, \binom{0}{-1}, \binom{1}{-1}\right)$$

Conway‘s Game of Life

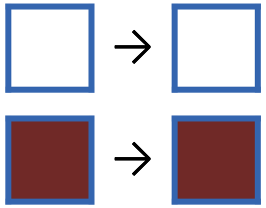
- Update rules:
  - An alive cell with fewer than two or more than three alive neighbors dies (“under-population” or “overcrowding”)



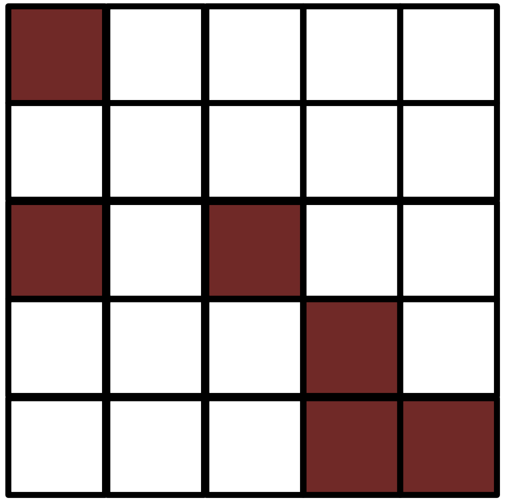
  - A dead cell with exactly three alive neighbors becomes alive (“reproduction”)



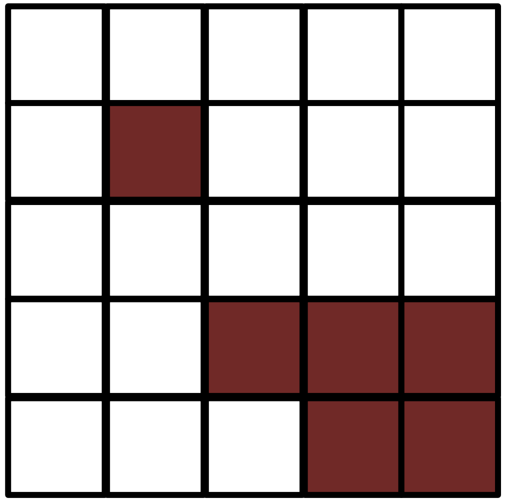
  - Cells keep their state in any other case



Conway‘s Game of Life



time $t=0$



time $t=1$

Conway's Game of Life

- Designed by John Horton Conway, 1970
- Why "Game of Life"?
  - Teaching purposes
  - Academic competitions
  - Fundamental/methodological research
  - Game → figures

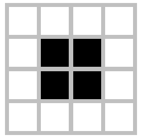
Probably ***worst*** example for a Cellular Automata simulation **model**,…

…but probably the ***best*** example to show the concepts of CAs.

Conway‘s Game of Life

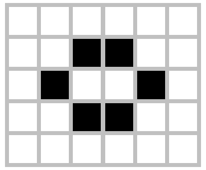
Pattern analysis of the Game of Life became its own science (although its applicability can be doubted).

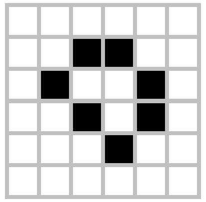
Static figures



Block

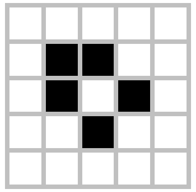
Beehive





Loaf

Boat

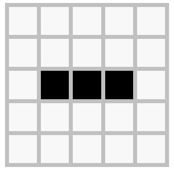


Source: http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Conway‘s Game of Life

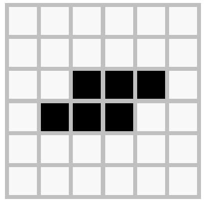
Pattern analysis of the Game of Life became its own science (although its applicability can be doubted).

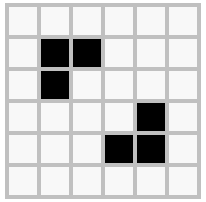
Oscillators



Blinker (period 2)

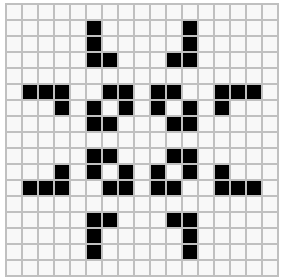
Toad (period 2)





Beacon (period 2)

Pulsar (period 3)



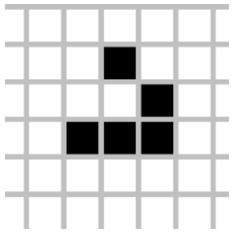
Source: http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Conway‘s Game of Life

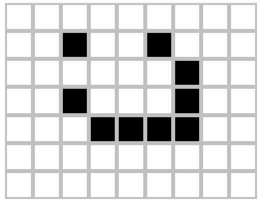
Pattern analysis of the Game of Life became its own science (although its applicability can be doubted).

Gliders (moving objects)

Lightweight spaceship (LWSS)



Glider



Source: http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Conway's Game of Life

Pattern analysis of the Game of Life became its own science (although its applicability can be doubted).

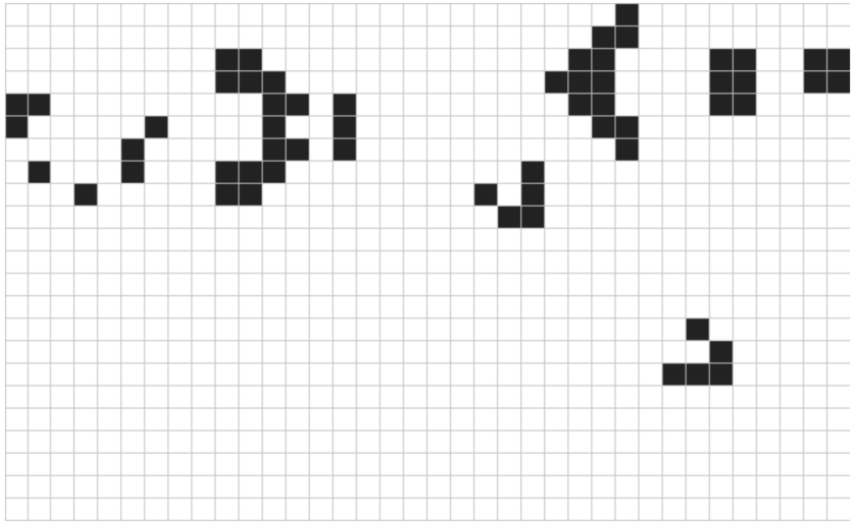
As it seemed as if any starting configuration of the GoL resulted in a static or oscillating end-configuration, Conway offered a price of 50$ for a pattern that resulted in an infinitely growing population.

Source: http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Conway‘s Game of Life

Pattern analysis of the Game of Life became its own science (although its applicability can be doubted).

Bill Gosper‘s answer:



Gosper Glider Gun

Source: http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Example

NAGEL SCHRECKENBERG MODEL

Application Example: Traffic Simulation

Nagel-Schreckenberg-Model

- discretisation of a road or motorway into cells of approximately 4m
- possible states:
  - $s = 0$: no vehicle
  - $s > 0$: speed of vehicle
- update rules (implicitly defined!):
  - accelerate: **IF** $v < v\_\max$ **AND** next vehicle $v + 1$ cells away **THEN** $v(t+1) = v(t) + 1$
  - brake: **IF** next vehicle j cells away **AND** $j < v$ **THEN** $v(t+1) = j - 1$
  - randomisation: $v(t+1) = v(t) - 1$ **with a certain probability**
  - movement: $s(t+1) = s(t) + v(t)$

Application Example: Traffic Simulation

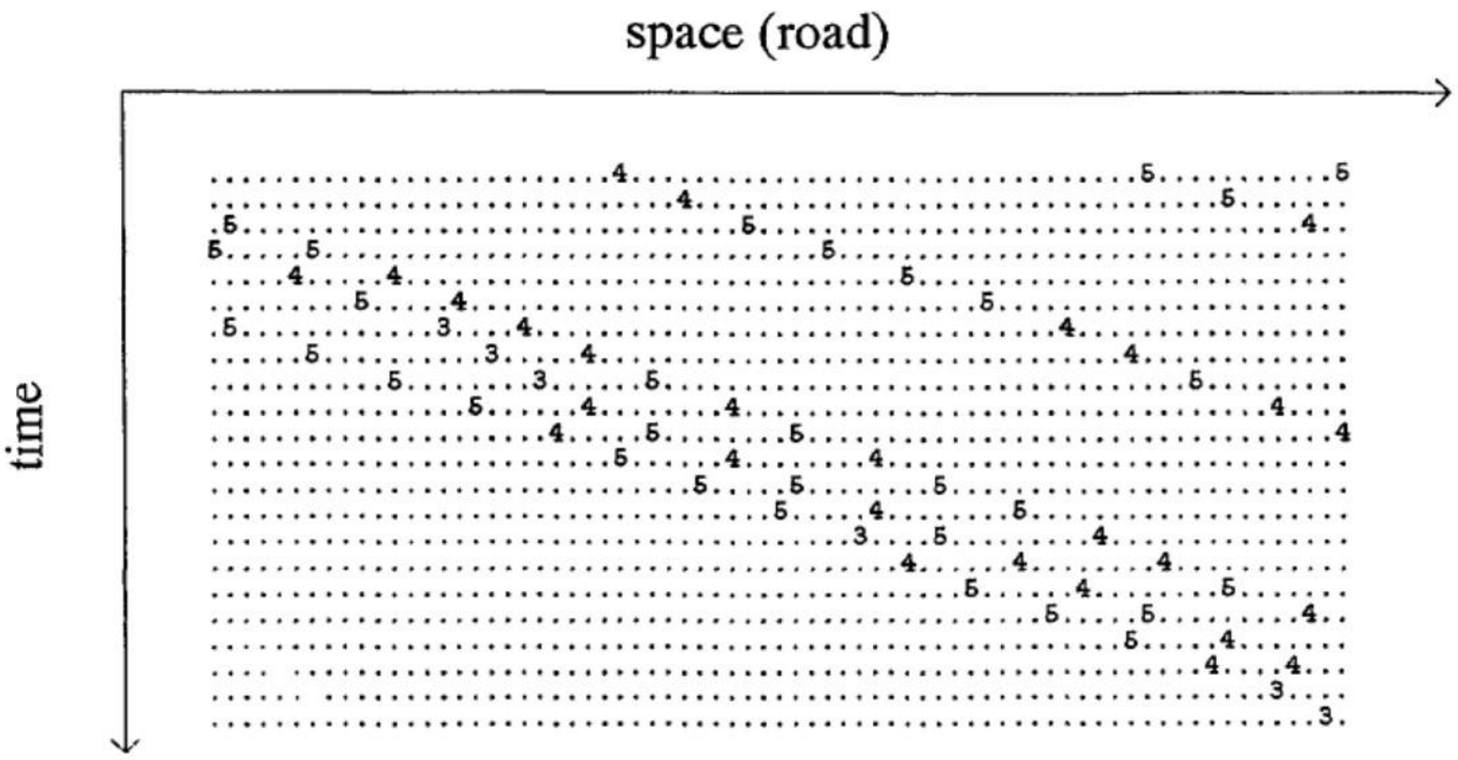


Fig.1. — Simulated traffic at a (low) density of 0.03 cars per site. Each new line shows the traffic lane after one further complete velocity-update and just before the car motion. Empty sites are represented by a dot, sites which are occupied by a car are represented by the integer number of its velocity. At low densities, we see undisturbed motion.

Application Example: Traffic Simulation

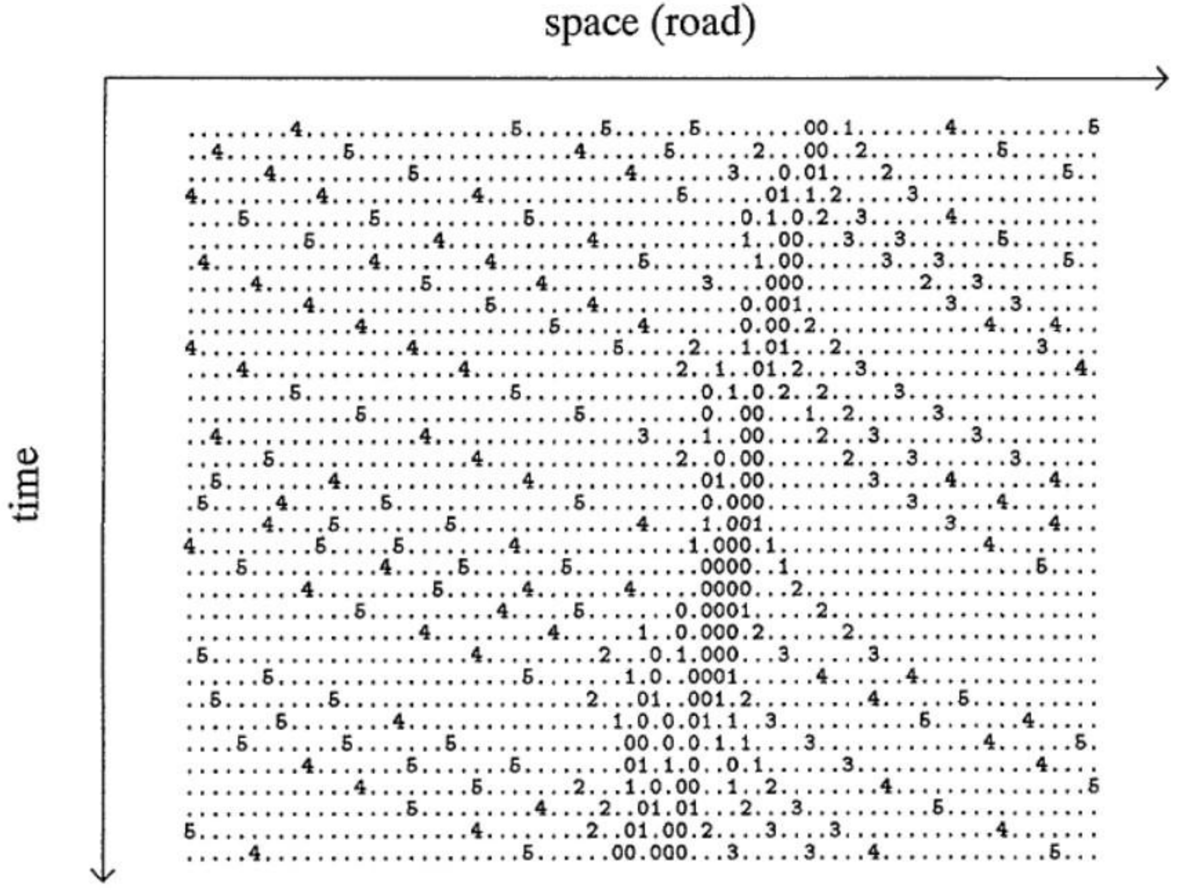


Fig.2. — Same picture as figure 1, but at a higher density of 0.1 cars per site. Note the backward motion of the traffic jam.

Application Example: Traffic Simulation

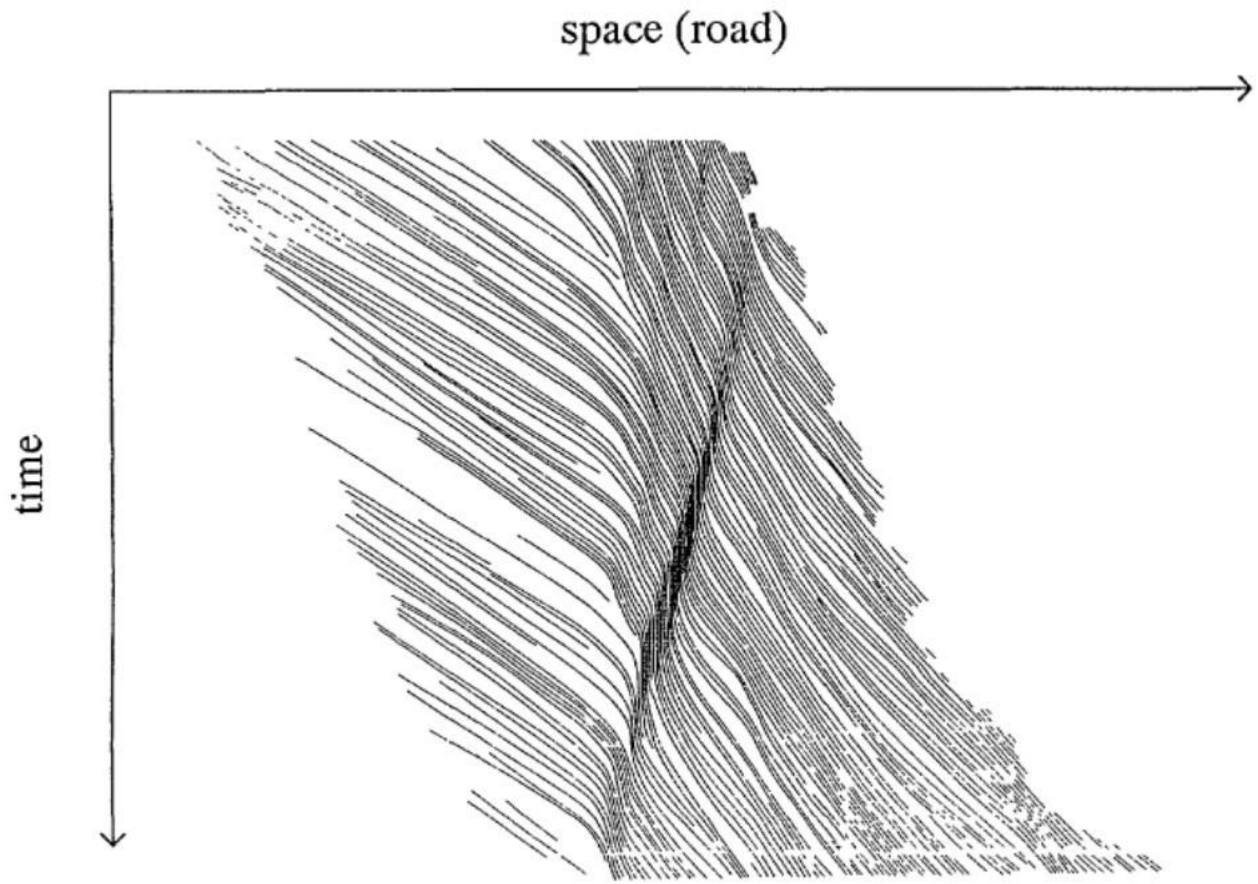


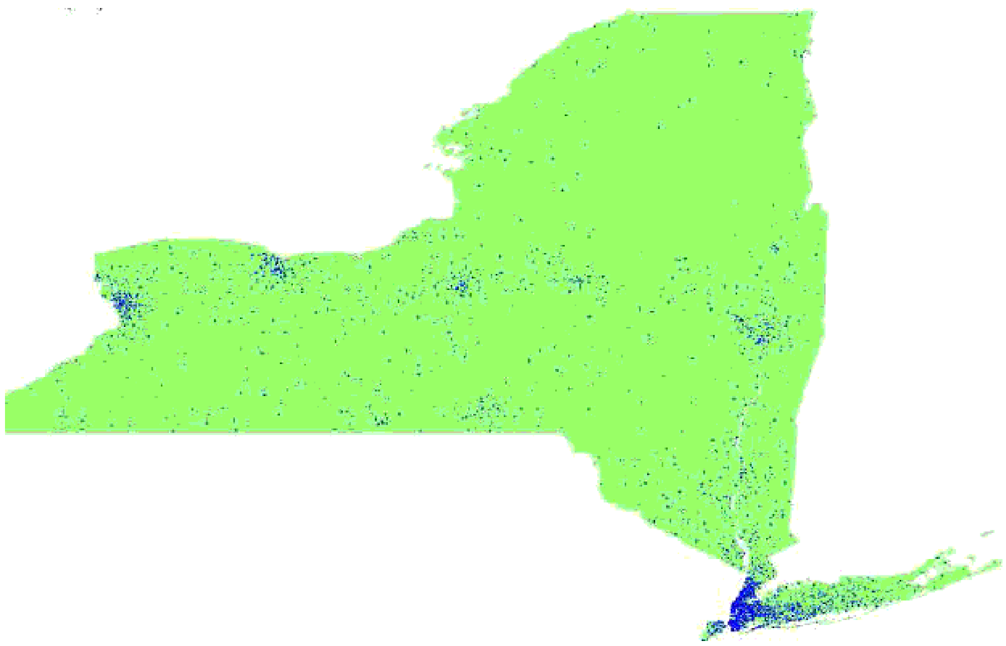
Fig.3. — Space-time-lines (trajectories) for cars from Aerial Photography (after [16]). Each line represents the movement of one vehicle in the space-time-domain.

Example

DYNAMIC MAPS

Dynamic Cartography

- map shows relation between sizes
- The dots symbolises cancer patients



Dynamic Cartography

aufzuteilen. Je kleiner man diese Regionen wählt, desto genauer ist die entstandene Karte. Der einzige Nachteil dieser Kartogramme ist, dass die ursprüngliche Form der bertachteten Fläche verloren geht, wie man in diesem Beispiel aus dem Artikel sehen kann.

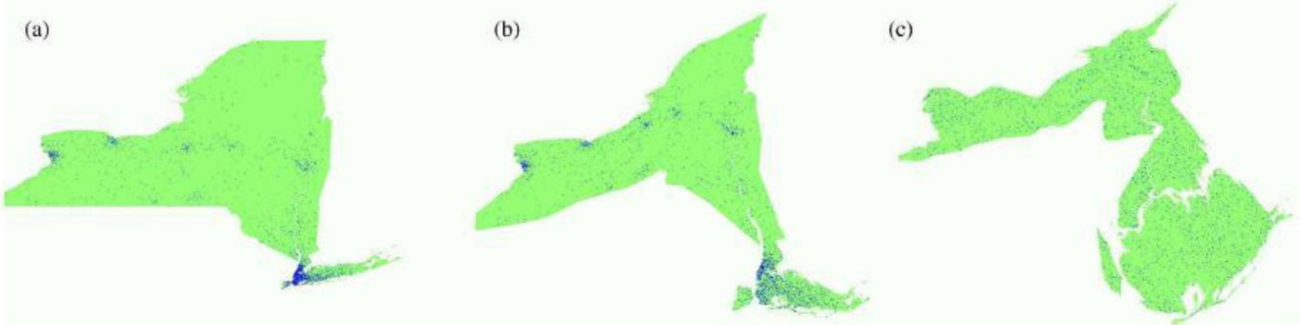


Figure 2: Visualization of lung cancer cases among males in the state of New York 1993–1997. Each dot represents ten cases, randomly placed within the zip-code area of occurrence. (a) The original map. (b) A cartogram using a coarse-grained population density with $\sigma = 0.3°$. (c) A cartogram using a much finer-grained population density ($\sigma = 0.04°$). (Data from the New York State Department of Health.)

Es beschreibt die Lungenkrebserkrankungen der männlichen Bevölkerung in New York von 1993 bis 1997. Die linke Abb. (a) zeigt die originale Proportion von New York. Nach

Dynamic Cartography

- Amount of cancer patients spread equally to squares in each region (e.g. staats)
- Diffusion from places with high density to low
- Diffusion continues until the density is equal distributed
- Regions with higher density grow, others shrink

  → **Cartogram**

Dynamic Cartography

Neumann model

1,5	1,5	2	1,5
1,5	2	2	1,5
1,5	1,5	2	2
1,5	1,5	2	1,5

5	0,9	0,9	0,9
0,9	0,9	0,9	0,9
5	0,9	0,9	0,9
5	0,9	0,9	0,9

Density depending model

1,5	1,5	2	1,5
1,5	2	2	1,5
1,5	1,5	2	2
1,5	1,5	2	1,5

2,5	1,2	1,2	2,5
1,2	1,2	1,2	1,2
2,5	1,2	1,2	1,2
2,5	2,5	1,2	2,5

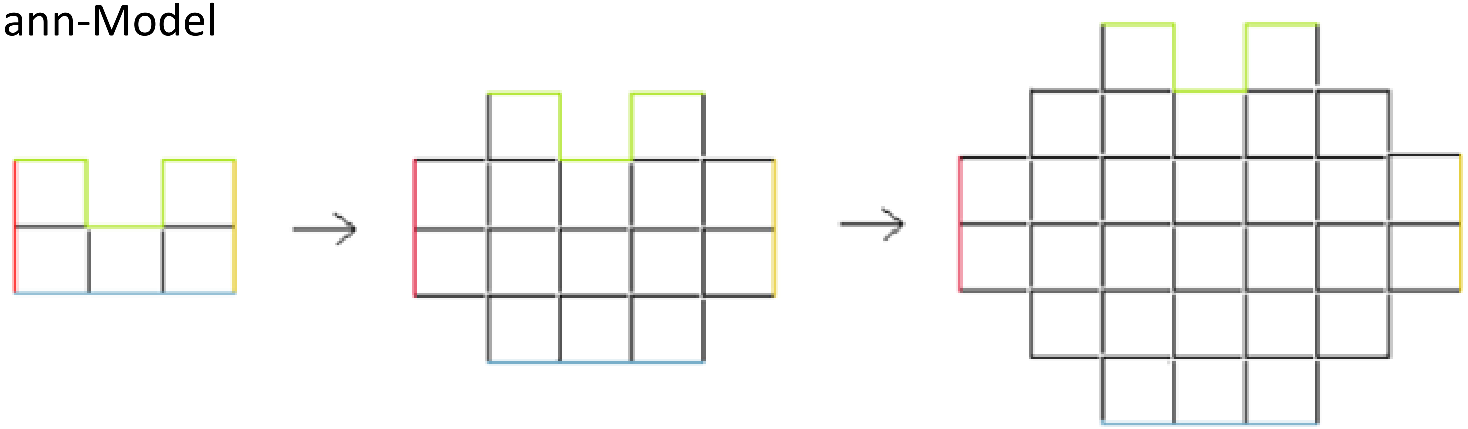
Moore model

1,5	1,5	2	1,5
1,5	2	2	1,5
1,5	1,5	2	2
1,5	1,5	2	1,5

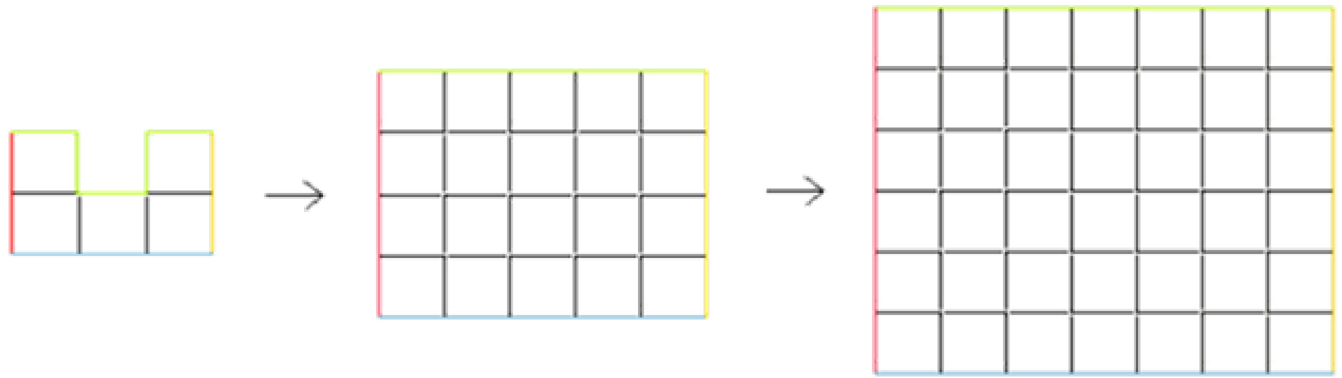
0,8	0,8	0,8	0,8
0,8	0,8	0,8	0,8
0,8	0,8	0,8	0,8
5	0,8	0,8	0,8

Dynamic Cartography - Boundaries

Neumann-Model



Moore-Model



Dynamic Cartography - Results

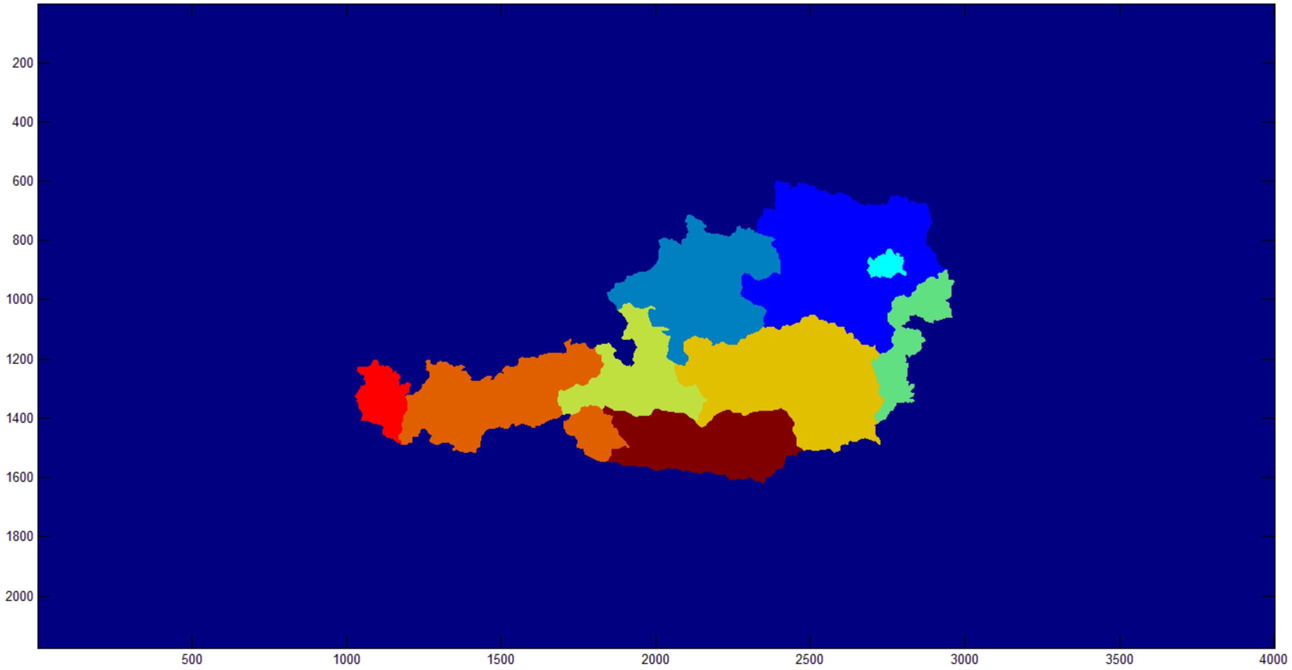
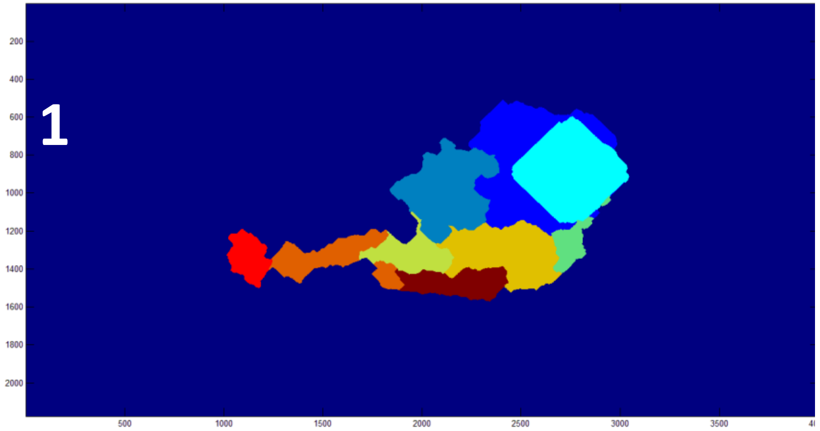
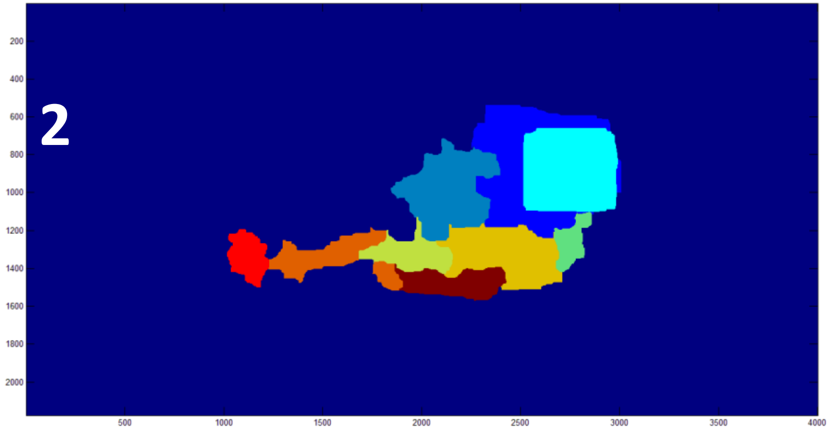
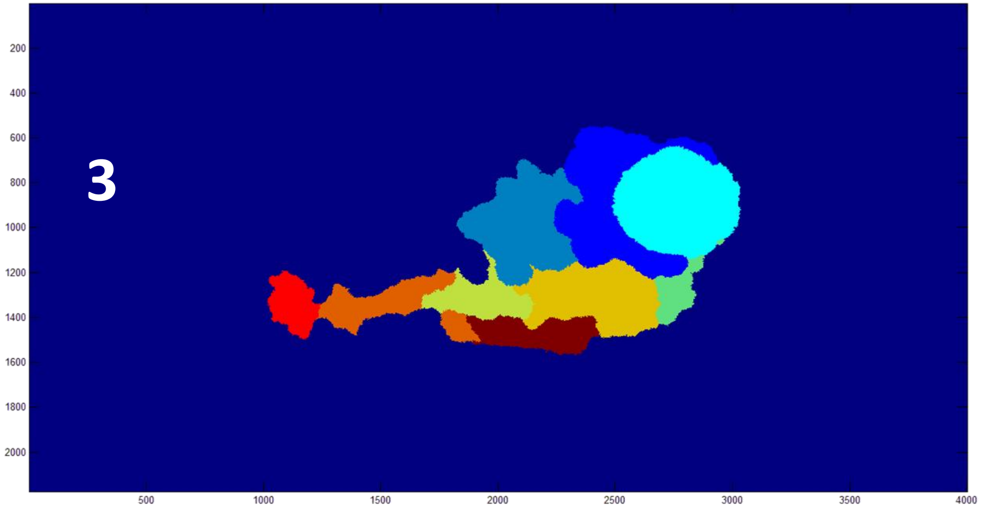


Abb 5.1. originale Österreichkarte

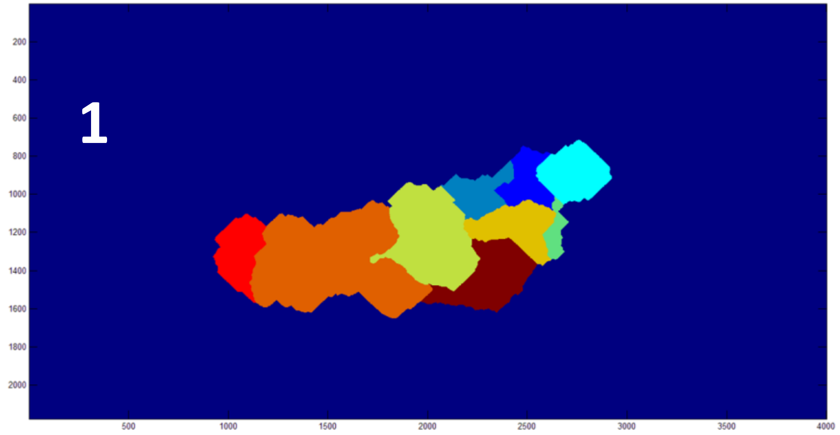
Dynamic Cartography - Population



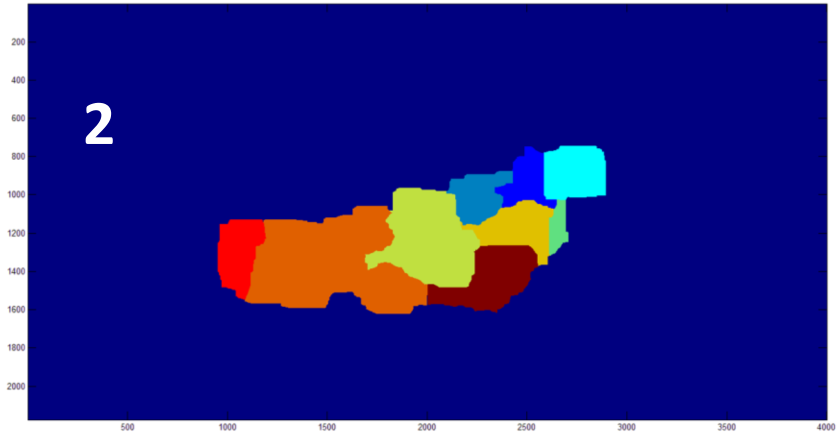




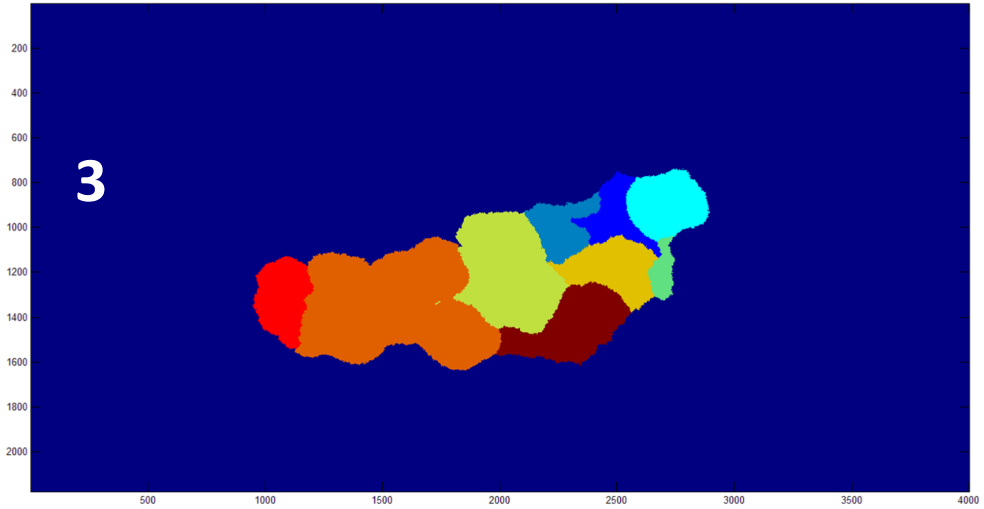
Dynamic Cartography - Tourism



Tourismus mit Neumann-Nachbarschaft

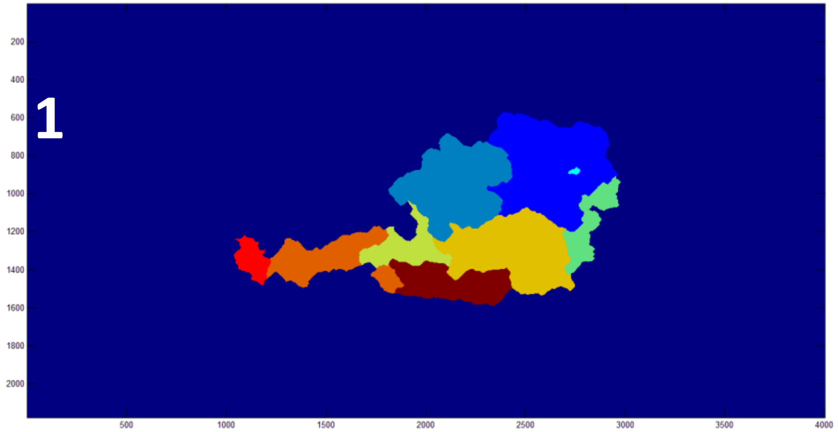


Tourismus mit Moore-Nachbarschaft

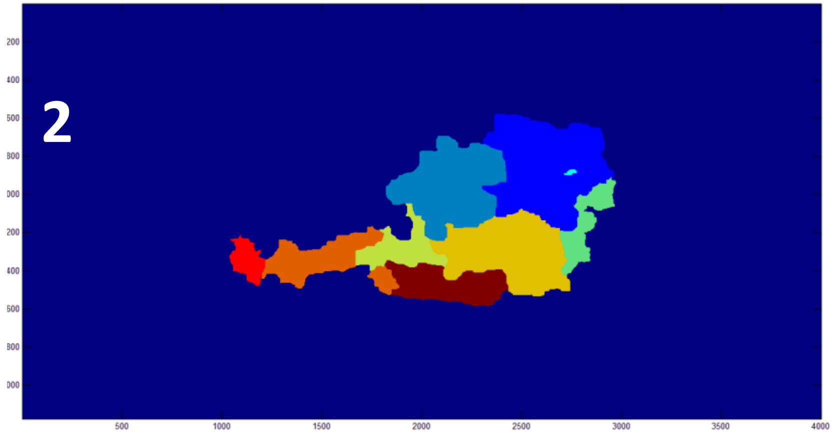


Tourismus mit "Wahrscheinlichkeitsschalter"

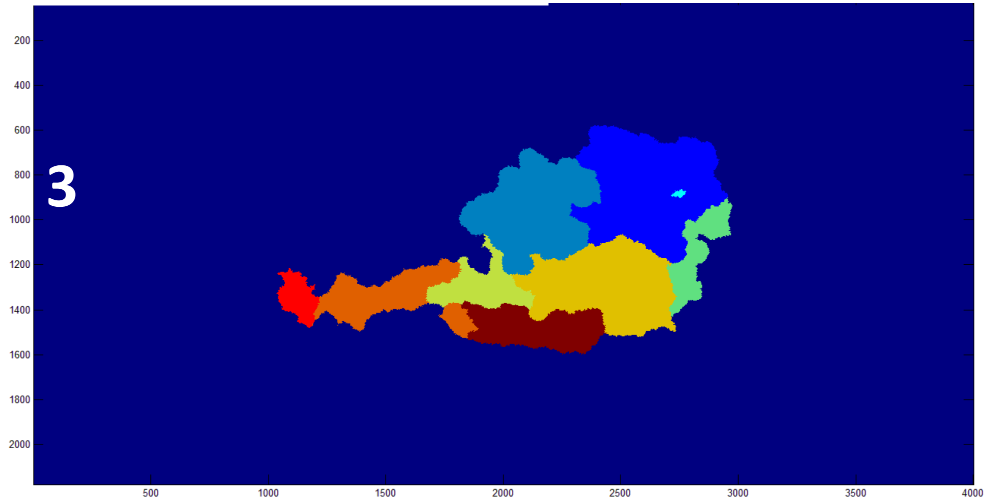
Dynamic Cartography – Hunting game



Haarwildjagd mit Neumann-Nachbarschaft

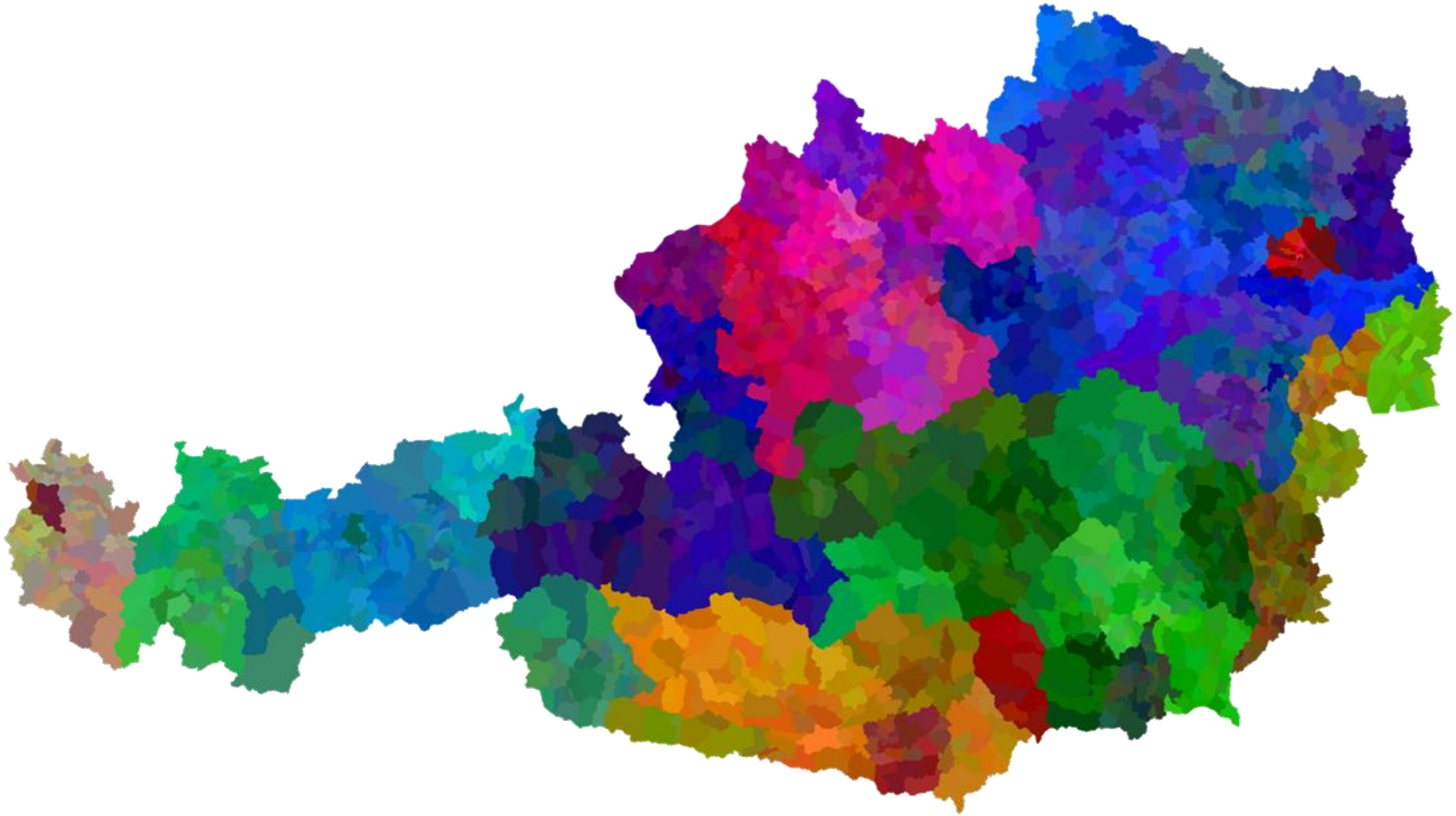


Haarwildjagd mit Moore-Nachbarschaft



Haarwildjagd mit "Wahrscheinlichkeitsschalter"

Cartogram – Population (Municipality Level)



Cartogram – Population (Municipality Level)

Example

LATTICE GAS CELLULAR AUTOMATA (LGCA)

Lattice Gas Cellular Automata

- Lattice Gas Cellular Automata (LGCA)
- Extension of the CA concept
- Intention: Simulate fluids and gases
- Invented by Hardy, Pomeau and de Pazzis (HPP automaton on square lattice), 1973
- Improved by Frisch, Hasslacher and Pomeau (FHP automaton on hexagonal grid), 1986

Lattice Gas Cellular Automata

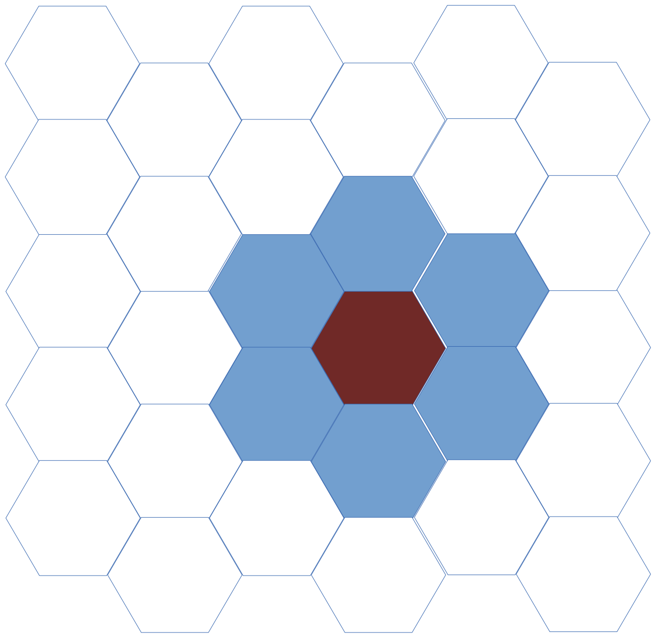
- **Ideas**
  - Cells do not have states but instead can contain particles
  - A particle can only proceed to a cell in the neighborhood
  - Instead of state updates, particles move to other cells
  - Particles represent the fluid or the gas

Lattice Gas Cellular Automata

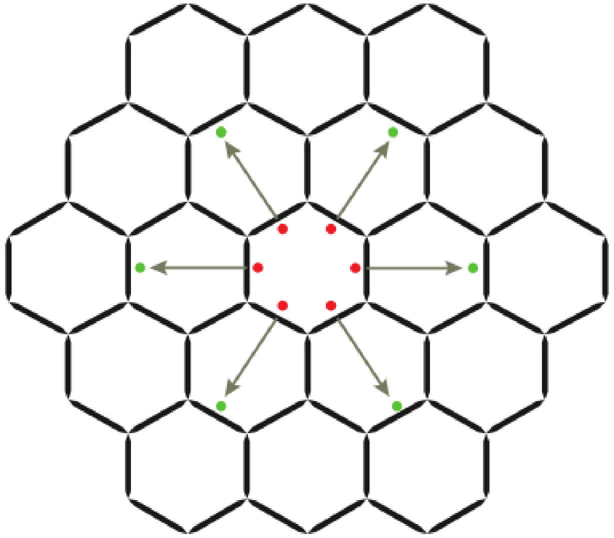
- **HPP**
  - square grid, Von-Neumann neighborhood, max. 4 particles per cell so that max. 1 particle goes to each neighbor
  - several issues when it comes to real interpretations (comparison with real fluids, validation)

- **FHP**
  - hexagonal grid
  - neighborhood = surrounding cells
  - max. 6 particles per cell, each going into a different direction → consistent definition
  - Corresponds to the Navier-Stokes-Equations → valid representation of fluid dynamics

FHP Model



neighbourhood



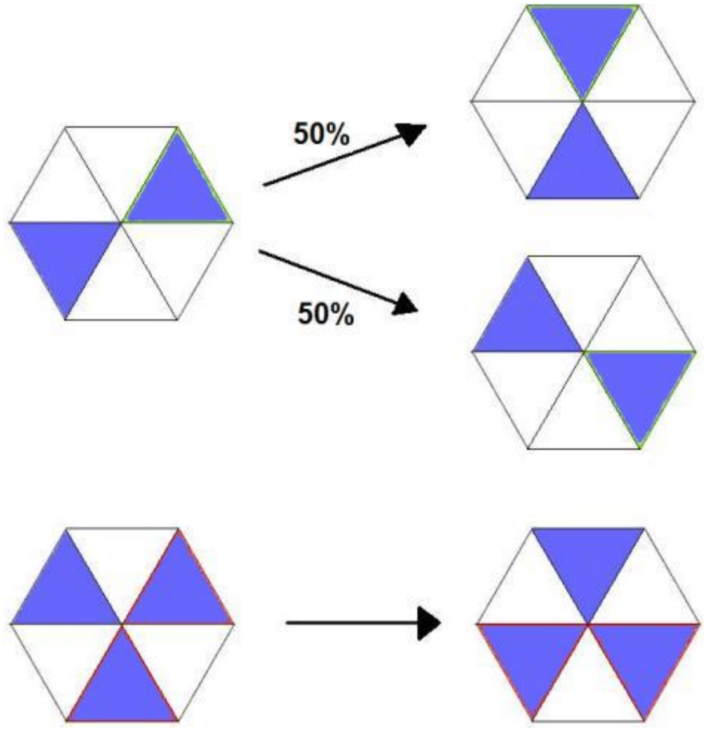
particles and directions

FHP Model

- Particle movements consist of two phases
  - Rotation of cells for special configurations
  - Movements of particles into their direction
- Developed by Wolf-Gladrow (2000)
- Different variations (FHP-I, FHP-II, FHP-III)

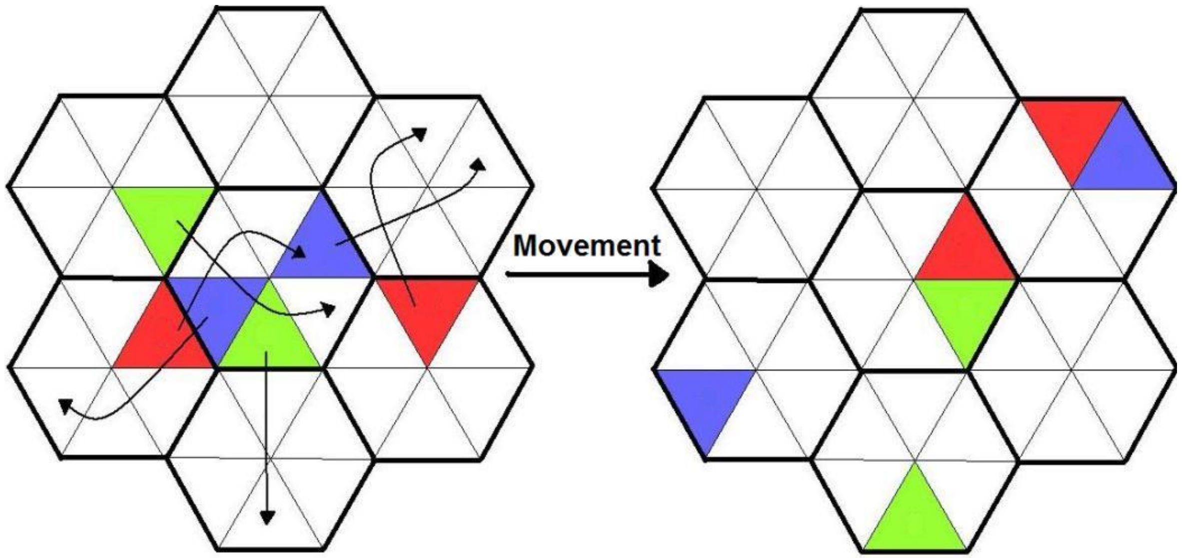
FHP Model

- Rotations
  - In the most simple case of FHP-I only for two situations
  - Provide a randomness



FHP Model

- Movements into designated directions

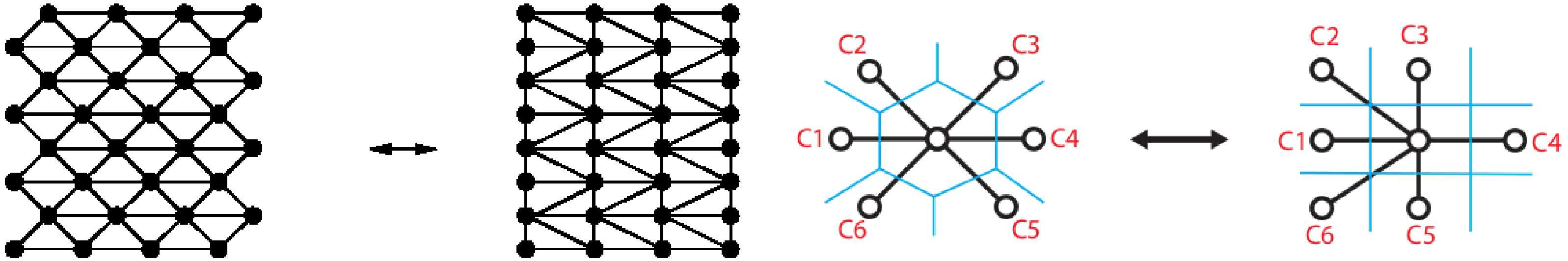


LGCAs

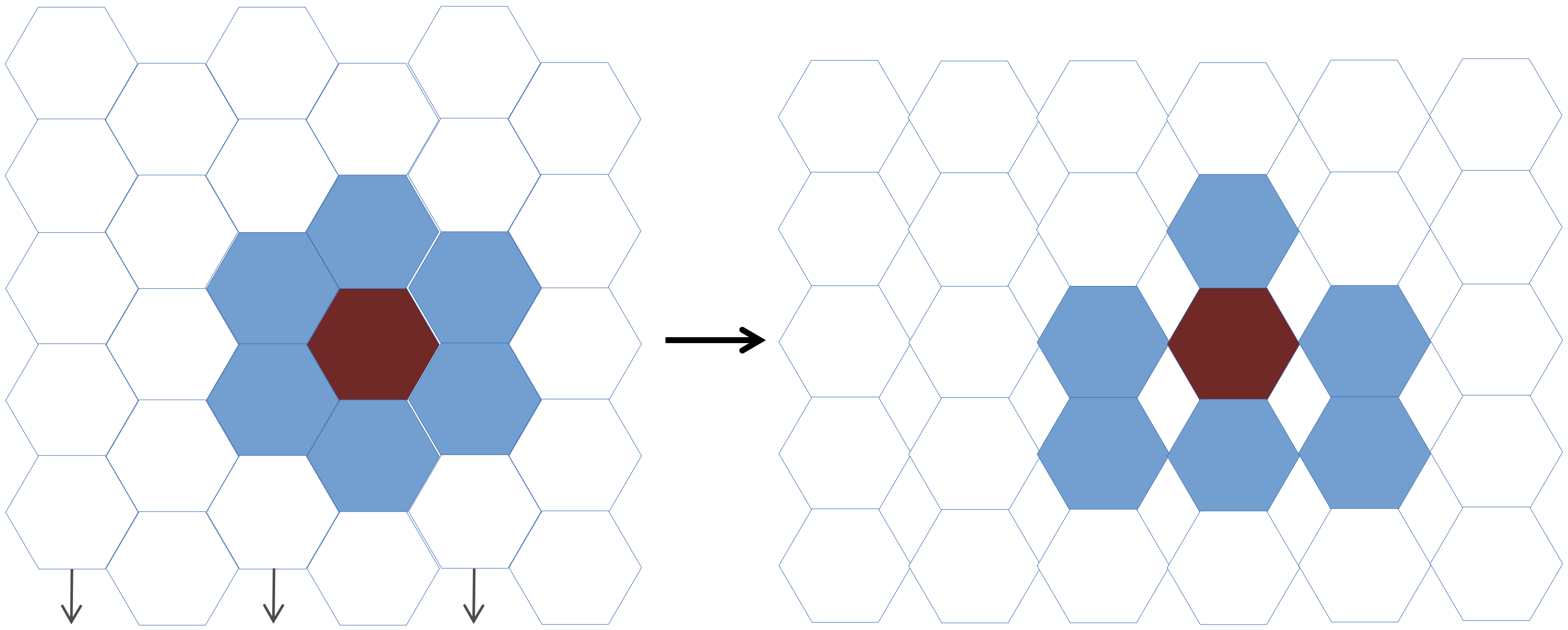
- **Simulations & Visualizations**
- **HPP**
- http://en.wikipedia.org/wiki/File:Gas_velocity.gif
- **FHP**
- http://www.youtube.com/watch?v=HluQpDFOceg
- http://www.youtube.com/watch?v=00W6H7BGZ94

Remark: Implementation of a hexagonal grid

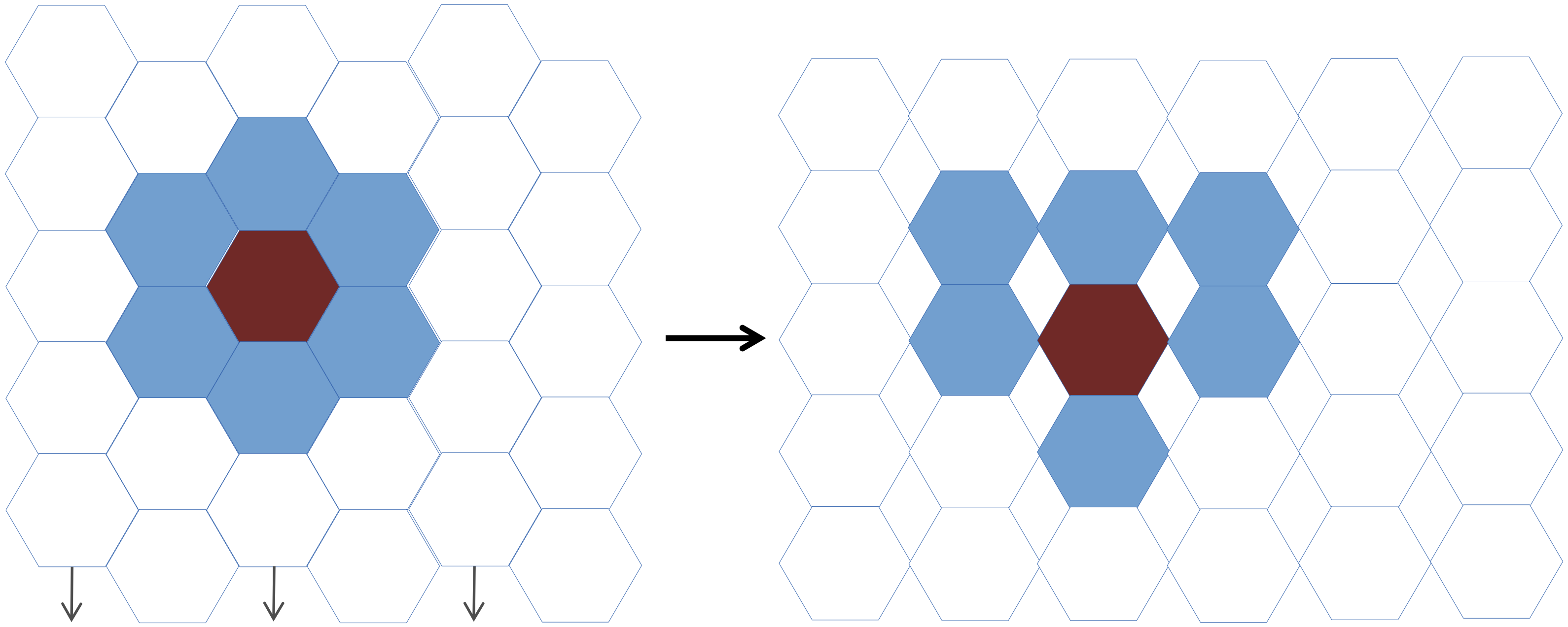
- **Implementation**
  - hexagonally arranged grid → assign to a square lattice
  - conditional neighborhoods



Remark: Implementation of a hexagonal grid



Remark: Implementation of a hexagonal grid

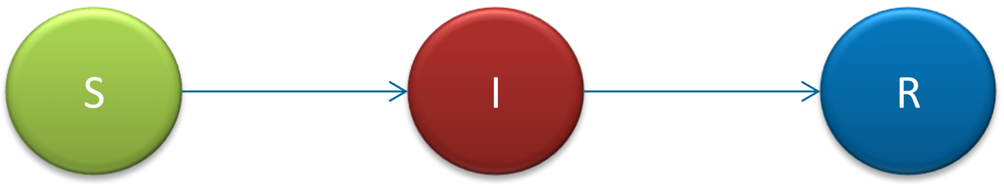


Example

EPIDEMIC SIMULATION WITH CA AND LGCA

SIR concept

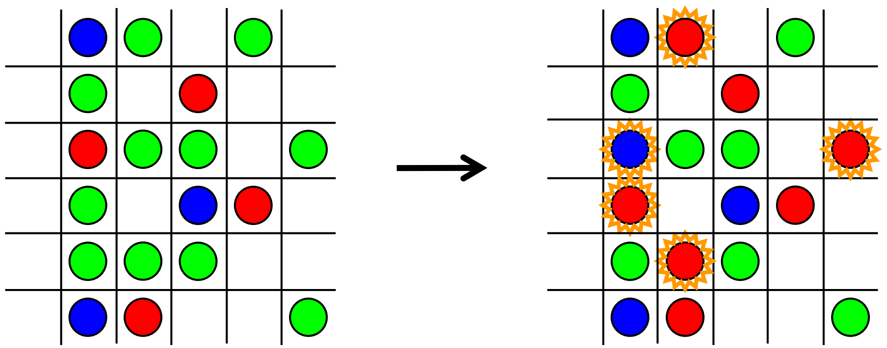
- Simulate the spread of an epidemics
- Susceptible (S) people become infected by infectous (I) and become resistant/recovered (R) after some time.
- Resistant persons cannot be infected again.



CA Implementation

CA Implementation of SIR epidemics:

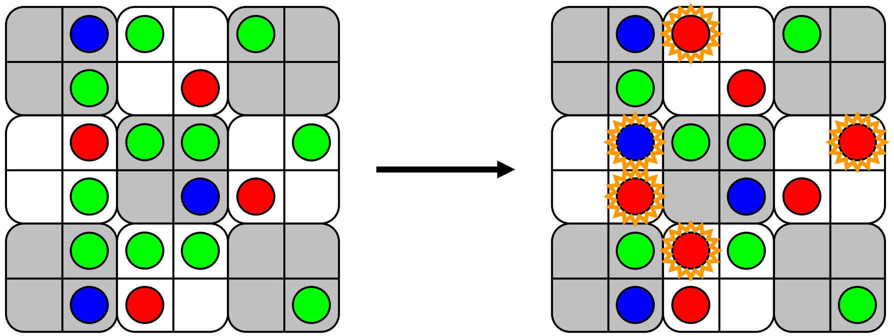
- Every cell in a rectangular (hexagonal..) lattice represents a person/group of persons/household/…
- Infecious cells recover after some time (with some probability).
- Infectious cells may spread the disease to their neighbours (e.g. Moore neighbourhood)



LGCA Implementation

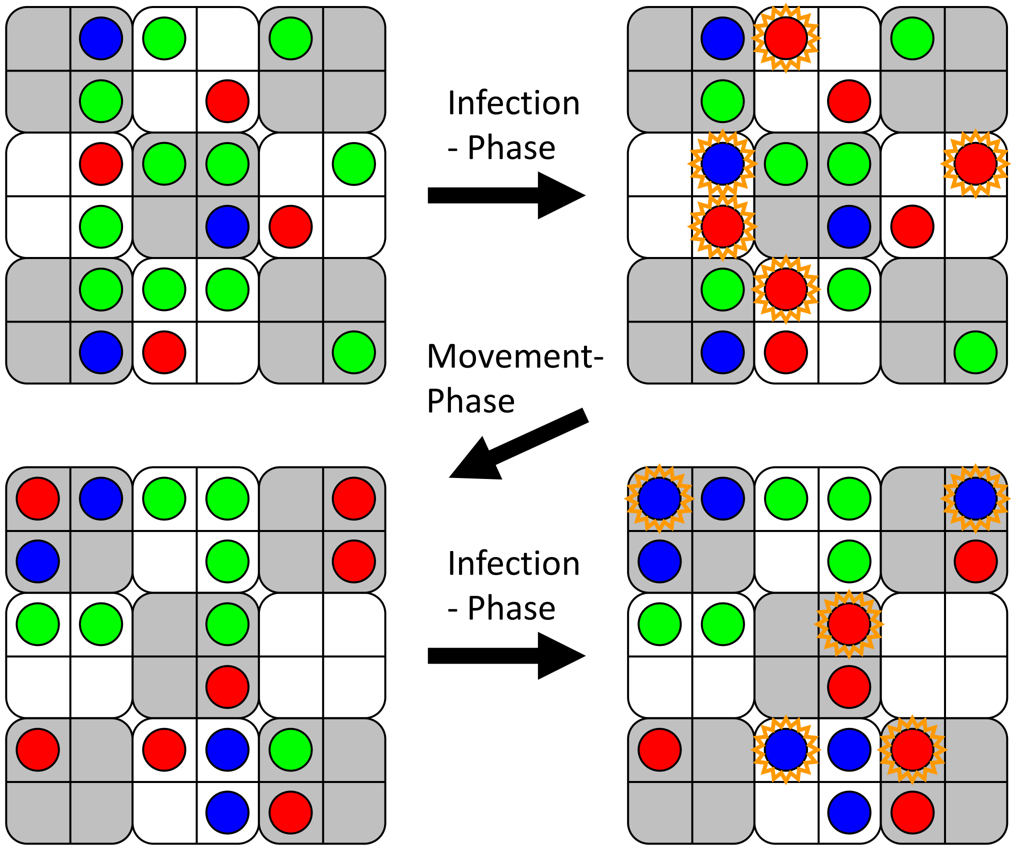
LGCA Implementation of SIR epidemics:

- Every cell in a rectangular (hexagonal..) lattice contains a number of persons (e.g. 4)
- Infecious persons recover after some time (with some probability).
- Infectious persons may spread the disease to all other persons in the cell

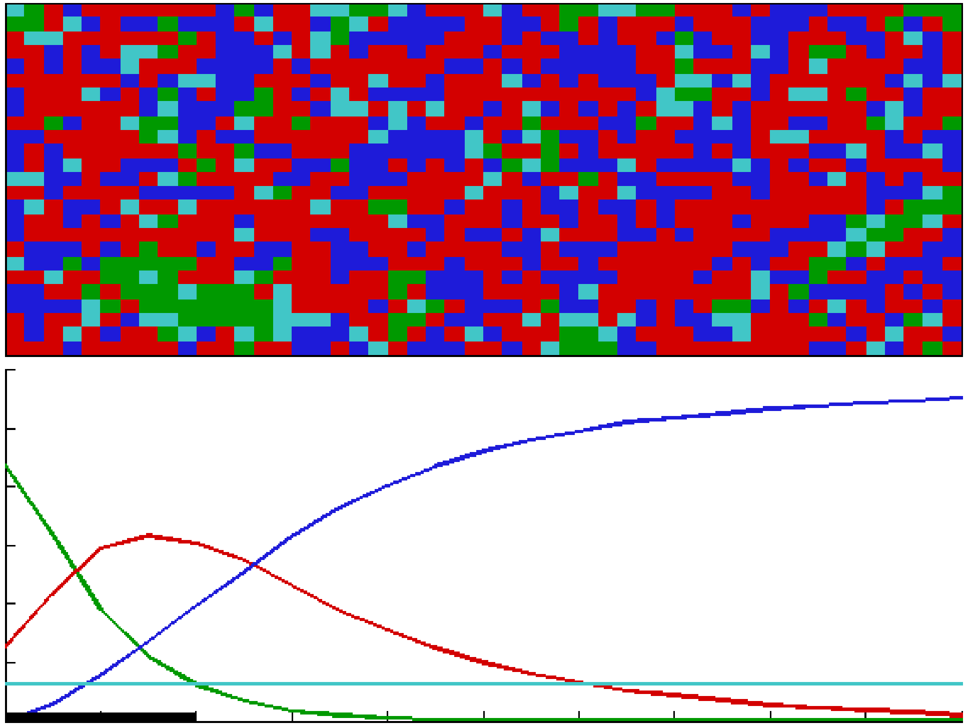


LGCA Implementation

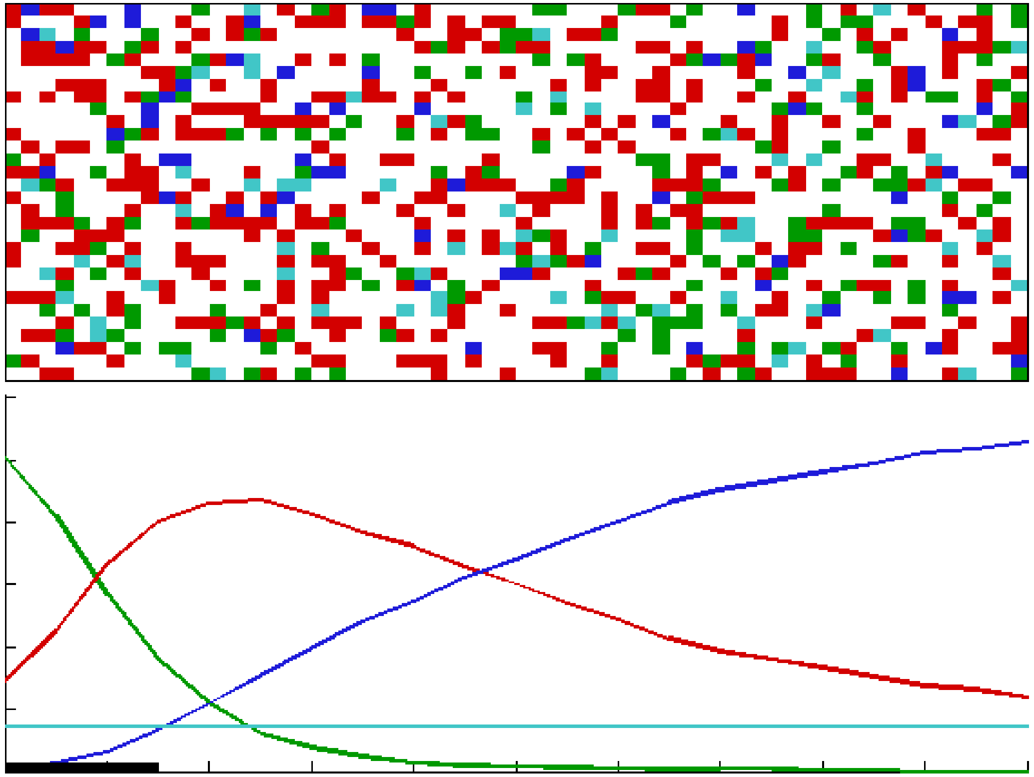
LGCA Implementation of SIR epidemics:



Epidemic simulation with CA



Epidemic simulation with HPP-LGCA



HISTORY OF
CELLULAR AUTOMATA

History of Cellular Automata

- 1925: Ising Modell
  - ferromagnetism, discrete model
- 1950: Von Neumann, Ulam
  - term “cellular automaton”
  - self reproductive, Von-Neumanns theory on logic automata

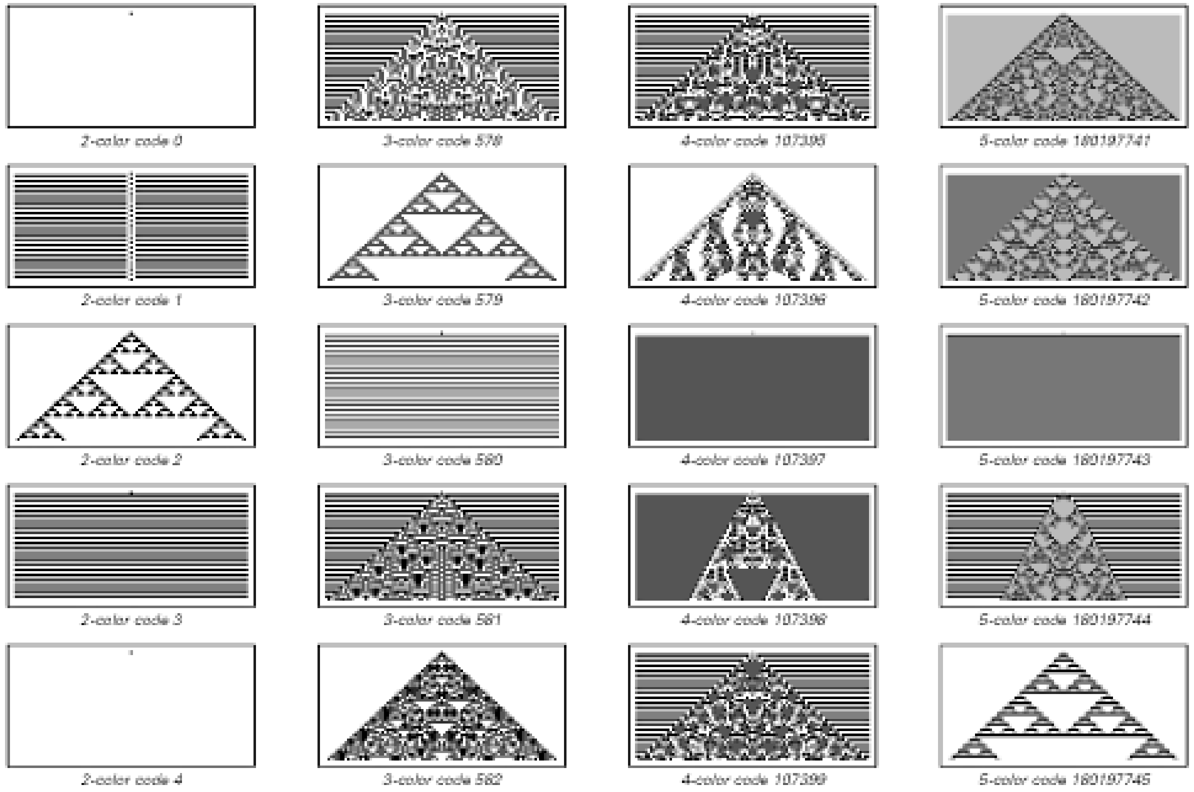
History of Cellular Automata

- 1950-1970: Zuse, et.al.
  - parallel algorithms
  - discrete processes (e.g. PDEs)
- 1970s: Hardy, Pomeau, de Pazzis
  - Lattice Gas Cellular Automata
- 1979: Conway's Game of Life

History of Cellular Automata

- 1980+: different applications
- 2002: Wolfram
  - complete classifications of 1-dimensional cellular automata

Stephen Wolfram, „A new Kind of Science“

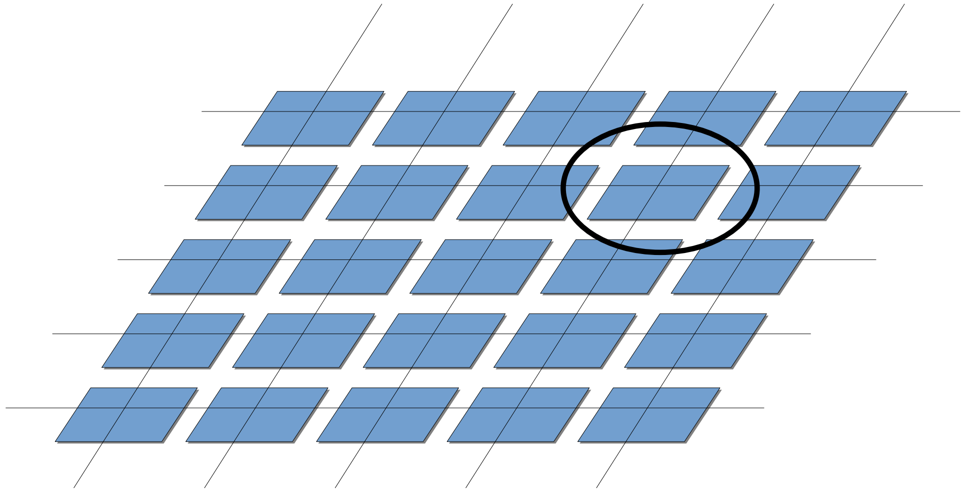


5. Conclusions

CONCLUSIONS

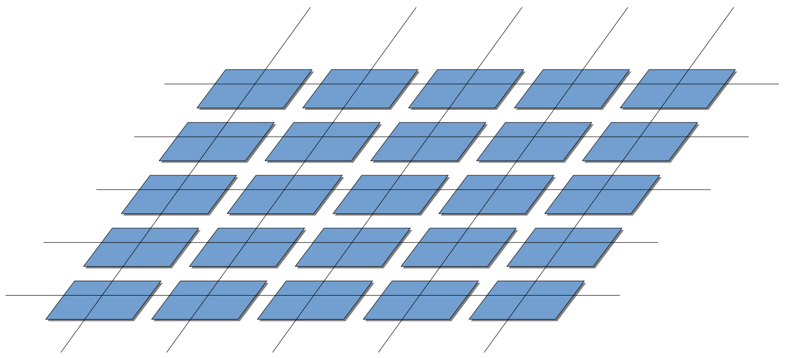
Characteristics of CA

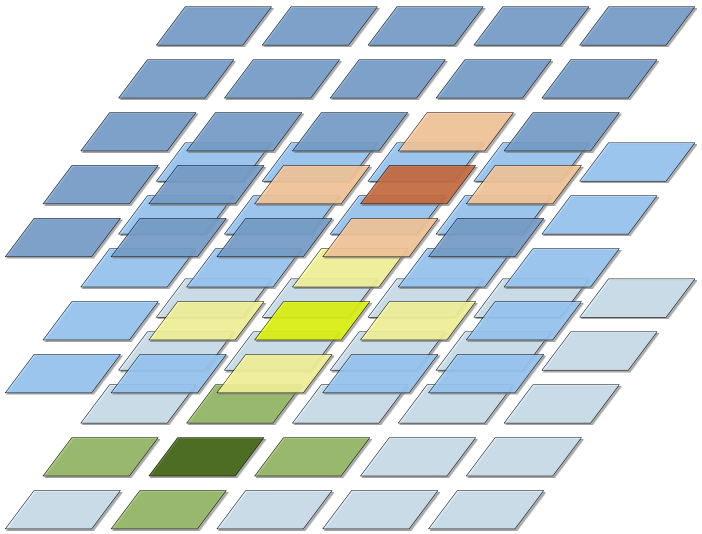
- Behaviour of the whole described by behaviour of the individual cells
  ⇒ **microscopic approach**



Characteristics of CA

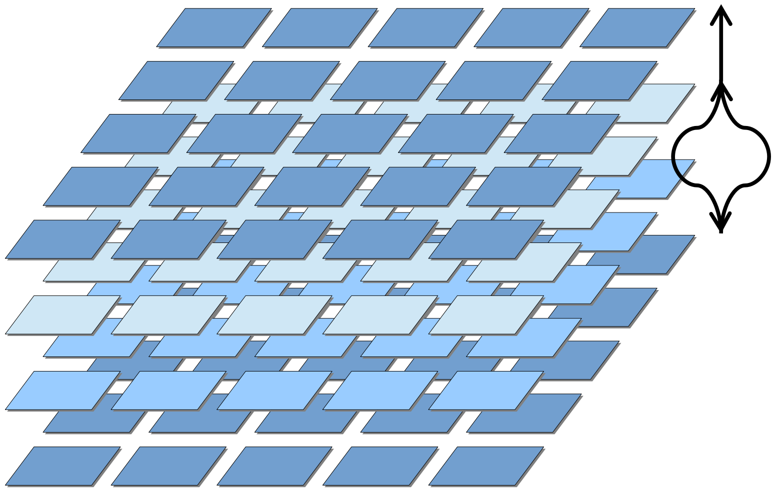
- Regular lattice, same kind of neighbourhoods





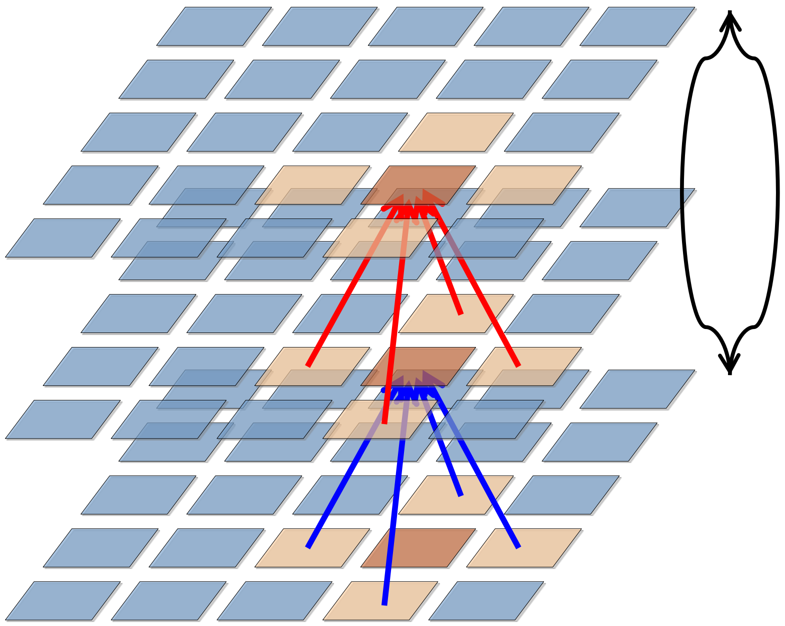
Characteristics of CA

- Discrete time, equidistant time steps



Characteristics of CA

- Spatial representation, locality



Characteristics of CA

- Spatial representation, locality