Cellular automata (CA)

- it is a relatively simple tool for modelling complex systems
  - CA often help to understand their behaviour, especially the self-organization property when global order appears spontaneously in systems which are completely defined only with local interactions (examples are flocks of birds, self-replicating life crystalization etc.)
  - but CA can be used also for modelling in fields of physics where other approaches (such as PDEs) are successfully used, like fluid dynamics, statistical physics etc., especially in systems with more phases or complicated boundary conditions which are changing all the time (deposition etc.)
  - the main advantage of CA is that using simple local rules for interaction of cells (or particles) we can describe many different types of systems (open or isolated, conservative or disipative etc.)

  note: open and disipative systems are necessary for self-organization

History of cellular automata (very brief)

- CA were reinvented several times in various fields under different names, but generally as the beginning of CA, it is considered the work of Stanislaw Ulam and John von Neumann in 1940s at Los Alamos NL who proposed a fully discrete dynamical system of cells having internal states described by a finite number of information bits and interacting with nearest neighbours only

  to simulate self-replicating systems

- CA get wide attention in 1970s
  when John Conway proposed game of life CA,
  even with very simple rules it has unexpectedly
  rich behaviour and it also has the property
  of universal computation

- in 1980s - Stephen Wolfram studied a family of
  simple 1D CA rules (now called Wolfram rules)
  and he noticed that CA is a discrete dynamical
  system which exhibits many of the behaviours
  encountered in continuous systems
  => important for modelling
  - later it was also realised that a <u>lattice gas model</u>
    developed by Hardy, Pomeau and de Pazzi's in 1970s
    (now sometimes called HPP rule)
    is actually CA, it is used as a very simple model
    in statistical physics
    => important step in development of the theory of CA
- in 1980s, the first specialized hardware was developed
  for efficient <u>parallel computations</u> with CA

<u>Basic characteristics of CA</u>

- describes systems <u>discrete in space</u> (lattices) <u>and time</u>
- all <u>states</u> of the system are described by <u>Boolean variables</u>
  => exact dynamics, no round-off errors
- but only <u>finite set of values</u> for any physical quantity
- <u>simple rules</u> for evolution describing local
  interactions with neighbouring cells
  but often <u>nontrivial dynamics</u>

Types of CA

- deterministic or stochastic (probabilistic)
  (evolution depends not only on the states of cells at the time t, but also on some random variable(s))
- synchronous (all cells updated at once)
  or asynchronous (cells updated sequentially or randomly)
- binary (1 Boolean variable) or more states (more Boolean var.)
- reversible or irreversible (usually)

Definition of deterministic synchronous cellular automaton

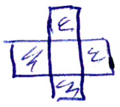
- in general, a CA is given by
  1) a regular lattice of cells (sites) in a d-dimensional space
  2) a set $\Phi(\vec{r},t) = \{\phi_1(\vec{r},t), \dots, \phi_m(\vec{r},t)\}$
     of Boolean variables attached to each site $\vec{r}$ of the lattice giving the local state of each cell at times $t = 0, 1, 2, \dots$
  3) a set of rules $R = \{R_1, R_2, \dots, R_m\}$ specifying the time evolution of the states $\Phi(\vec{r},t)$ as
     $$\phi_j(\vec{r},t+1) = R_j\left(\Phi(\vec{r},t), \Phi(\vec{r}+\vec{\delta}_1,t), \dots, \Phi(\vec{r}+\vec{\delta}_n,t)\right)$$
     where $\vec{r}+\vec{\delta}_k$, $k = 1, \dots, n$ denote the cells from a given neighbourhood of the cell at $\vec{r}$.

Notes: a) the rule R is identical for all sites (homogeneous) independent of $\vec{r}$ (it depends explicitly only on Φ) and of time t,
however, spatial and temporal inhomogeneities can be introduced by some $\phi_j(\vec{r},t)$ systematically equal to 1 in some locations (e.g. boundaries)

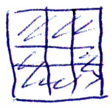
to mark cells for which different rules apply,
or $\phi_j(\vec{r},t)$ can change with ~~time~~ resulting
e.g. in different rules for odd or even times

b) this general definition includes also dynamics
where a new state $\Phi(\vec{r},t+1)$ is given by
the states at times $t, t-1, \dots, t-k$
for we can introduce variables ϕ_j which keep
necessary information from previous times

c) although the neighbourhood of a cell can be
in principle arbitrary (but the same for all cells),
for simplicity and efficiency one usually uses
CAs with von Neumann neighbourhood
(the nearest cells)

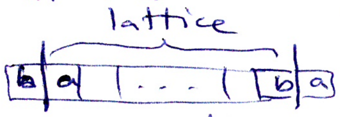


or with Moore neighbourhood
(the nearest and the second nearest cells)

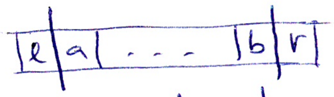


Boundary conditions

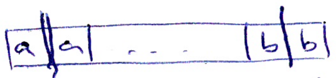
- in simulations, we have to deal only with finite lattices and thus with boundaries
- using additional variables which carry information whether a cell belongs to a specific boundary, we can have different rules at boundaries
- usually, various types of extensions are used at the boundary:



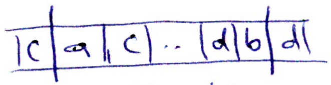
periodic



fixed value



zero gradient
(adiabatic)



reflection

Number of cellular automata

- let us consider a general CA with

  q possible states in each cell and

  K cells in one neighbourhood (including the cell for which the neighbourhood is defined)

  (e.g. for von Neumann neighb. $K = 2d+1$

  and for Moore neighb. $K = 3^d$

  where d is the dimension of lattice space),

then there is $N_{CA} = q^{q^K}$ of different CA

because q^K is the number of states of the whole neighbourhood

and we can assign to each such a state

q possible „new" states of the cell defining the neighbourhood

- for a binary CA: $q=2$, thus in 1D with $K=3$ we get $N_{CA} = 2^{2^3} = 256$ cellular automata

  (their classification was obtained by S. Wolfram in Reviews of Modern Physics 55, 601 (1983)

  see also rules 1 - 256 in Mathematica's CellularAutomaton function)

but in 2D we get

von Neumann: $K=5 \Rightarrow N_{CA} = 2^{2^5} = 2^{32} \simeq 4\cdot 10^9$

Moore : $K=9 \Rightarrow N_{CA} = 2^{2^9} = 2^{512} \simeq 10^{154}$

we see that the number of CAs grows very fast

thus they are sometimes classified only according to their typical behaviour if the initial state is chosen randomly

$\Rightarrow$ 4 basic classes - see examples in 1D.Cellular.Automata.nb

Examples of cellular automata

1) one-dimensional simple CA to model traffic

- it is given by Wolfram's rule 184 (see 10.Cellular.Automata.nb)
- cells have two possible states
  1 - there's a car at that site
  0 - there's no car, empty cell
- rules simulate motion of ars in one direction with velocity 1 or 0:
  a car can move to the neighbouring right cell if this cell is empty, otherwise it stays

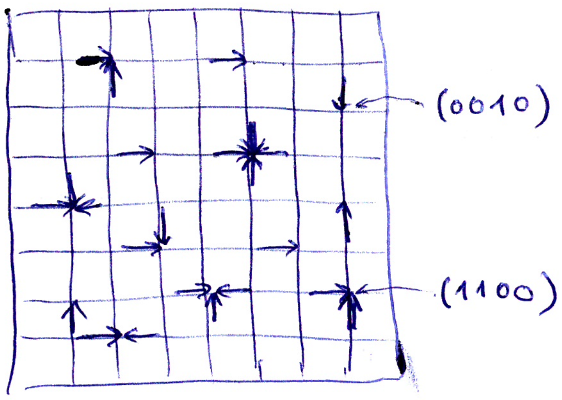
=> „traffic jams" propagate in the opposite direction to cars motion

2) John Conway's Game of Life

- a well-known CA with simple rules but complex behaviour
- cells have again two possible states - alive (1) or dead (0)
- 2 rules: 1) the cells stays alive if it is surrounded by 2 or 3 live cells, otherwise dies
  2) a dead cell becomes alive, if there are exactly 3 live cells in its neighbourhood
- we consider the Moore neighbourhood here (8 surrounding cells) and periodic boundry conditions
- examples can be found at wikipedia or in Game.Of.Life.nb

3) Lattice gas model and the HPP rule

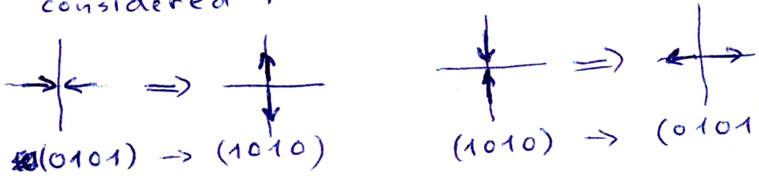
- developed in 1970s by Hardy, Pomeau and de Pazzis but only in 1980s it was recognized as a cellular automaton
- fully discrete dynamics of particles on a lattice (2D, square) with conservation of momentum and particle number and in its basic form it is reversible
- the number of particles at one site is limited to one particle incoming from one direction (this exclusion principle is actually forced by definition of CA because we cannot describe an arbitrary number of particles ~~in~~ on one site ~~within~~ with Boolean variables)



- in the HPP rule, four bits are used to describe incoming particles to a site $\vec{r}$ at the time t

e.g. state $s(\vec{r},t) = (1,011)$ means there are 3 incoming particles from directions 1,3,4
$\uparrow \rightarrow \downarrow \leftarrow$

- the HPP rule is usually split into two steps
  1) collisions: determine new directions of particles
     if there is a collision, only two collisions are considered (in all other configurations use only step 2))



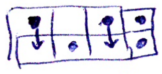
  2) motion: after that move all particles along the lattice in their respective direction

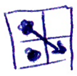
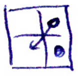
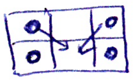
- notice that the exclusion principle will be satisfied if the initial conditions were also chosen to satisfy it and that locally the momentum and particle number are conserved during a collision

- examples of evolution are in Lattice.Gas.nb where as a boundary condition, the reflection of a particle from the wall is used

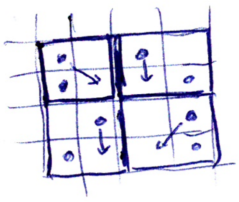
4) Sand pile rule

- a simple CA simulating a sand pile or sand motion due to gravity
- again two possible states (except for boundaries)
  - 1 – there's a sand grain in a cell
    0 – no grain
- rules in the 2D square lattice:

  if there is an empty cell bellow a cell with a grain the grain will fall : 

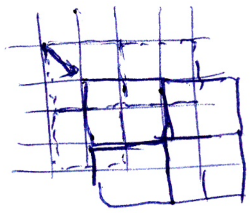
- but a grain can also fall aside  or 
- to avoid conflicts such as 

  we can use the Margolus neighbourhoods which are changing over the time:



← we solve motion of grains in each 2x2 block independently

and then we shift blocks by 1 to the ~~left~~ right and down



if we consider a boundary condition as a wall in all cells arround the lattice there is no trouble in shifting neighbourhoods

- see Sand.Pile.nb for an example of implementation