CITS4403 – April 24, 2013

Modelling Complex Systems

Cellular Automata (Part I)

Pascal Buenzli
pascal.buenzli@uwa.edu.au
Lab G17, CSSE
The University of Western Australia

Overview

- CA: History & Introduction
  - von Neumann, self-replication
  - Conway's Game of Life
  - Modelling physical systems. Some examples
- CA: Formal definition
- Brownian motion, temperature, random walk, diffusion

CA: History & Introduction

John von Neumann
(1903-1957)

- Late 40s: involved in design of first digital computers
- Self-reproducing systems
- Seminal work for CA:

Theory of self-reproducing Automata,
John von Neumann,

http://en.wikipedia.org/wiki/John_von_Neumann

edited and completed by Arthur W. Burks
University of Illinois Press, Urbana and London, 1966

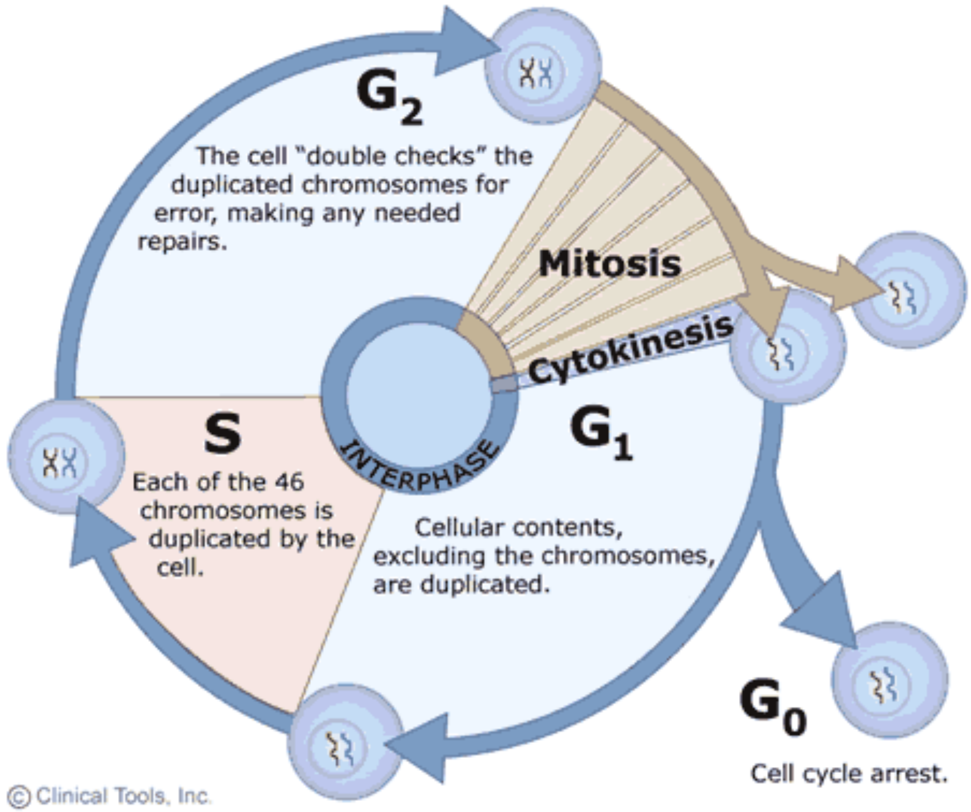
- Idea: Human brain → Machine to solve complex problems

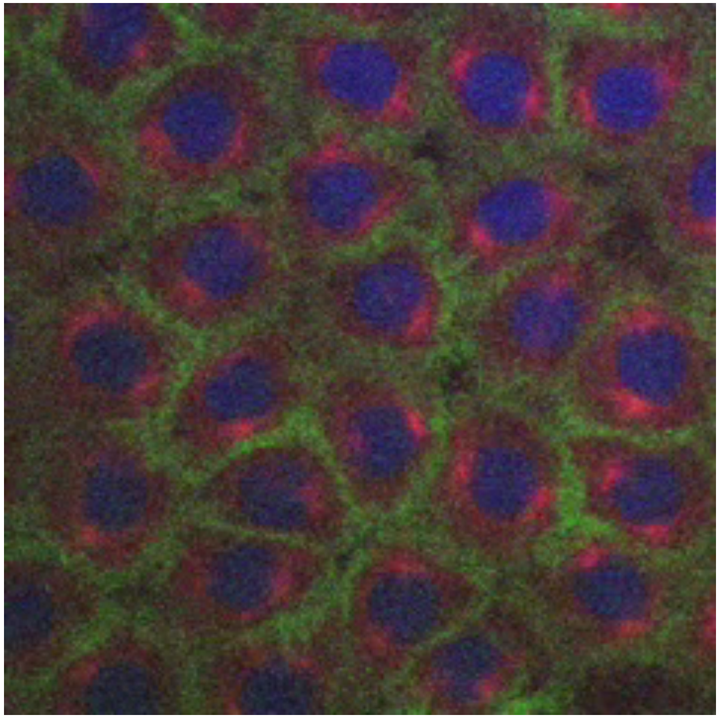
- Such a machine should contain
  - self-control mechanisms
  - self-repair mechanisms
  - no difference between processors and data

- Machine capable of building itself
  - What properties does a system need to be self-replicating?
  - Logical abstraction of self-replicating mechanisms

Self-replicating examples

Cell division





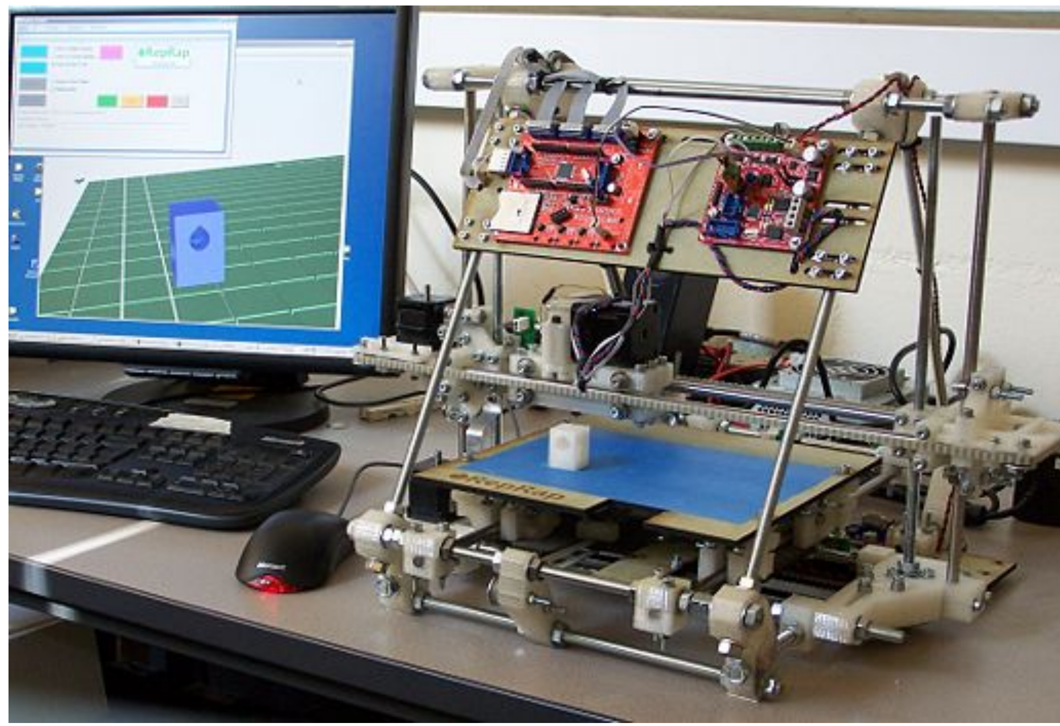
http://bio.research.ucsc.edu/people/sullivan

RepRap: free desktop 3D printer

http://RepRap.org: `Self-replicating' 3D printer

***RepRap is about making self-replicating machines**, and making them freely available for the benefit of everyone*.

***RepRap** is a free desktop 3D printer capable of printing plastic objects. Since many parts of RepRap are made from plastic and RepRap can print those parts, RepRap is a self-replicating machine - one that anyone can build given time and materials. It also means that - if you've got a RepRap - you can print lots of useful stuff, and you can print another RepRap for a friend...*



(Back to von Neumann)

- Following suggestions of S. Ulam (1952):
  - Fully discrete universe of identical `cells'
  - Each cell has an internal `state' : e.g. finite number of bits
  - Discrete time step evolution
  - Automata, who only know a simple recipe to compute their new internal state.
  - The evolution rule is the same for all cells and a function of the states of the neighbouring cells.

- Fully discrete dynamics systems: Cellular Automata

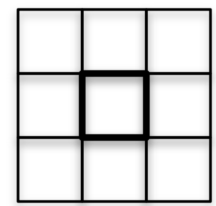
- von Neumann's first self-replicating CA:
  - 2D square lattice
  - 29 states/cell
  - self-replicating structure = several thousands cells!

- Proof of principle
  - usually: a machine builds objects of lesser complexity than itself
  - Self-replicating CA: identical complexity and capabilities

- Property of `Universal computation'
  - there is an initial config. of the CA which leads to the solution of any computer algorithm.

Example: Conway’s `Game of Life’

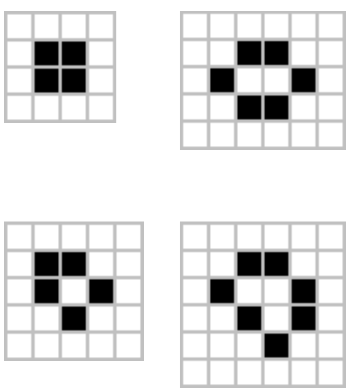
- John Conway (1970)
- Motivation: simplify Neumann’s idea
- Each cell is either alive (state 1) or dead (state 0)
- Lattice 2D, 8 nearest neighbours (Moore)



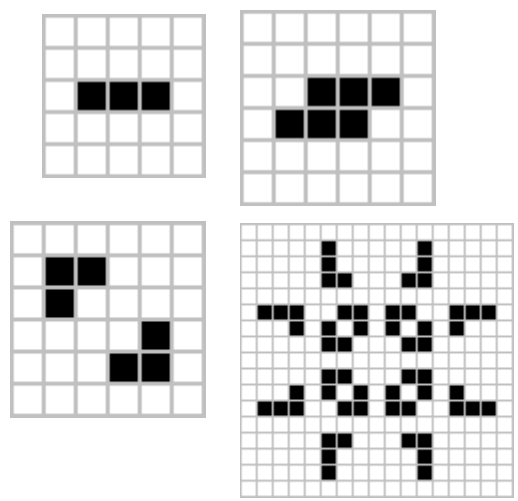
- Evolution rule (B3/S23)
  - A dead cell comes back to life if surrounded by (exactly) 3 live cells
  - A live cell dies if surrounded by less than 2 (isolation) or more than 3 (overcrowdness) live cells.

- Some Properties:
  - Capable of computational universality (Turing complete)
  - Emergence and self-organisation: complex pattern emerge from very simple (microscopic) rules
- Simulation example:
  - Java Applet: http://www.ibiblio.org/lifepatterns/
- Patterns:

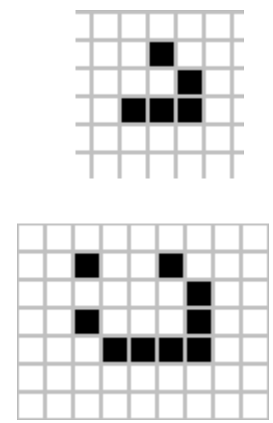
Still lives



Oscillators



`Spaceships/gliders'



http://en.wikipedia.org/wiki/Conway's_Game_of_Life http://en.wikipedia.org/wiki/Conway’s_Game_of_Life

Gosper’s glider gun:

Breeder leaving glider guns in its wake:

- Philosophical implications:

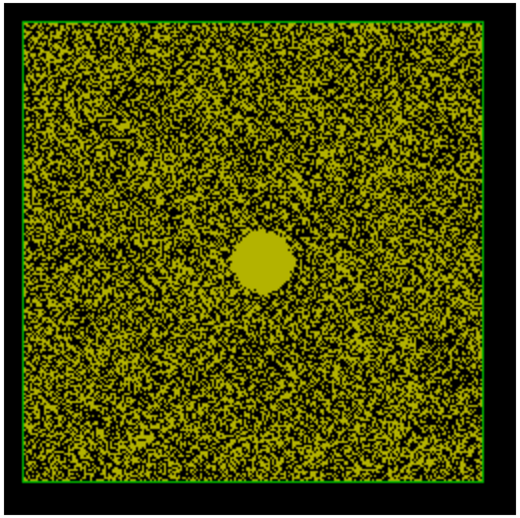
  Chaos, design and organisation can spontaneously emerge from simple deterministic physical laws

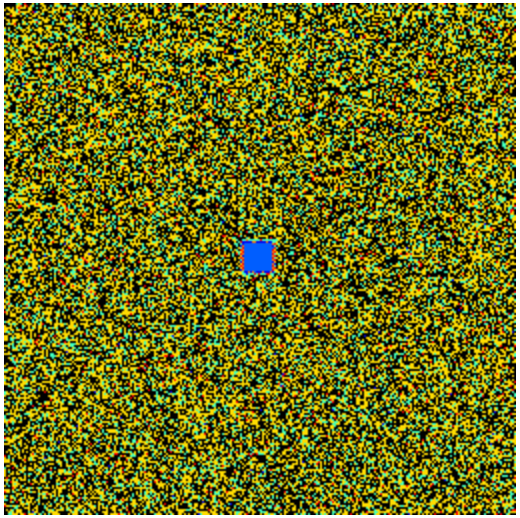
Modelling physical systems with CA

- The physical world could be a very large CA...?
- CA can provide a modeling environment for physical systems
- CA = synthetic model of a system in which physical laws (microscopic interaction) are expressed as simple local rules on a discrete space-time structure

Some examples (overview)

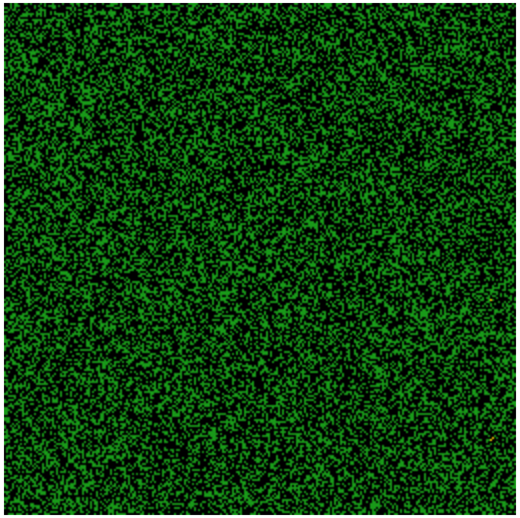
- Lattice gas (HPP, FHP):
  - Hardy, Pomeau, de Pazzis (1970)
  - Frisch, Hasslacher, Pomeau (1986)
- Discrete dynamics of particles moving and colliding
- Conservation of momentum and particle numbers
- FHP shown to be equivalent to Navier-Stokes equation (in some appropriate limits)



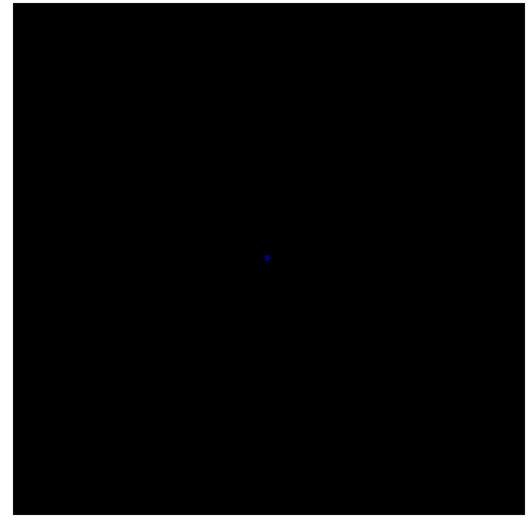


Chopard & Droz, http://theory.physics.unige.ch/cours/outils/

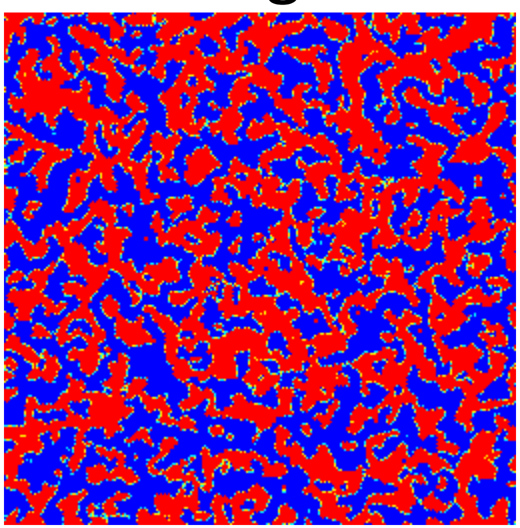
- Forest fires



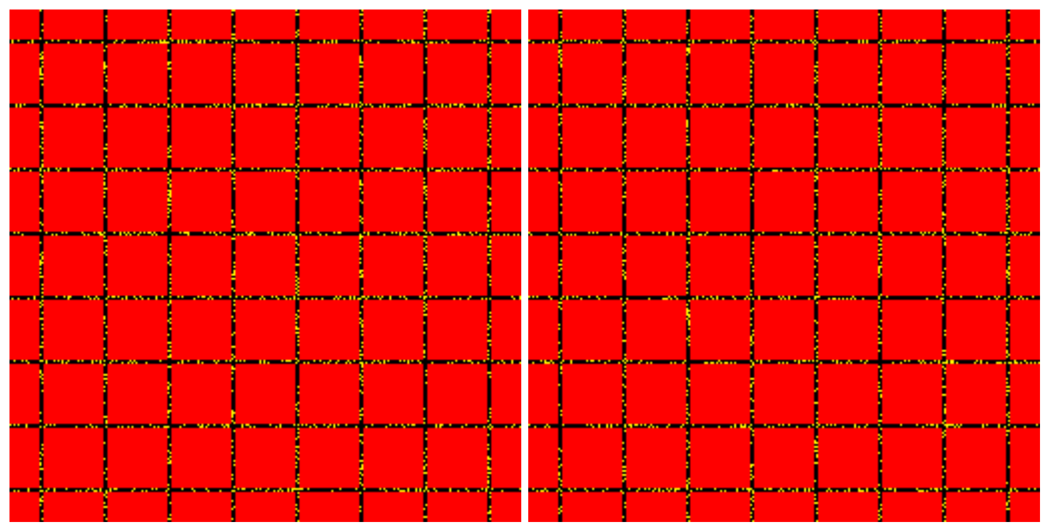
- Snow flakes



- Annealing



- Traffic flow



Chopard & Droz, http://theory.physics.unige.ch/cours/outils/

CA: Formal Definition

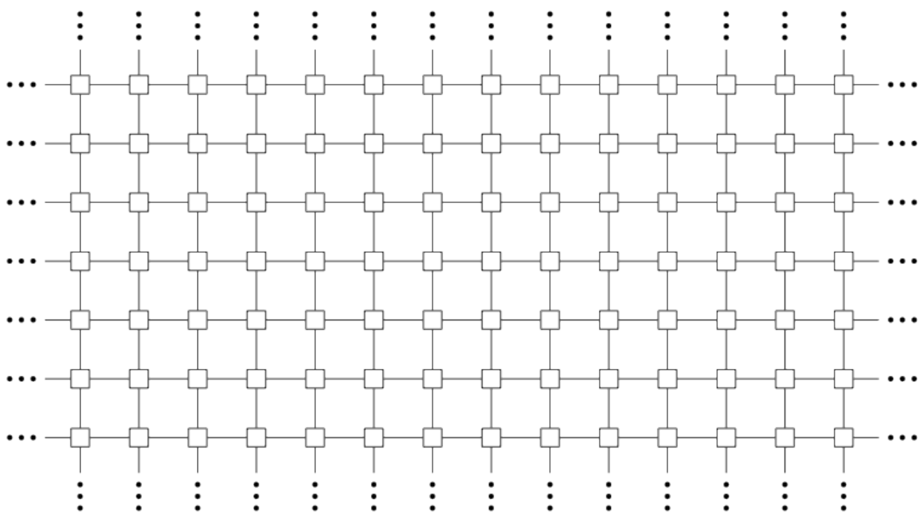
A Cellular Automaton requires in general

1. A regular lattice *L* of ordered cells
2. A finite state space *S* for each cell;
3. A neighbourhood *N* that each cell interacts with
4. An evolution rule *R* which specifies the time evolution of the states

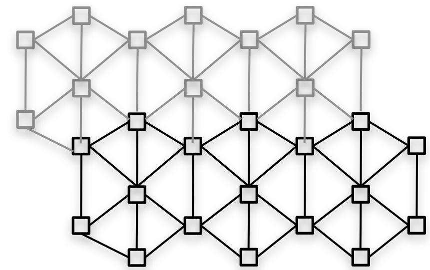
1. A regular lattice L of ordered cells

1D: L = **Z**

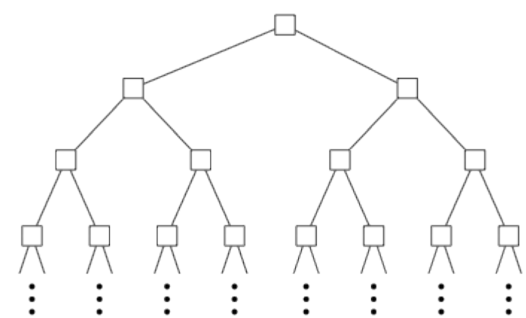
2D, 3D: L = **Z**^{2}, **Z**^{3}



Hexagonal: L = **H**



Other: L = **T**$_2$



2. A finite state space S for each cell

- The state space of the cells is a finite list of possible states:
  $S = \{S_1, S_2, \ldots S_m\}$ $\boxed{}$ in $\{\boxed{0}, \boxed{1}, \boxed{2}, \boxed{3}\}$

- Boolean representation of the state of a cell at position $\boldsymbol{r}$ (discrete) on the lattice at time t = 0,1,2,... (discrete):
  $s(\boldsymbol{r}, t) = S_i = [b_1(\boldsymbol{r}, t), b_2(\boldsymbol{r}, t), \ldots, b_m(\boldsymbol{r}, t)]$, $b_i(\boldsymbol{r}, t)$ in $\{0,1\}$

- Notation: often $\boldsymbol{r}=(i,j)$ is used (in 2D) and the notation:
  $s(\boldsymbol{r}, t) = s_t(\boldsymbol{r}) = s_t(i,j)$; sometimes also $s_{\boldsymbol{r}}(t)$, $s_{i,j}(t)$ (don't be confused...!)

- A (global) configuration or microscopic state of the system is given by the set of each cell's state:
  $\{s_{\boldsymbol{r}}\}_{\boldsymbol{r} \text{ in } L}$

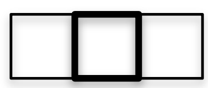
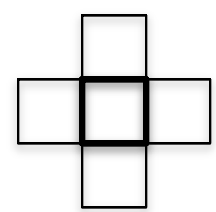
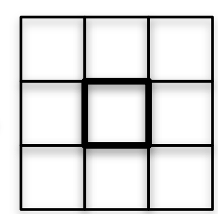
3. A neighbourhood N that each cell interacts with

- A cell interacts only with cells in a certain neighbourhood
- This neighbourhood is specified as a set of relative coordinates:

  $N = \{\boldsymbol{n}_1,\dots,\boldsymbol{n}_q\}$

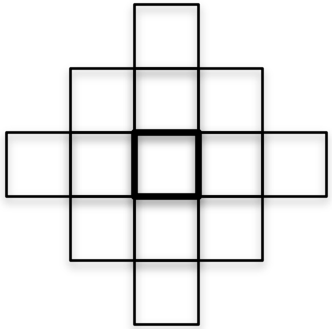
- I.e., the neighbour i of the cell at $\boldsymbol{r}$ is located at $\boldsymbol{r} + \boldsymbol{n}$
- A local configuration at $\boldsymbol{r}$ is given by the set of each cell's state in the neighbourhood of $\boldsymbol{r}$: $\{s_{\boldsymbol{r}+\boldsymbol{n}}\}_{\boldsymbol{n} \text{ in } N}$

Examples:

- $L = \mathbf{Z}$, $N = \{-1, 0, 1\}$: 
- $L = \mathbf{Z}$, $N = \{-10, -3, 15, 32\}$
- $L = \mathbf{Z}^2$, $N = \{(0,1), (-1,0), (0,0), (1,0), (0,-1)\}$ (von Neumann) 
- $L = \mathbf{Z}^2$, $N = \{(-1,-1), (-1,0), (-1,1), (0,-1), (0,0), \dots, (1, 1)\}$ (Moore) 

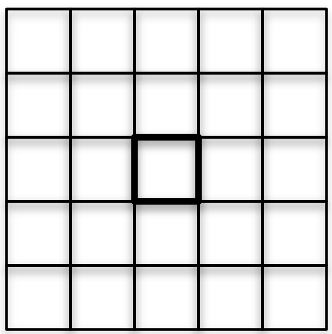
- von Neumann neighboorhoods
  - range R, in d dimensions
  - $N_R = \{ \boldsymbol{r}=(i_1,\ldots,i_d): \Sigma_j |i_j| \leq R \}$

N_2



- Moore neighbourhoods
  - range R, in d dimensions
  - $M_R = \{ \boldsymbol{r}=(i_1,\ldots,i_d): \max_j |i_j| \leq R \}$

M_2



4. An evolution rule *R* which specifies the time evolution of the states

- The evolution rule assigns a new state to the cell depending on the states of its neighbouring cells:

  R: maps $S^{|N|} = S \times S \times \ldots \times S \longrightarrow S$ during a time step:

  $$s(\mathbf{r}, t+1) = R(s(\mathbf{r}, t), s(\mathbf{r}+\mathbf{n}_1, t), s(\mathbf{r}+\mathbf{n}_2, t), \ldots, s(\mathbf{r}+\mathbf{n}_q, t))$$

- The evolution rule is often given as a table;
- The local neighbourhood determines the new state:

N:	-1	0	1	R
	0	0	0	0
	0	0	1	1
	0	1	0	1
	0	1	1	1
	1	0	0	0
	1	0	1	1
	1	1	0	1
	1	1	1	0

Rule 110, Wolfram (1984)

- The rule is applied simultaneously for all cell: synchronous dynamics

- The rule is identical for all sites: it is homogeneous (independent of ***r***)

- Spatial and temporal inhomogeneities can be introduced:
  - Define additional bits of the state $s(\boldsymbol{r},t)$ set to e.g. 1 for given ***r*** and *t* to mark particular locations and times where different rules may apply; E.g. Boundary cells

- New state at t+1: only a function of previous state at t
  - Longer memories can be introduced: dependence on states at t-1, t-2, ...
  - keep a copy of the previous states in the current state by using extra bits: E.g: $s_{t+1}=R(s_t,s_{t-1})$. Define $u_t=\{s_t,s_{t-1}\}$. Then $u_{t+1} = R(u_t)$.

- Boundary conditions
  - Computer: finite lattice -> boundaries.
  - a boundary site does not have the same neighbourhood -> different evolution rule
  - Being or not at the boundary: code at the site and choose rule appropriately!
  - Another possibility: extend the neighbourhood for the sites at the boundary (virtual cells):