

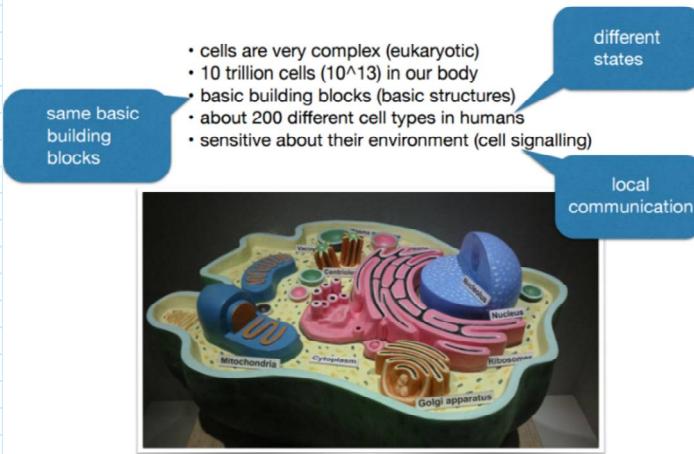
Cellular Systems

22 May 2017 02:14

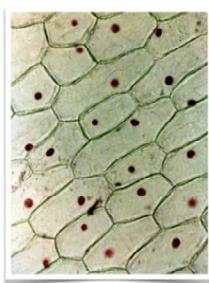
Key idea: simple local rules between discrete cells can lead to complex global structure and/or behaviour.

Biological Cells:

The Cell – A Closer Look

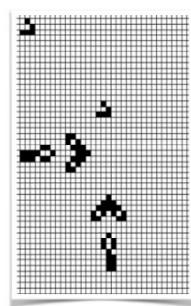
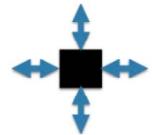


Abstraction



Understanding cell systems

Abstraction



Behaving like cell systems

Self-replicating systems:

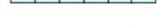
- J. von Neumann's universal constructor.
- Wolfram's elementary cellular automata (CA)

Building blocks of cellular system:

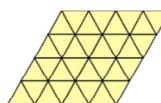
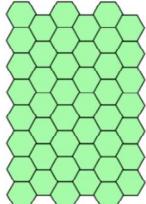
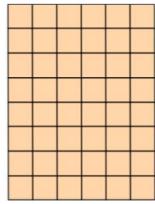
- Virtual space
- Spatial interaction
- Cell state
- Transition rules.

Virtual Space

1D



2D



Spatial interaction

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

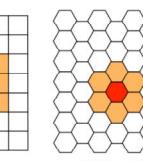
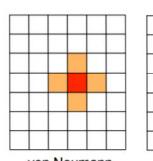
$r=1$ $r=2$...

1D



...

2D



...

von Neumann

Moore

Hexagonal

Transition rule

Each cell has a state, often represented as a number. The value of the state is often represented by cell colours. There can be a special quiescent state s_0 .

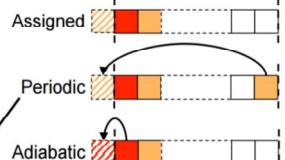
$$\begin{aligned} S &= \{s_0, \dots, s_{k-1}\} \\ &= \{0, \dots, k-1\} \\ &= \{\text{white}, \dots, \text{black}\} \end{aligned}$$

The transition rule is the fundamental

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



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.

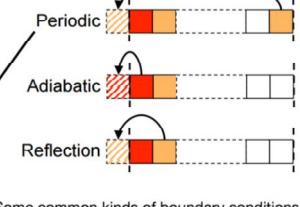
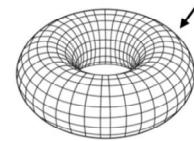
The transition rule can be represented as a **transition table**, although this becomes rapidly impractical.

transition func

$\delta : \text{State} \times [\text{State}] \rightarrow \text{State}$
at each time step,
 $s_{t+1}(i) = \delta(s_t(i), n_s)$
where $n_s = n$ neighbors

may lack the cells required to form the prescribed neighbourhood

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

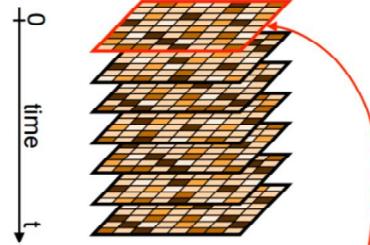
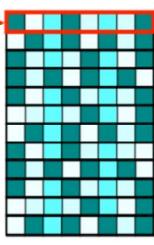
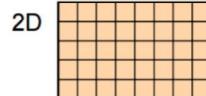


Some common kinds of boundary conditions

Initial conditions

image where y-axis shows development

1D



video of simulation

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)

Example: Traffic Modeling

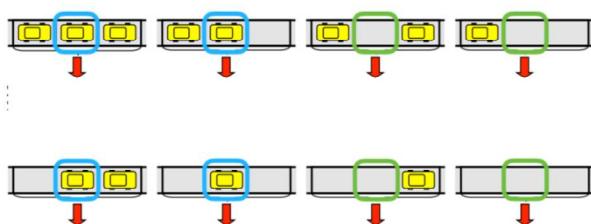


$S = \{s_0, \dots, s_{n-1}\}$
= $\{0, \dots, k-1\}$
= $\{*, \dots, *\}$

k states n cells in the neighborhood
k^n

8 cases

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.

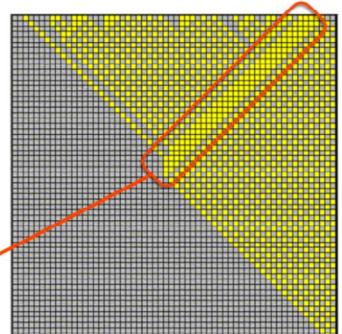


Example: Traffic Jam



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).



t=1 t=2 t=3

key idea: δ moves cars forward, where they can't move forward if they are blocked.

Doesn't account for drivers slowing down before upcoming traffic, and other more complex decisions.

The general rules for designing CA:

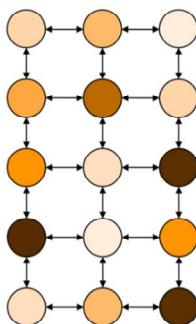
1. Assign CA space geometry
2. Neighborhood geometry
3. Define cell states

2. Neighborhood geometry
3. Define cell states
4. Define δ
5. Assign initial conditions
6. Define boundary conditions
7. Repeatedly update global cells by applying δ .

Informal definition of CA

A Cellular Automaton is:

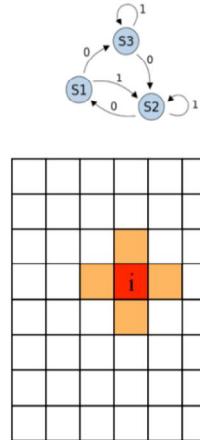
- a **geometrically structured** and **discrete** collection of
- **identical** (simple) systems called **cells**
- that **interact** only **locally**
- with each cell having a **local state** (memory) that can take a **finite number** of values
- and a (simple) **rule** used to **update** the state of all cells
- at **discrete time steps**
- and **synchronously** for all the cells of the automaton (global "signal")



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

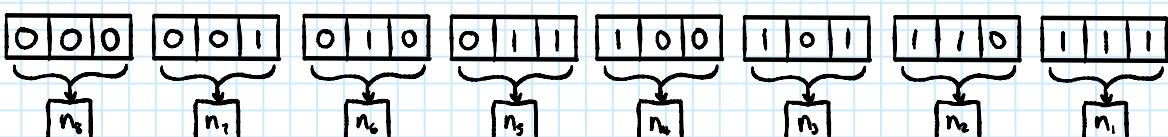


$$s_i(t+1) = \phi(s_j(t); j \in N_i)$$

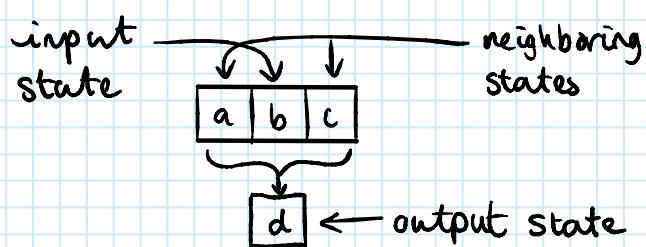
Wolfram's Rule Code for Elementary CA

Elementary CAs are 1D binary CAs, $k=2$ with minimal range, $r=1$

CAs are index with Wolfram's code, Rule n is constructed by taking n 's base-2 representation, $n = \langle n_1, n_2, \dots, n_8 \rangle$, $n_i \in \{0, 1\}$, assuming $0 < n < 256$, the transition rule is defined

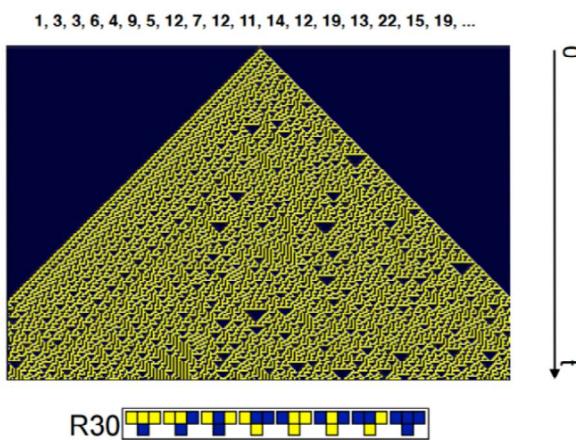


where,

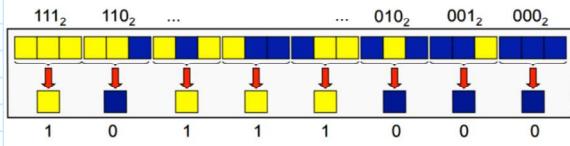


Example of elementary CA

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

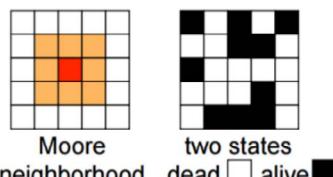


Traffic rule (R187)



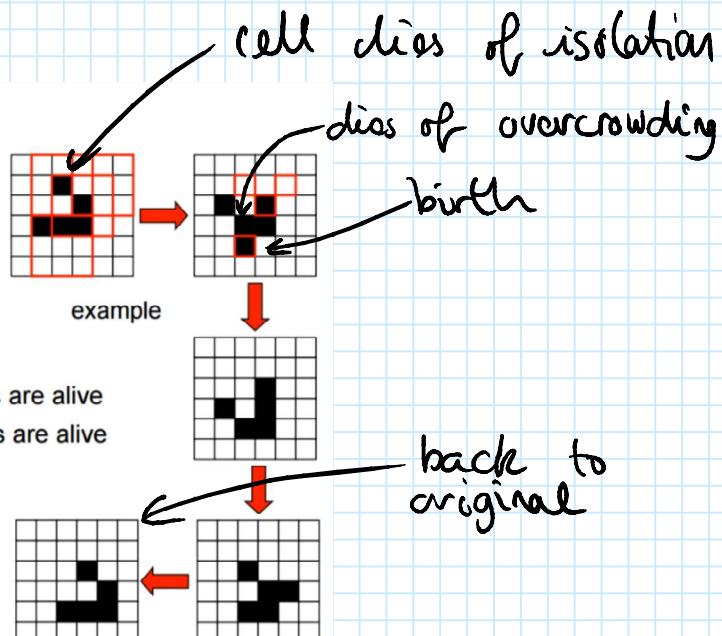
Game of Life (Gol)

Classical 2D CA: Life



Outer totalistic rule (John Conway)

- Birth $\square \rightarrow \blacksquare$ if exactly 3 neighbors are alive
 - Survival $\blacksquare \rightarrow \blacksquare$ if 2 or 3 neighbors are alive
 - Death $\blacksquare \rightarrow \square$
- from "isolation" if 0 or 1 n. a.
from "overcrowding" if more than 3 neighbors are alive
(often coded as "rule 23/3")



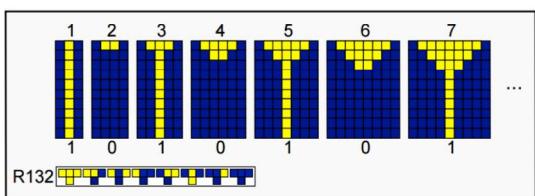
- Gol is Turing complete
- it is computationally irreducible and universal

More examples:

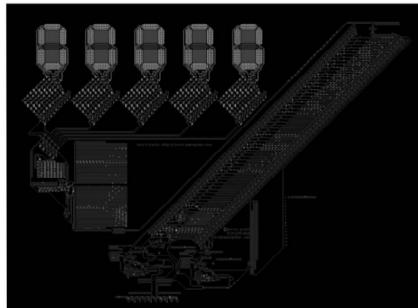
Computation with CA

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



The Wireworld computer

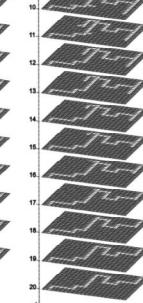


Example: CA maze solver



- 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

Particle CA

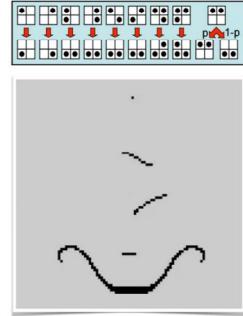


CA can be used to model phenomena that involve **particles**. The **transition rule** can be specified in terms of the motion of particles within blocks of two by two cells (**block rules**).

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



Reversibility:

Some CAs are reversible - that is given some global state A it is possible to find some other configuration B such that $A = \delta(B)$ (applying the transition rule globally).

Beyond the trivial cases (R0, R8, R64, R128), reversibility is achieved if the CA conserves state information. For example the traffic rule (R187) is reversible because car "mass" is conserved.

Variants, Extensions

Probabilistic CA

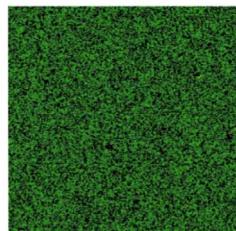
So far we have considered only **deterministic CA**.

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



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**)
- ...