ECM3412
Nature-Inspired Computation

Artificial Life and Cellular Automata

Today's Plan

- Introduction to Artificial Life
- Cellular Automata
  - Cells
  - States
  - State transition rules
  - Neighbourhoods
  - Running a CA
  - Stopping Criteria
- Workshop/Demo

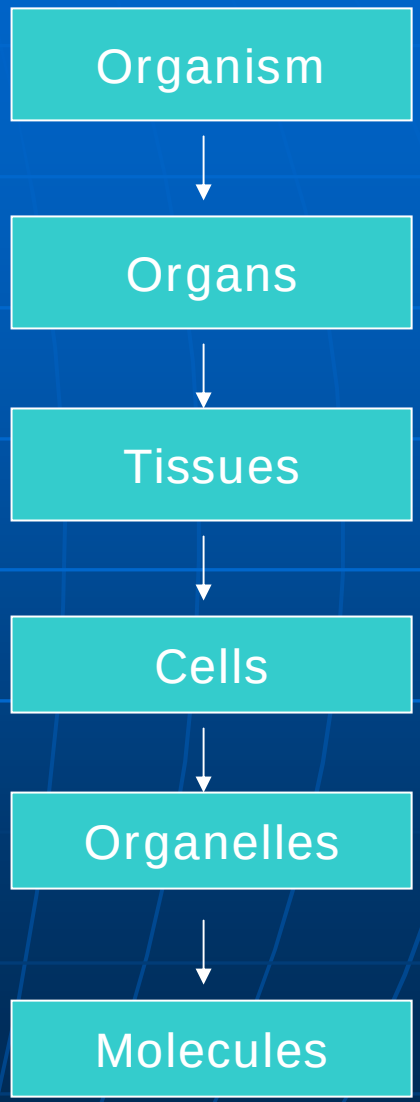
Artificial Life (ALife)

- To this point, we've used nature as the inspiration for algorithms
  - Genetic algorithms – evolution
  - Ant colony algorithms – ant colonies
  - Particle swarm optimisation – flocking/swarming behaviours
  - Neural networks – the brain
  - Artificial Immune Systems – the human immune system

- Artificial life is somewhat different
  - Computer systems simulating life

Artificial Life II

- Artificial life is about better understanding what it is to be alive.
- Biology is primarily reductionist – an explanation of a behaviour or phenomenon at one level can be explained by further investigation at the level below (see left)
- This is a reasonable top-down approach.
- Artificial life takes a bottom-up approach.



Artificial Life III

- Study into Alife is conducted primarily at 3 levels
  - Wetware – using bits from biology (e.g. RNA, DNA) to investigate evolution
  - Software (what we have been/will be dealing with) – simulating biological systems
  - Hardware – for instance, robotics.

- And with 2 distinct philosophies
  - Strong ALife – life is not just restricted to a carbon-based chemical process. Life can be 'created' *in silico*.
  - Weak ALife – computer simulations are just that, simulations and investigations of life

Artificial Life IV

- In fact, all the techniques we've seen so far can be considered Artificial Life in so much as:
  - Genetic algorithms are simulating or actually doing evolution
  - Ant colony algorithms are simulating the real behaviour of ants
  - Particle swarm algorithms are simulating the real behaviour of flocks
- What if we consider strong Alife?
  - Actual evolution, ants and flocks?
- Almost certainly not, but what about a 'life' Turing Test?

Artificial Life V

- We will be looking today at a software-based technique – cellular automata.

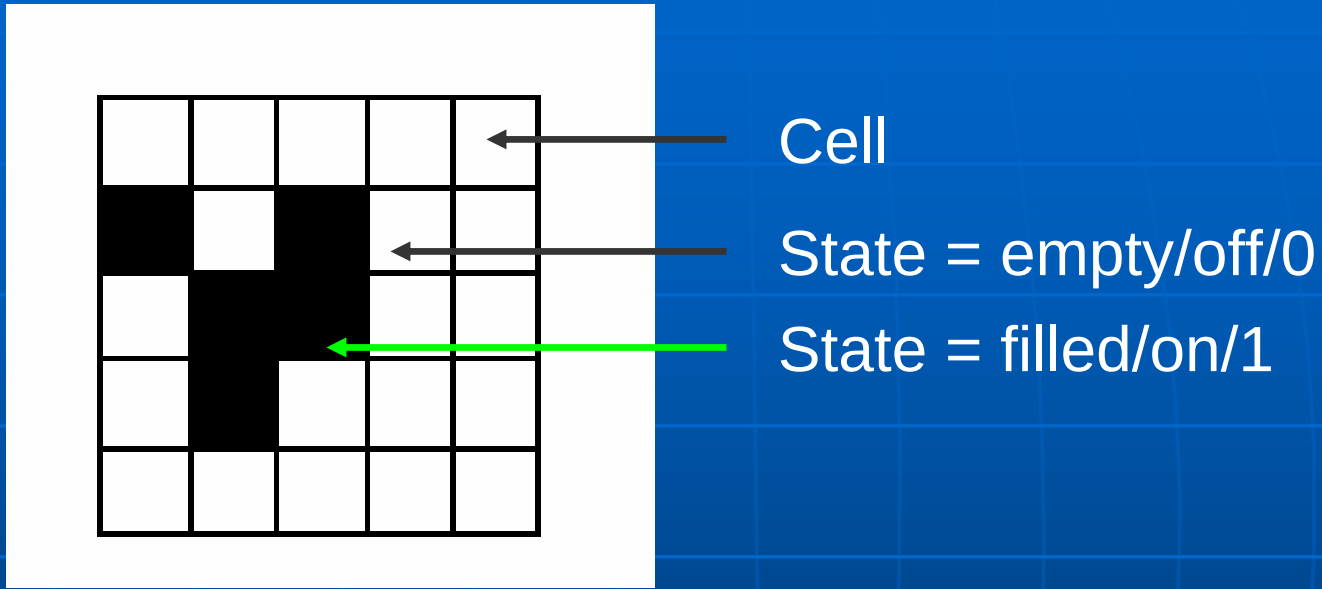
- One of the original Alife techniques, cellular automata embodies the bottom-up approach
- It is involved with the emergent behaviour of collections of simple elements
  - Similar to the ‘emergent’ behaviour seen in swarm intelligence

- These automata are mainly used for the simulation of biological systems, although they can be used for optimisation

Cellular Automata Introduction

- Cellular Automata originally devised in the late 1940s by Stan Ulam (a mathematician) and John von Neumann.
- Originally devised as a method of representing a stylised universe, with rules (e.g. laws of thermodynamics) acting over the entire universe.
- Have subsequently been used for a wide variety of purposes in simulating systems from chemistry and physics
- CAs have started to be used in bioinformatics and other areas
- Consist of a grid or lattice of ‘cells’

Cellular Automata

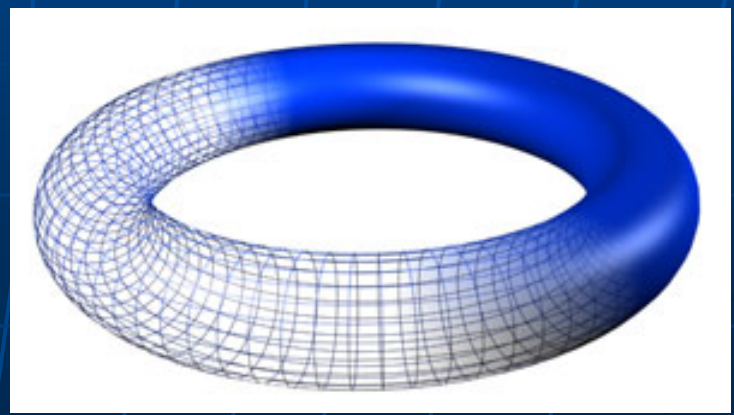


- An automaton consists of a grid/lattice of cells each of which can be in a (normally small and finite) number of states
- The figure shows a 5x5 automaton where each cell can be in a filled or empty state.

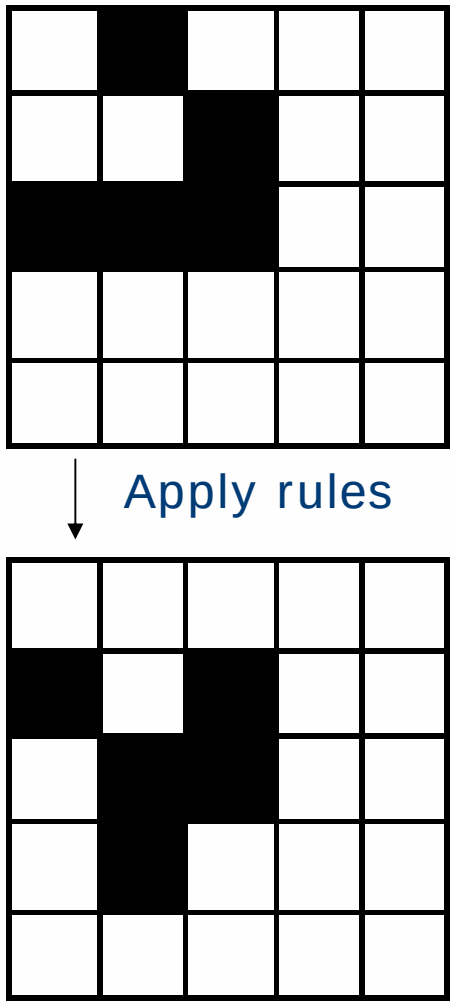
Cellular Automata II

- An automaton can be
  - 1-D (i.e. just a line of cells)
  - 2-D (as we have already seen)
  - 3-D+ there is no theoretical limit to the number of dimensions

- Also, automata are often toroidal (cells ‘wrap around’ to the other side)



Execution



- The CA ‘runs’ by changing the states of the cells by the state transition rules (next slide).
- These state transition rules depend on the state of the cell and it’s neighbours
- Every cell in the automaton has it’s rules applied before the automaton is updated
- Each timestep the automaton can be seen as a system configuration for that particular snapshot in time.

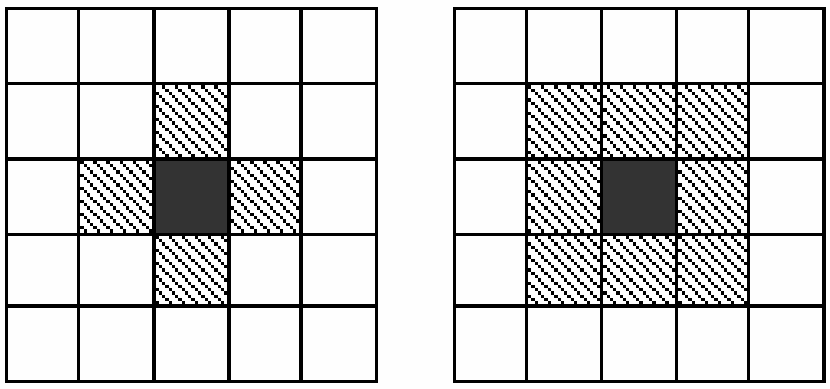
State Transition Rules

- The states of an automaton change over time in discrete *timesteps*
- The state of each cell is modified in parallel at each timestep according to the *state transition rules*
- These determine the new states of each of the cells in the next timestep from the states of that cells *neighbours*

```
For (int i=0 to CellCount)
{
  Cell[i].State[t+1] = STR(Cell[i].Neighbour.State[t]
}
```

Neighbourhoods

- Neighbourhoods are important as mechanisms for controlling the execution of the CA
- Neighbourhoods determine the extent of the interaction between cells in the grid
- Two popular neighbourhoods are:



Conway's Life

- Conways "Game of Life" is the most often cited CA. The rules used are:
  1. If a cell is off (state 0) and exactly three of its neighbours are on (state 1) then that cell becomes on (state 1) in the next timestep, otherwise it remains off.
  2. If a cell is on and either two or three of its neighbours are then on the next timestep, that cell remains on, otherwise it is turned off.
- Even a simple set of rules like this can have unexpected results.

Conway's Game of Life

- Probably the most famous cellular automaton
- Is "nature-inspired"
- The rules are meant to represent life itself
  - A dead cell will come to life (be born) if 3 of it's neighbours are alive
  - Alive cells must not be overcrowded (more than 3 alive neighbours) or lonely (less than 2 alive neighbours) otherwise they will die.

Workshop

Cellular Automata

- Important properties which make a CA a CA:
  - Localism
    - States are updated based on the properties of the neighbourhood
  - Parallelism
    - The state of every cell is updated in parallel
  - Homogeneity
    - The same set of rules is applied across the automaton
- These properties distinguish cellular automata from other types of automata or algorithm

Types of Cellular Automata

- It is not possible to predict, in advance, what behaviour will be displayed by the CA given a set of rules.
- There are a number of possible states into which a CA can descend into
- **Wolfram** proposed a classification scheme based on these criteria:

1. Evolution leads to a homogeneous state.
2. Evolution leads to a set of separated simple stable or periodic structures.
3. Evolution leads to a chaotic pattern.
4. Evolution leads to complex localized structures, sometimes long-lived.

Next Time

- Applications of cellular automata