

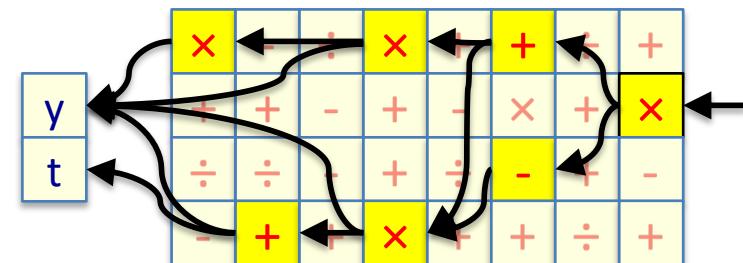
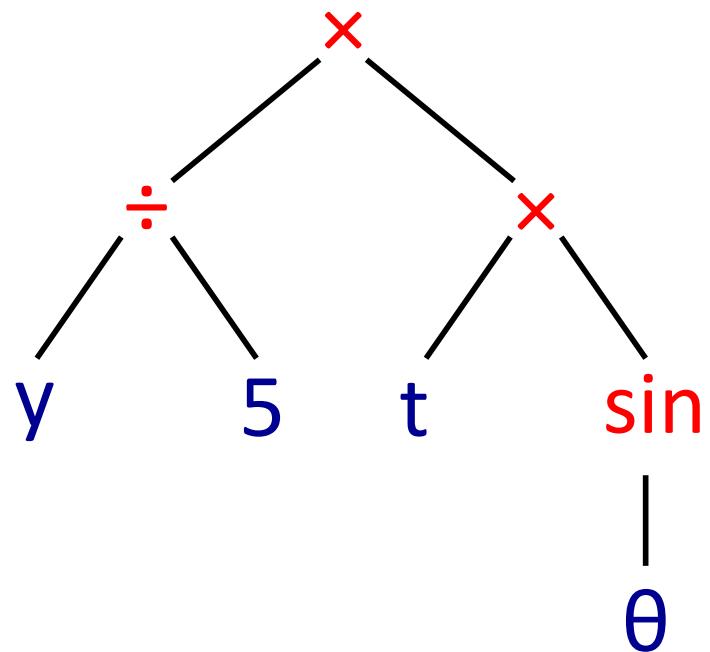
# Cellular Automata

---

Dr. Michael Lones  
Room EM.G31  
[M.Lones@hw.ac.uk](mailto:M.Lones@hw.ac.uk)

# Previously

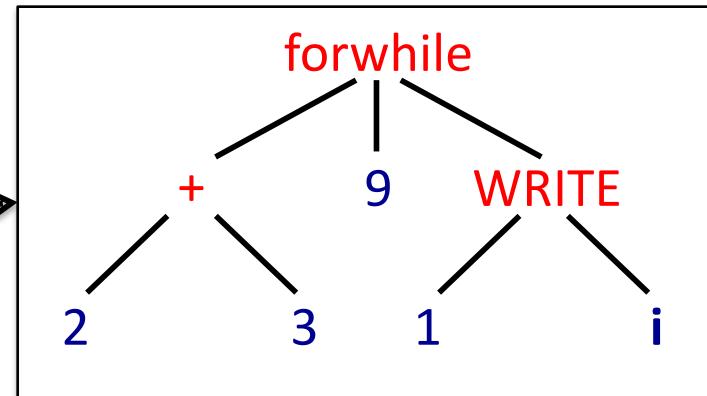
- ◊ Genetic programming
  - ▷ Is about evolving computer programs
  - ▷ Mostly conventional representations: tree, graph, list
  - ▷ Mostly conventional issues: syntax, language



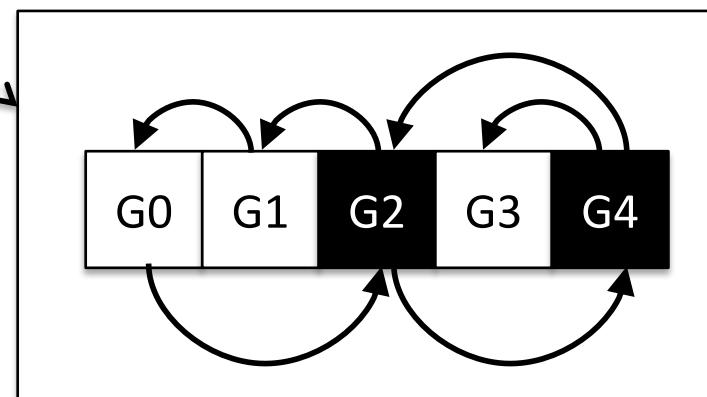
10	18	40	27	6	39	21	47	13	31	19	8
----	----	----	----	---	----	----	----	----	----	----	---

# This Week's Lecture

- ◊ Models of how biological systems "compute"
  - ▷ They represent computation in a very different way



X



# Computation

- ◊ Conventional
  - ▷ Centralised
  - ▷ Top-down
  - ▷ Halting
  - ▷ Static
  - ▷ Exact
  - ▷ Fragile
  - ▷ Synchronous



- ◊ Biological
  - ▷ Distributed
  - ▷ Bottom-up (emergent)
  - ▷ Ongoing
  - ▷ Dynamical
  - ▷ Inexact
  - ▷ Robust
  - ▷ Asynchronous

- ◊ See Mitchell, “**Biological Computation**,” 2010  
[https://pdxscholar.library.pdx.edu/compsci\\_fac/2/](https://pdxscholar.library.pdx.edu/compsci_fac/2/)

# Cellular Automata (CA)

## ◊ What is a cellular automaton?

- ▷ A model of distributed computation
  - of the sort seen in biology
- ▷ A demonstration of “emergence”
  - **complex** behaviour emerges from interactions between **simple** rules
- ▷ Developed by Ulam and von Neumann in the 1940s/50s
- ▷ Popularised by John **Conway**’s work on the ‘Game of Life’ in the 1970s
- ▷ Significant later work by Stephen **Wolfram** from the 1980s onwards



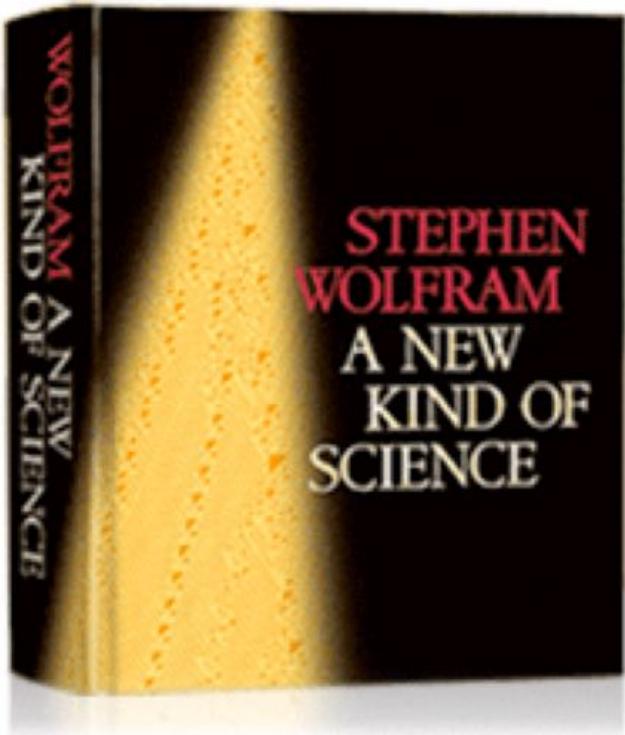
Stanislaw Ulam



John von Neumann

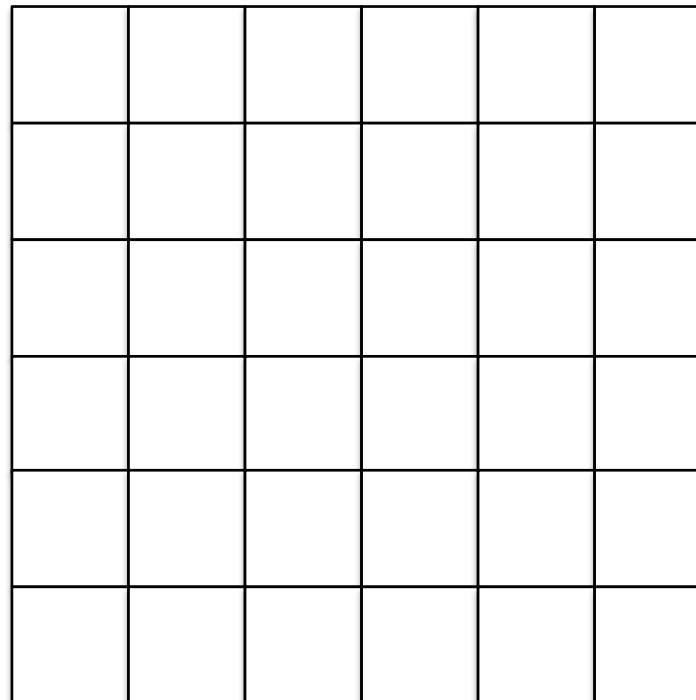
# Book – free to read online

- ▶ Stephen Wolfram, **A New Kind of Science**, 2002
- ▶ <https://www.wolframscience.com/nksonline/toc.html>



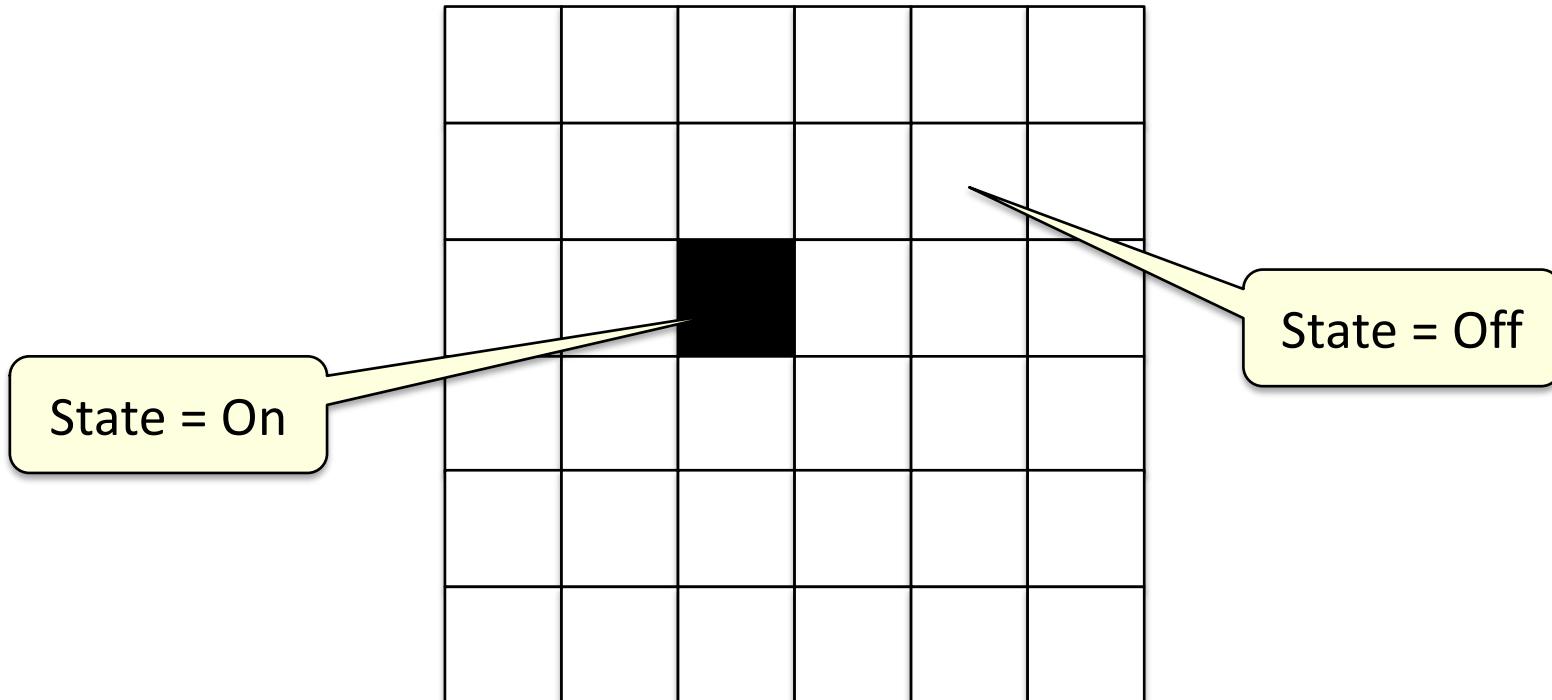
# Definition

- ◊ Computation takes place on a grid, which may have 1, 2 or more dimensions, e.g. a 2D CA:



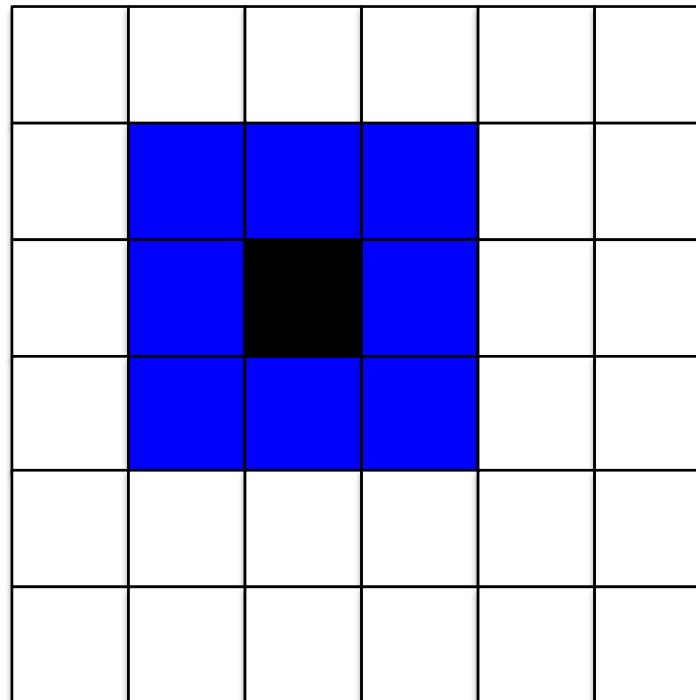
# Definition

- ❖ At each grid location is a **cell**
  - ▷ Which has a **state**
  - ▷ In many cases this is binary:



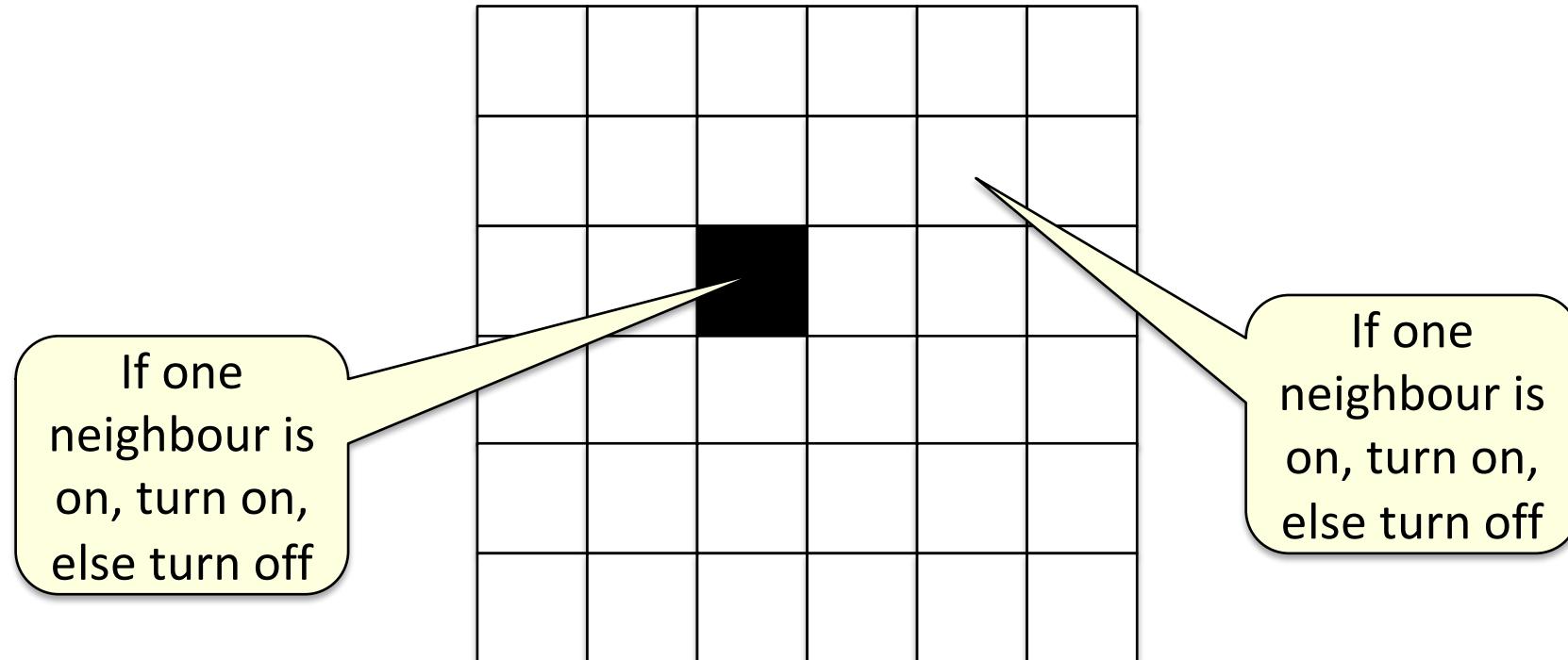
# Definition

- ❖ Each cell contains an **automaton**
  - ▷ Which observes a **neighbourhood** around the cell



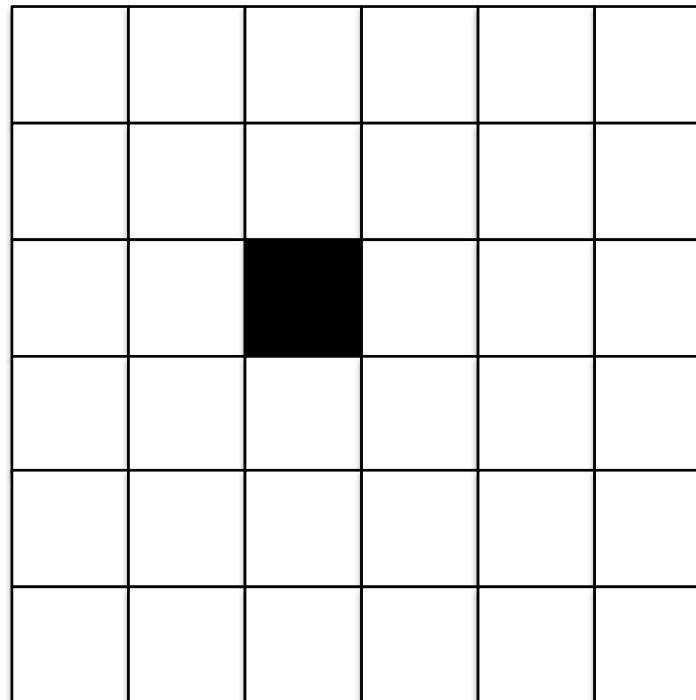
# Definition

- ❖ Each cell contains an automaton
  - ▷ and applies an **update rule** based on this neighbourhood
  - ▷ Every automaton uses the same update rule



# Definition

- ◊ The CA is run over a number of discrete time steps
  - ▷ At each time step, each automaton applies its update rule
  - ▷ Time = 0

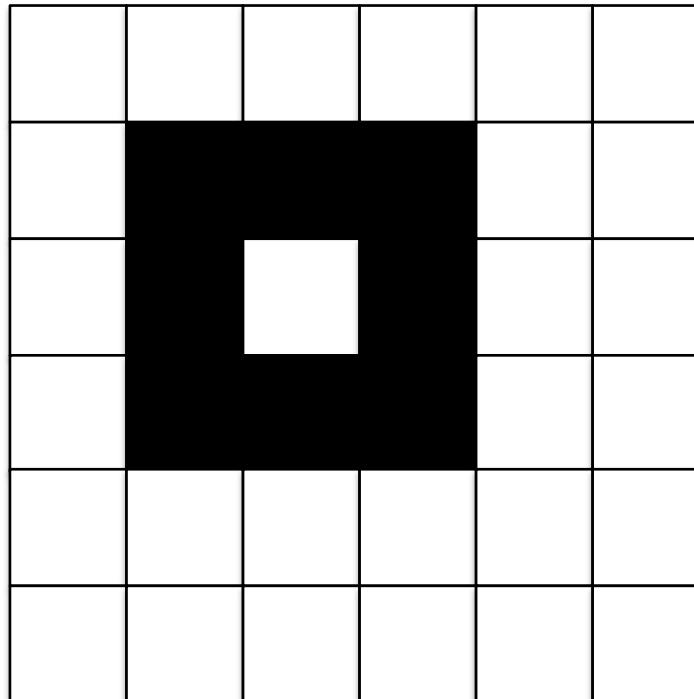


If one neighbour is on, turn on, else turn off

# Definition

- ❖ The CA is run over a number of discrete time steps
  - ▷ At each time step, each automaton applies its update rule
  - ▷ Time = 1

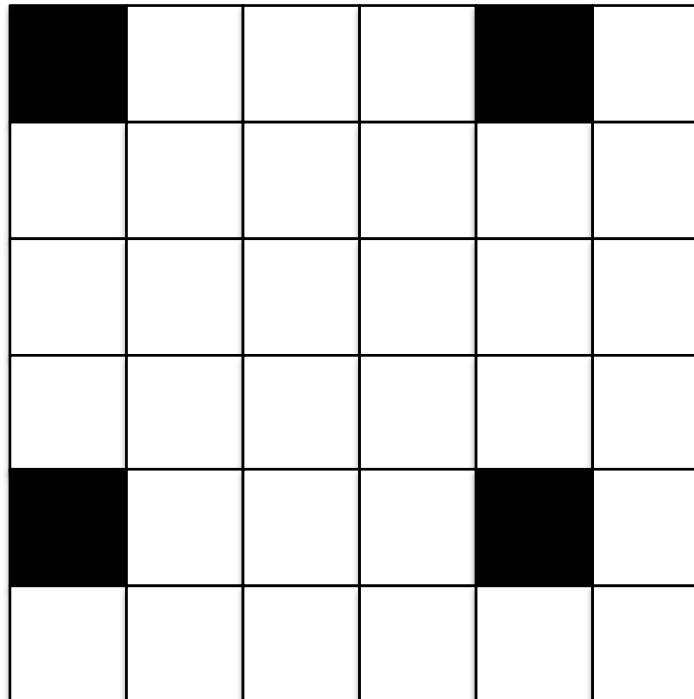
If one neighbour is on, turn on, else turn off



# Definition

- ❖ The CA is run over a number of discrete time steps
  - ▷ At each time step, each automaton applies its update rule
  - ▷ Time = 2

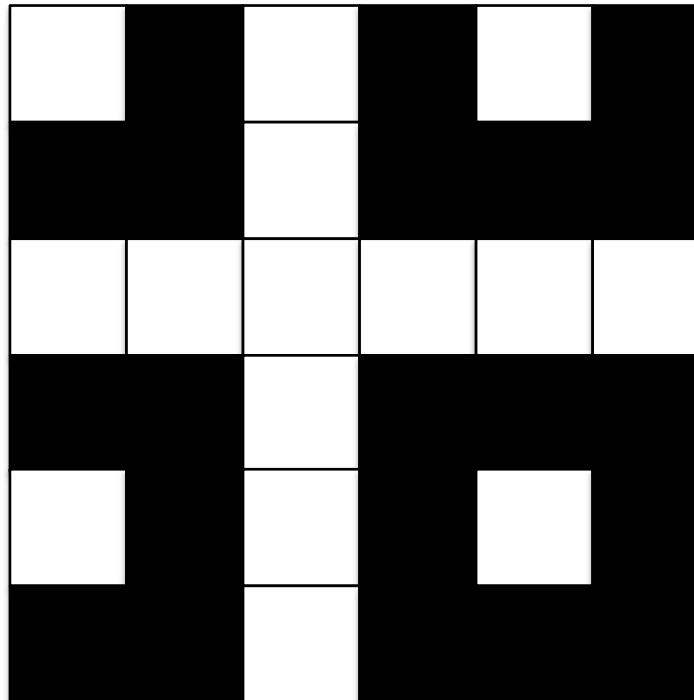
If one neighbour is on, turn on, else turn off



# Definition

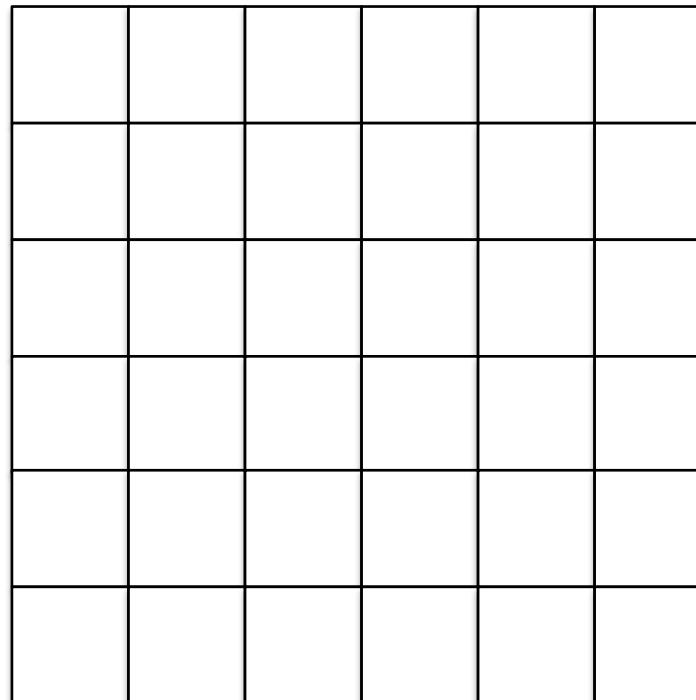
- ❖ The CA is run over a number of discrete time steps
  - ▷ At each time step, each automaton applies its update rule
  - ▷ Time = 3

If one neighbour is on, turn on, else turn off



# Definition

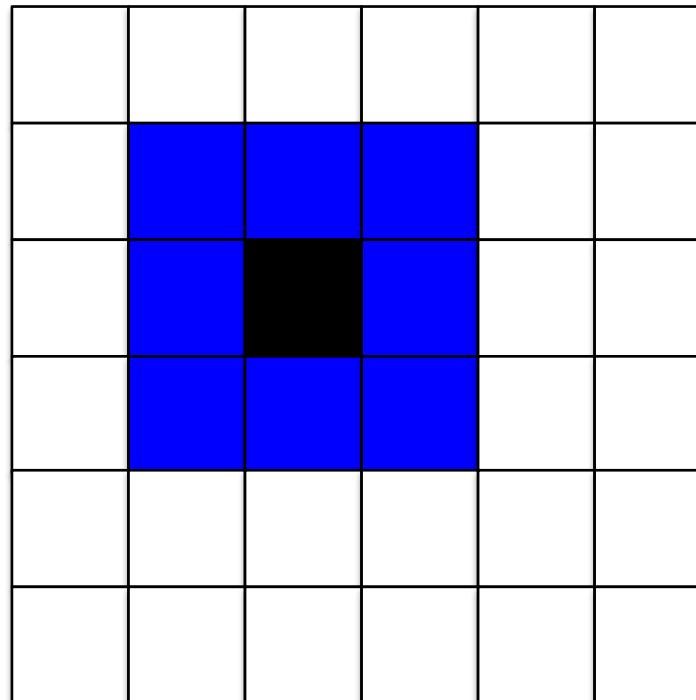
- ◊ The CA is run over a number of discrete time steps
  - ▷ At each time step, each automaton applies its update rule
  - ▷ Time = 4




If one neighbour is on, turn on, else turn off

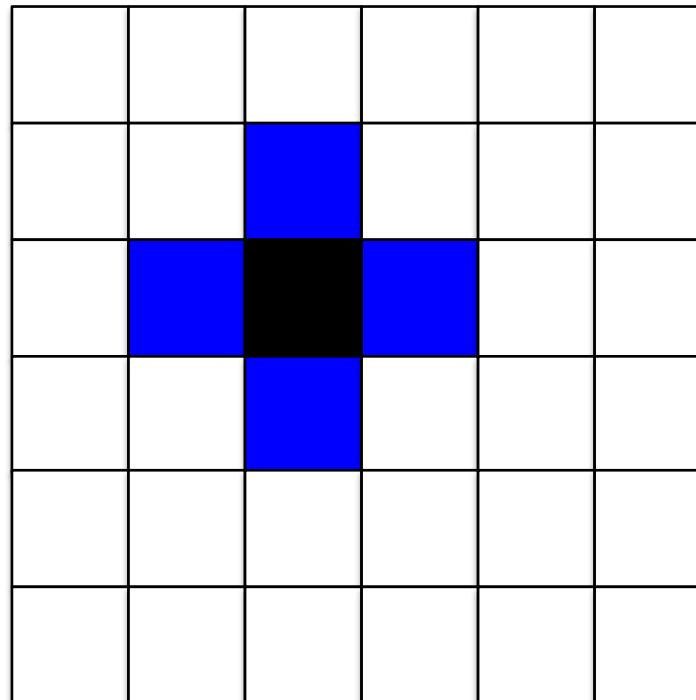
# Neighbourhoods

- ❖ A number of different neighbourhoods are used in CAs
  - ▷ This is called a **Moore** neighbourhood



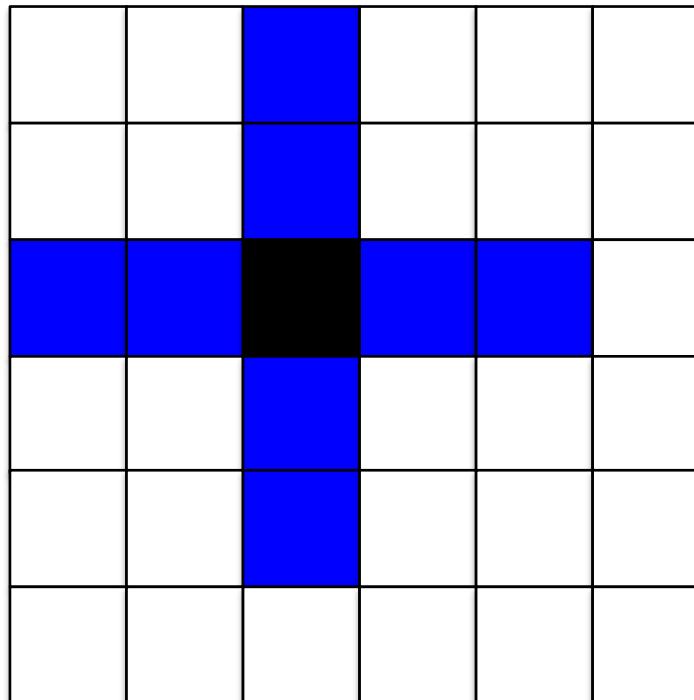
# Neighbourhoods

- ❖ A number of different neighbourhoods are used in CAs
  - ▷ This is called a **von Neumann** neighbourhood



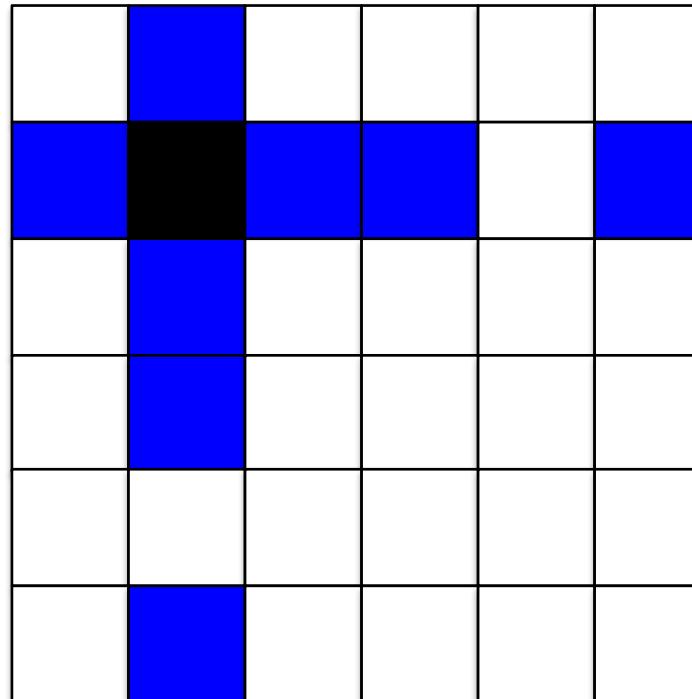
# Neighbourhoods

- ❖ A number of different neighbourhoods are used in CAs
  - ▷ This is called an **extended von Neumann** neighbourhood



# Neighbourhoods

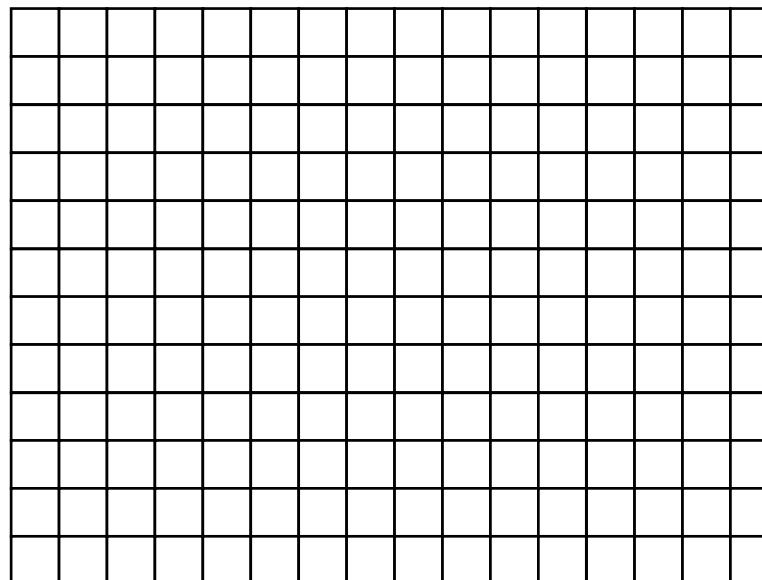
- ❖ A number of different neighbourhoods are used in CAs
  - At the edges, **toroidal** neighbourhoods are often used
  - These are known as **periodic boundary conditions**



Any Questions?

# Simple Example

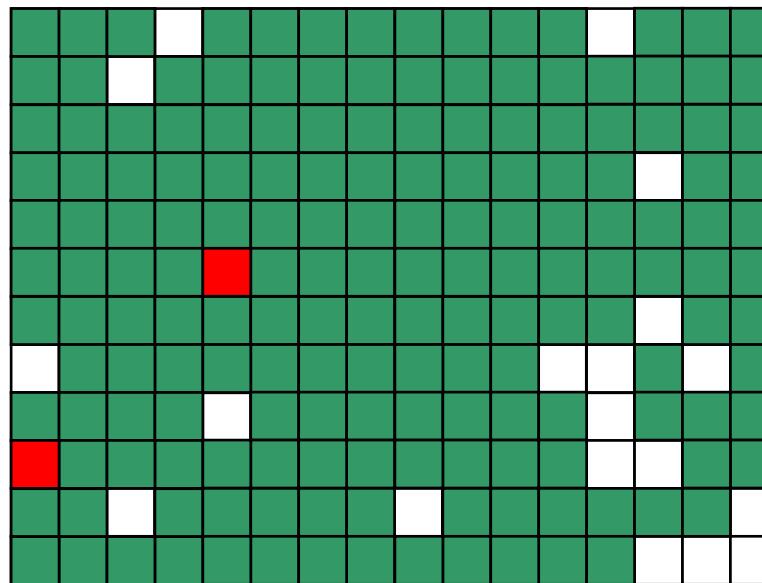
- ❖ Modelling how a forest fire spreads\*
  - ▷ A forest is modelled as a 2D grid of automata
  - ▷ A tree may or may not grow in each cell



\*Thanks to David Corne for this example

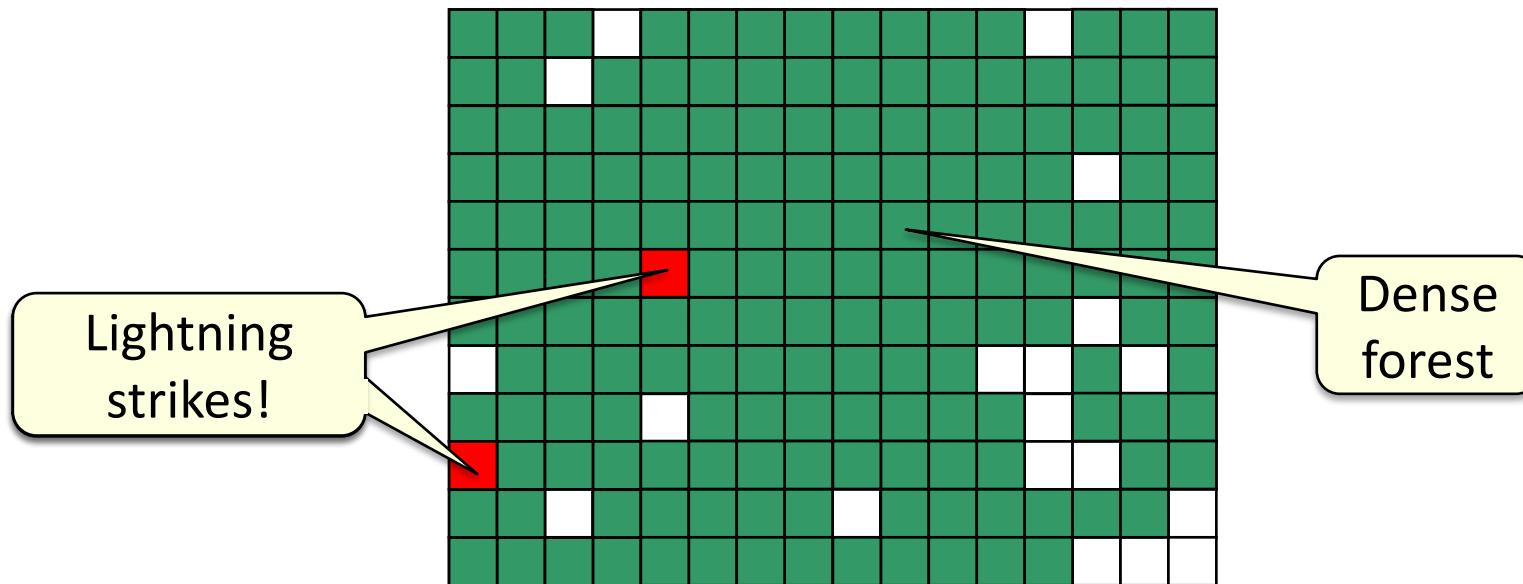
# Simple Example

- ◊ Each automaton has one of three states
  - ▷ **Empty** indicating no tree
  - ▷ **ok\_tree** indicating a healthy tree
  - ▷ **fire\_tree** a tree that's on fire



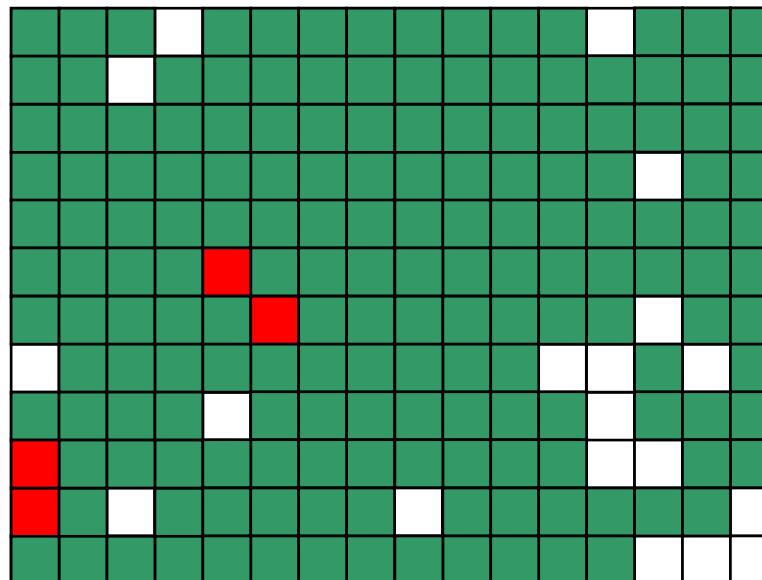
# Simple Example

- ❖ Each automata applied this rule in a Moore neighbourhood
    - ▷ If a tree is not on fire, and has  $n$  neighbours on fire, it catches fire with probability  $n/8$ . If on fire for 3 steps, a tree dies
    - ▷ Time = 0



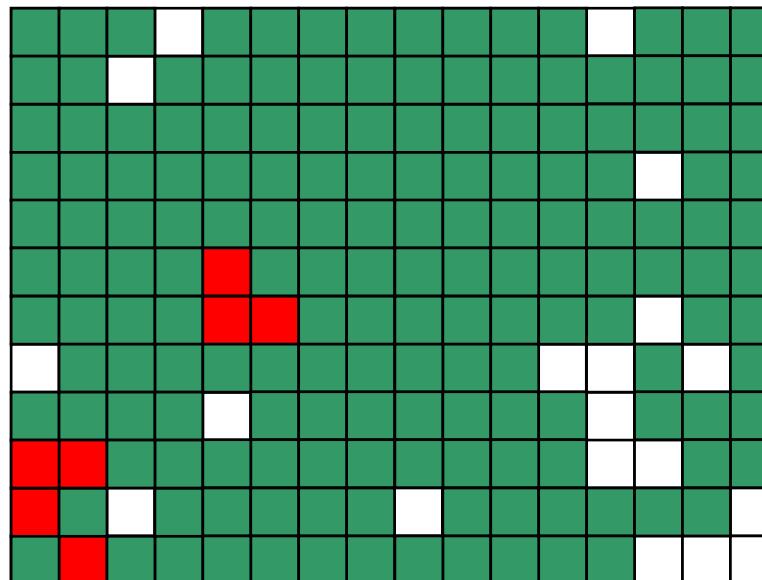
# Simple Example

- ❖ Each automata applied this rule in a Moore neighbourhood
  - If a tree is not on fire, and has  $n$  neighbours on fire, it catches fire with probability  $n/8$ . If on fire for 3 steps, a tree dies
  - Time = 1



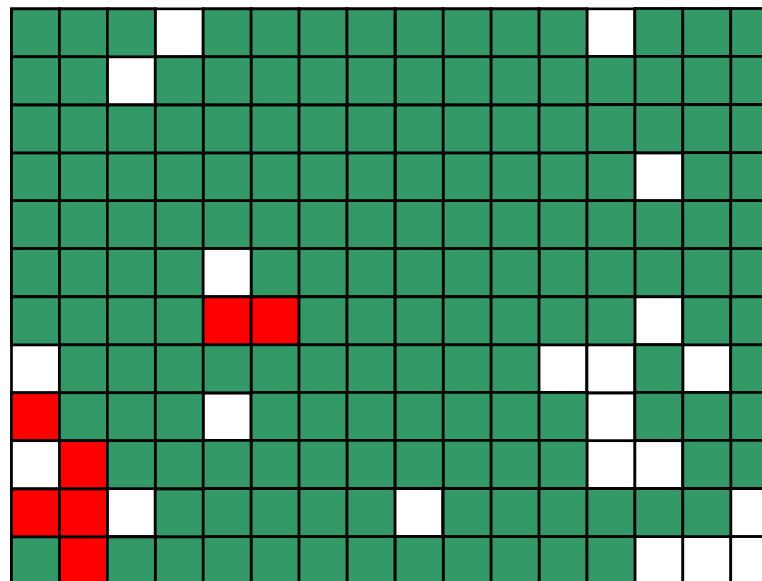
# Simple Example

- ❖ Each automata applied this rule in a Moore neighbourhood
  - If a tree is not on fire, and has  $n$  neighbours on fire, it catches fire with probability  $n/8$ . If on fire for 3 steps, a tree dies
  - Time = 2



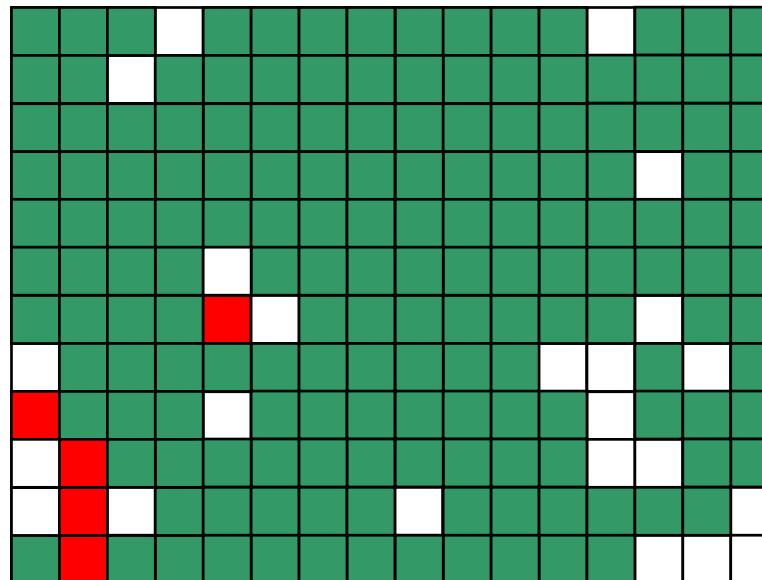
# Simple Example

- ❖ Each automata applied this rule in a Moore neighbourhood
  - ▷ If a tree is not on fire, and has  $n$  neighbours on fire, it catches fire with probability  $n/8$ . If on fire for 3 steps, a tree dies
  - ▷ Time = 3



# Simple Example

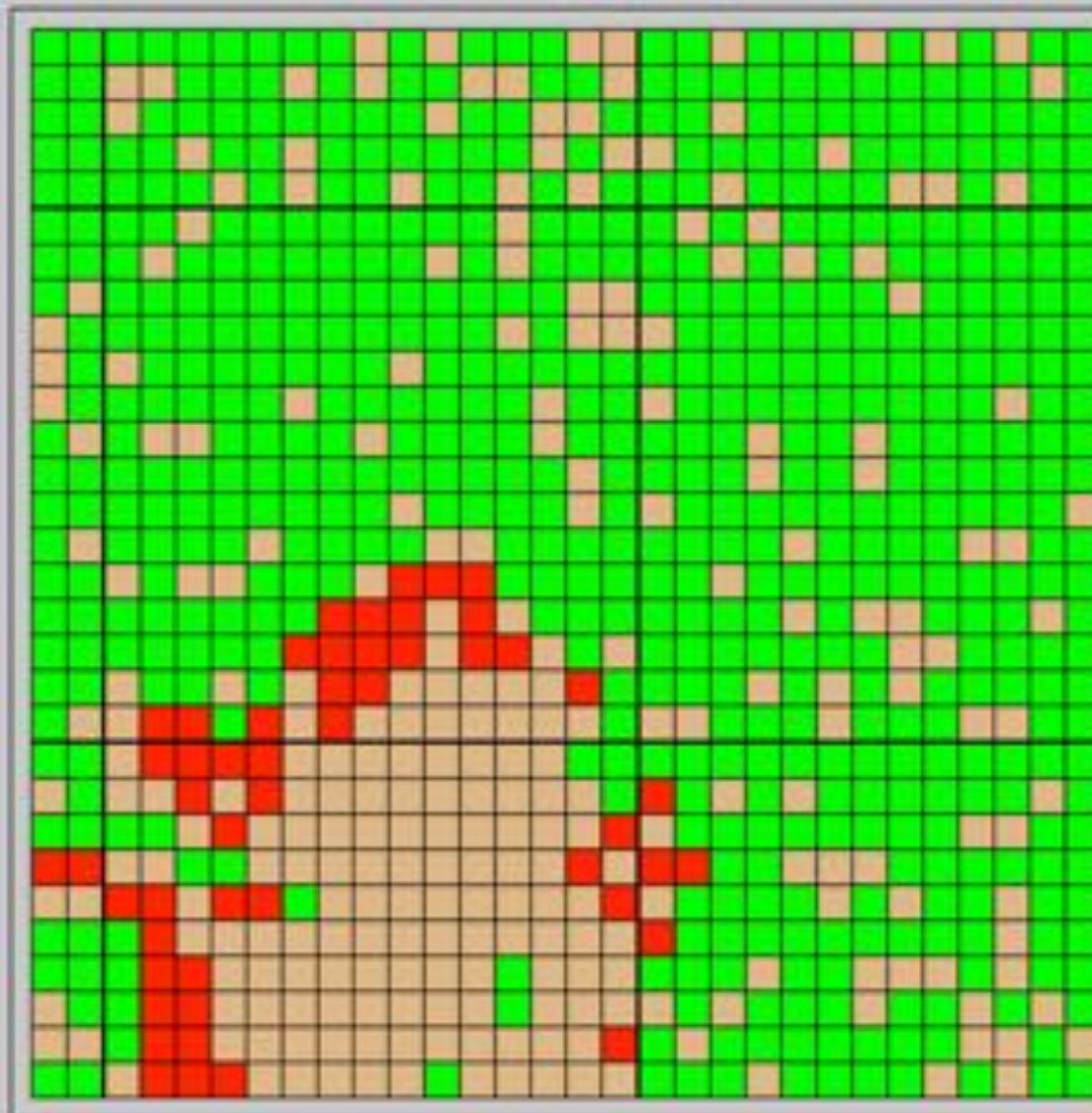
- ❖ Each automata applied this rule in a Moore neighbourhood
  - If a tree is not on fire, and has  $n$  neighbours on fire, it catches fire with probability  $n/8$ . If on fire for 3 steps, a tree dies
  - Time = 4



Enter grid size: 30

Enter tree density: 0.8

Enter burn life: 5



Initialise

Run

Pause

grid size: 30

tree density: 0.80

burn life: 5

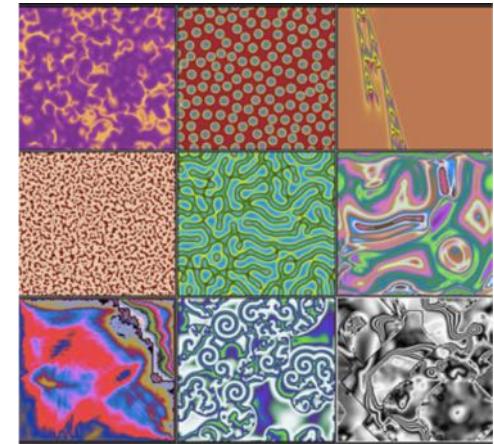
cell size : 17.07

A cellular automaton simulating forest fire. With 'burn life' at 5, what density of trees save the forest?  
Try it.

# Cellular Automata (CA)

## ◊ What are cellular automata used for?

- ▷ Modelling spatial processes
  - e.g. forest fires, disease spread
- ▷ Modelling physical processes
  - e.g. crystal formation, thermodynamics
- ▷ Modelling biological processes
  - e.g. pattern formation, self-replication
- ▷ Solving computational problems
  - e.g. random number generators, ciphers
- ▷ Parallel processing architectures
  - e.g. systolic arrays, Connection Machine →



<http://www.rudyrucker.com>

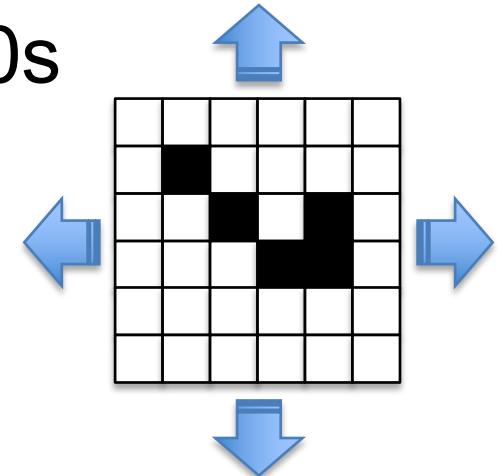


[www.mission-base.com/tamiko](http://www.mission-base.com/tamiko)

# Conway's Game of Life

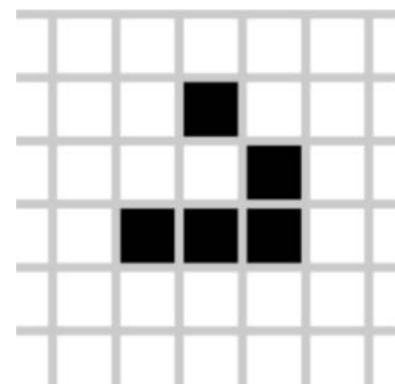
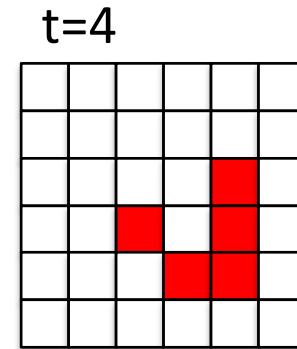
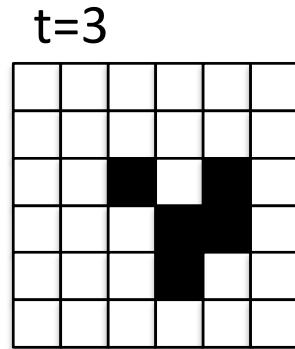
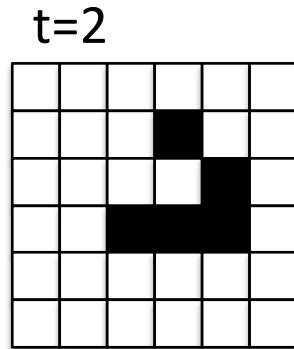
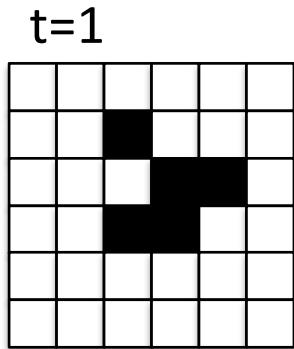
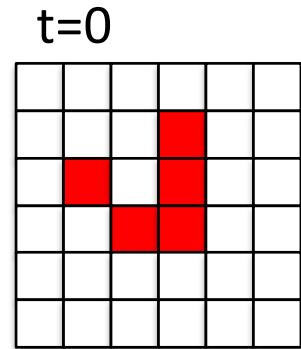
# Conway's Game of Life

- ◊ Developed by John Conway in the 1970s
  - ▷ A very simple model of a living system
  - ▷ Surprisingly complex behaviour
  - ▷ Led to wider interest in CAs
  
- ◊ 2 states (**live**, **dead**), Moore neighbourhood, 4 rules:
  - ▷ A live cell with <2 live neighbours dies (*under-population*)
  - ▷ A live cell with 2-3 live neighbours remains alive
  - ▷ A live cell with >3 live neighbours dies (*over-crowding*)
  - ▷ A dead cell with 3 live neighbours becomes a live cell (*reproduction*)



# Spaceships

- ◇ Moving elements that emerge from these rules
  - ▷ The most famous is the glider:



# Spaceships

- ◊ Moving elements that emerge from these rules
  - ▷ Some more complex examples:



[http://en.wikipedia.org/wiki/File:Animated\\_spaceships.gif](http://en.wikipedia.org/wiki/File:Animated_spaceships.gif) (animated)

# Guns

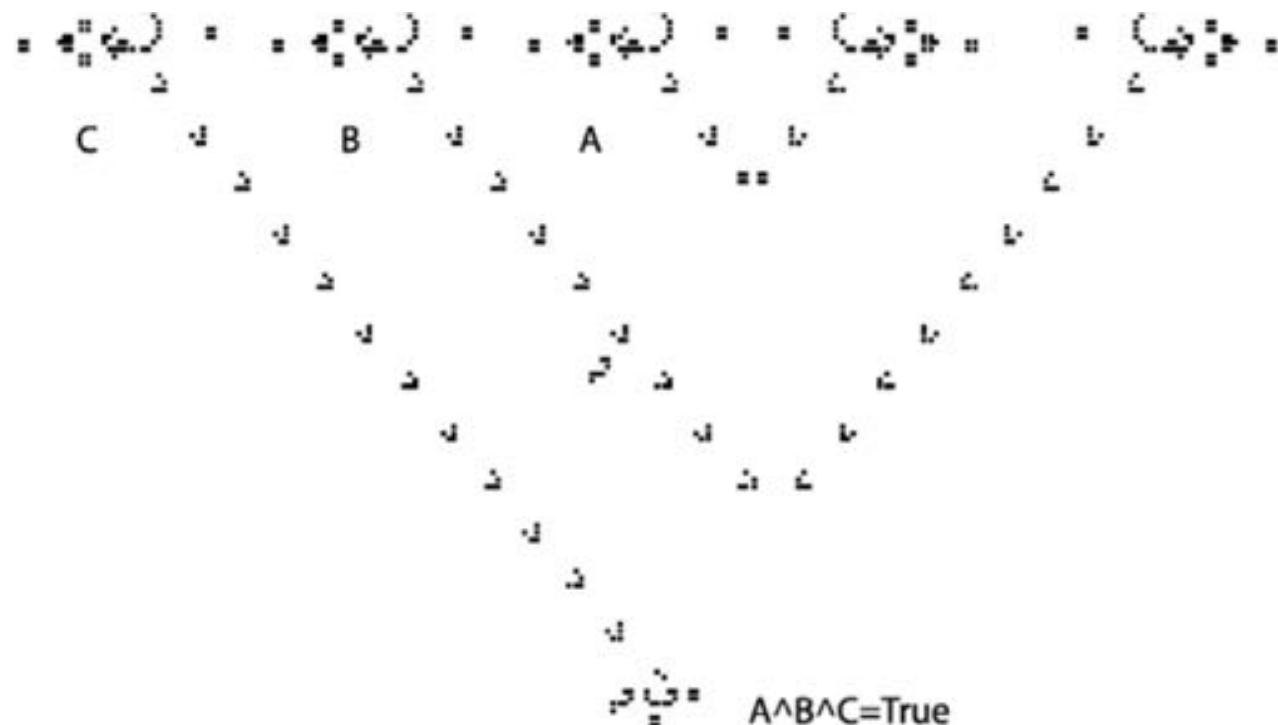
- ◊ Structures that generate streams of spaceships
  - ▷ Gosper's glider gun is the smallest known example:



[http://en.wikipedia.org/wiki/File:Gosper's\\_glider\\_gun.gif](http://en.wikipedia.org/wiki/File:Gosper's_glider_gun.gif) (animated)

# Turing Completeness

- ◊ Game of Life famously shown to be Turing complete
  - ▷ i.e. capable of universal computation
  - ▷ Proven by implementing logic gates with gliders



# Turing Completeness

- ◊ E.g. an AND gate:

## AND Gate

an output signal only with both inputs

A	B	Out
0	0	0
1	0	0
0	1	0
1	1	1

For interest only / Not examinable

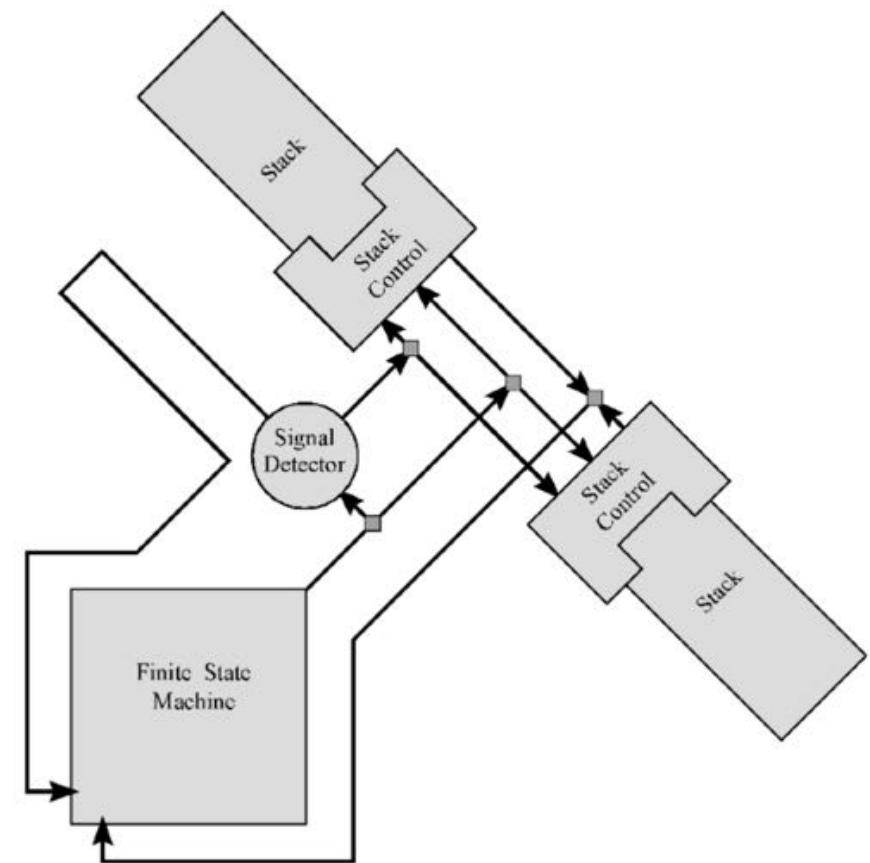
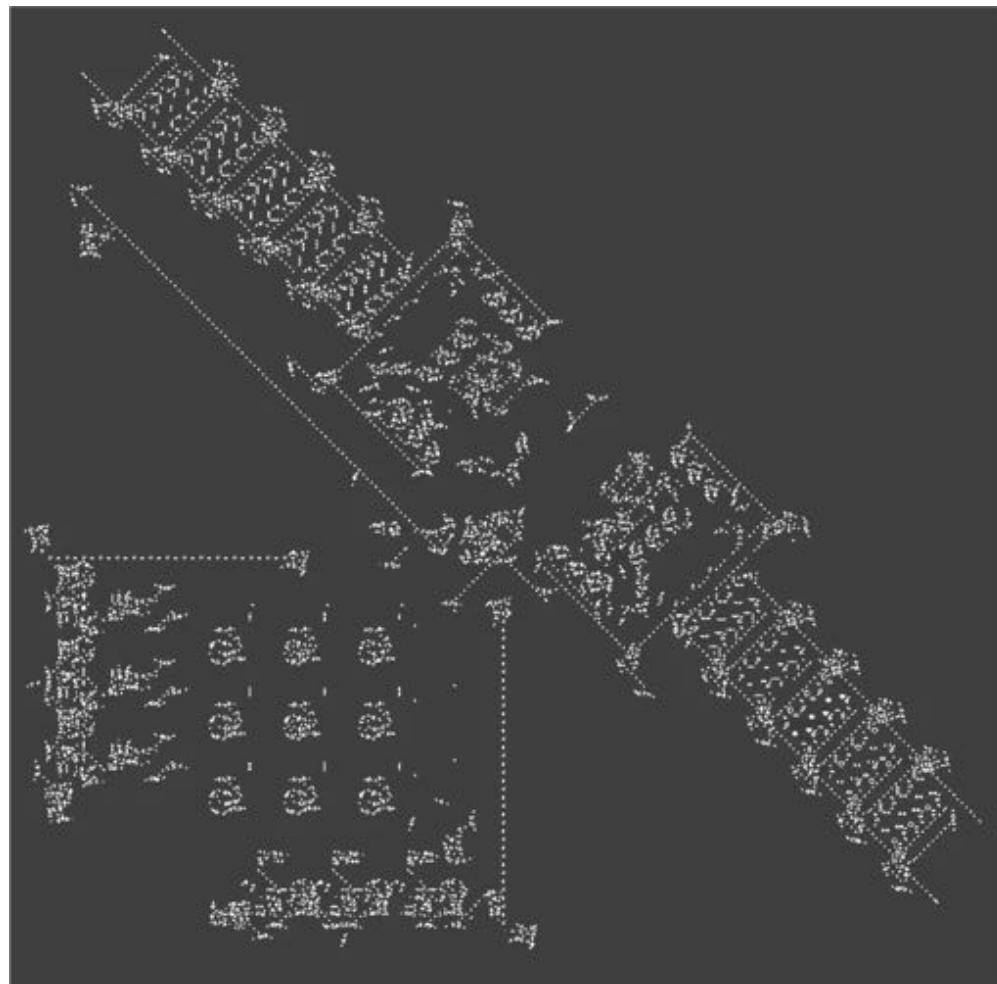
# Collision-Based Computing



<https://www.youtube.com/watch?v=KIRoZ0zTXvw>

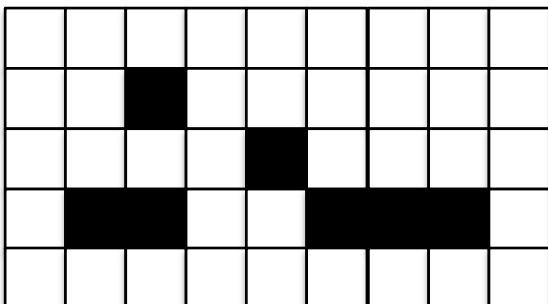
# Turing Machine

- ❖ Rendell, A Universal Turing Machine in Conway's Game of Life
  - ▷ [http://uncomp.uwe.ac.uk/CAAA2011/Program\\_files/764-772.pdf](http://uncomp.uwe.ac.uk/CAAA2011/Program_files/764-772.pdf)



# Methuselahs

- ◊ Patterns that grow and take a long time to stabilise
  - ▷ Complexity emerges from simple rule and initial state
  - ▷ Can be seen as carrying out a complex computation
- ◊ Acorn: size 7, grows to 1057, lasts 5206 time steps
  - ▷ Stable pattern consists of 41 blinkers, 4 traffic lights, 34 blocks, 30 beehives, 1 honey farm, 13 gliders, 8 boats, 5 loaves, 3 ships, 2 barges, 2 ponds and 1 mango



<http://www.conwaylife.com/wiki/Methuselah>

# Acorn

00 (0.00, 0.00, 5.00) 0 / 58088

+

<https://www.youtube.com/watch?v=U2dB57bwIWQ>  
<http://altsoph.com/projects/conwaytree/>

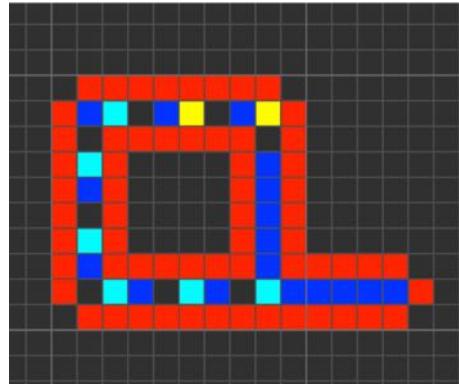
# Game of Life

- ◊ A computationally interesting cellular automata
  - ▷ Simple definition, complex behaviour
  - ▷ Unexpected emergent phenomena
  - ▷ i.e. spaceships, methuselah
  - ▷ Computationally universal
- ◊ Lots of Game of Life implementations:
  - ▷ <http://www.bitstorm.org/gameoflife/> [Java, online]
  - ▷ <http://golly.sourceforge.net> [cross-platform]
  - ▷ Do try this at home!

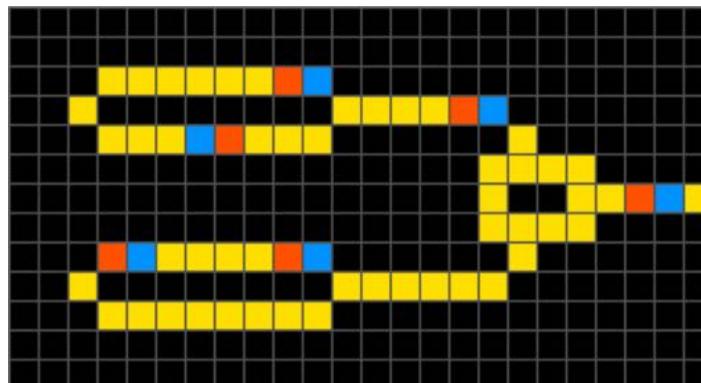
Any Questions?

# Multi-Valued States

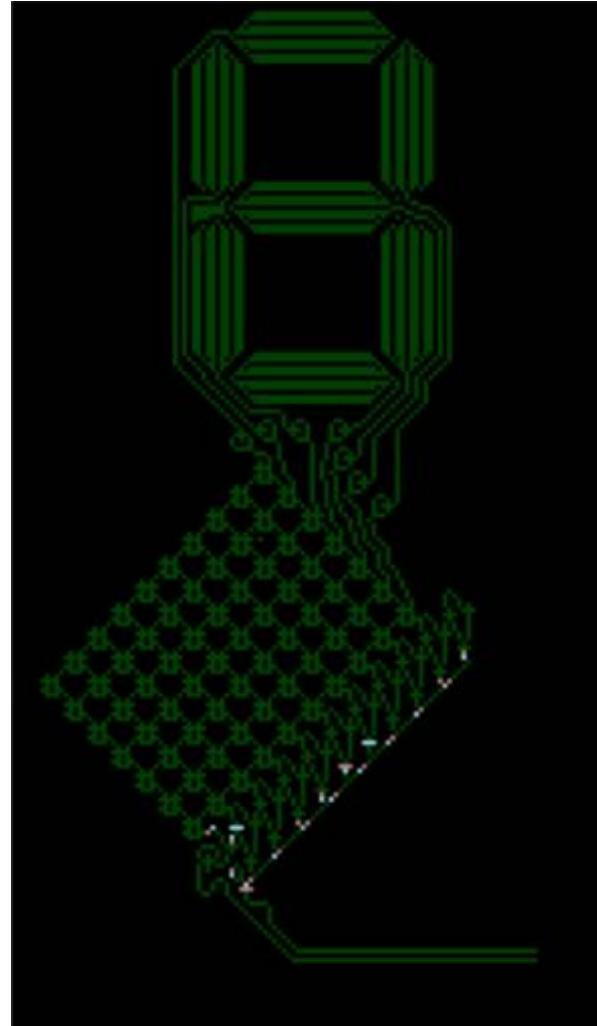
- ◊ Various multi-valued state CAs have been studied
  - ▷ e.g. Langton's loops model self-replication
    - Uses 8 states:



- ▷ e.g. WireWorld models electron flow in circuits
  - Uses 4 states:



# WireWorld



[http://en.wikipedia.org/wiki/File:Animated\\_display.gif](http://en.wikipedia.org/wiki/File:Animated_display.gif) (animated)

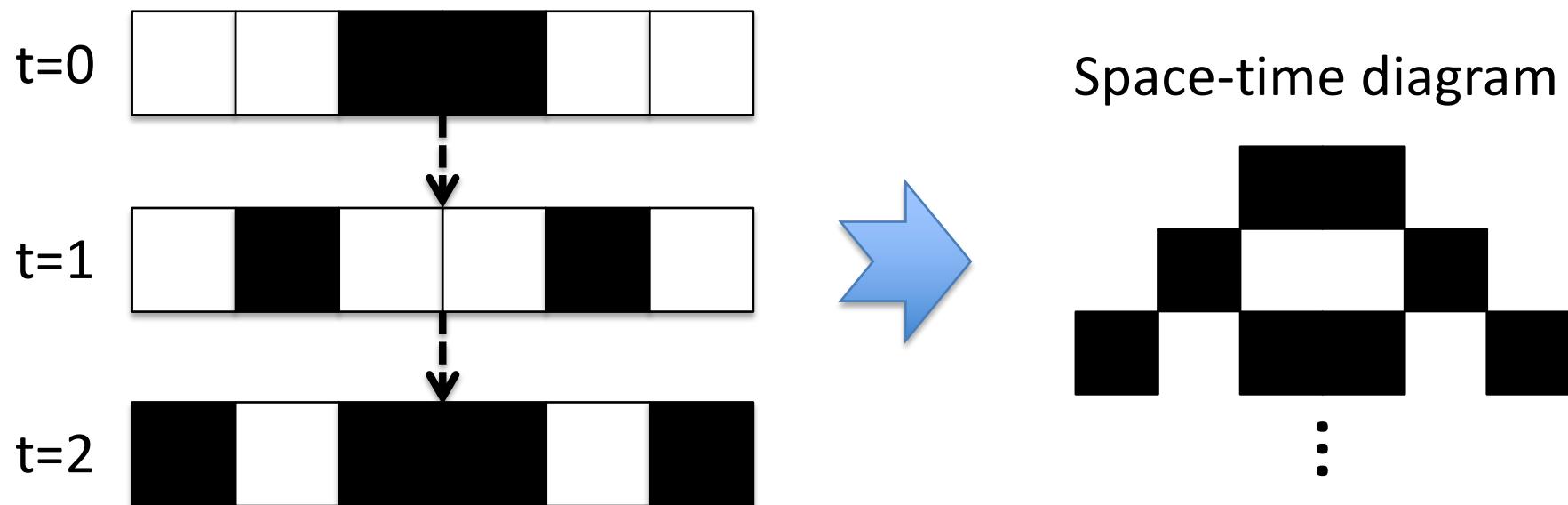
# Other Model Extensions

- ◊ Probabilistic cellular automata
  - Transitions occur with a certain probability
- ◊ Asynchronous cellular automata
  - Updates don't occur at the same time
- ◊ Use of non-rectangular grids
  - e.g. irregular Penrose tilings
- ◊ Continuous cellular automata
  - Continuous state, functions or spaces



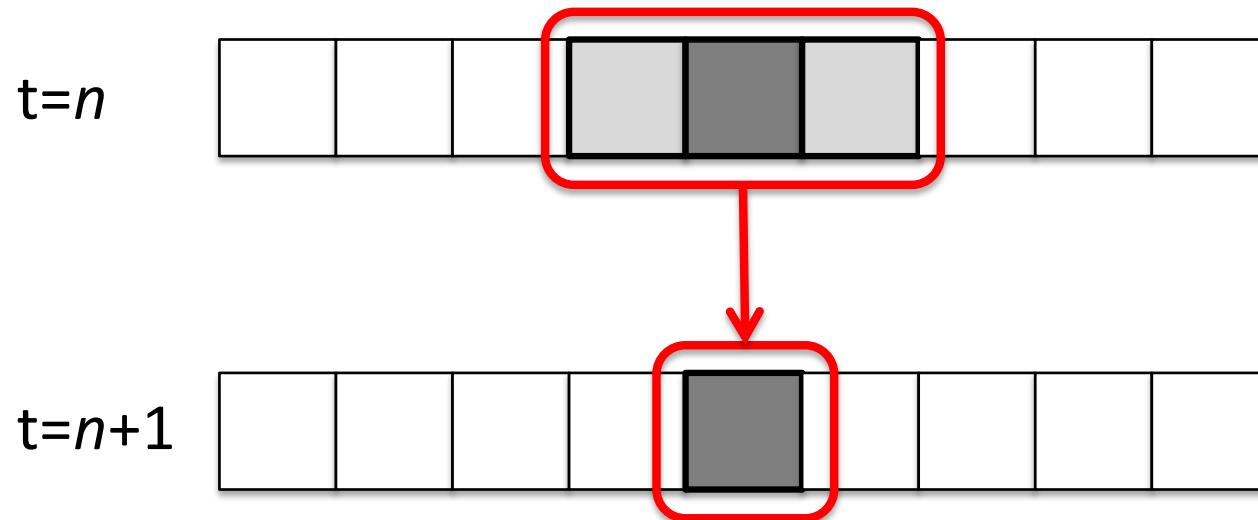
# Elementary Cellular Automata

- ◊ 1D binary CAs that take place on a single grid row
  - ▷ Appear simple, but can be deceptively complex
  - ▷ Probably the most studied form of CA
  - ▷ Stephen Wolfram's work on these is very well known



# Elementary Cellular Automata

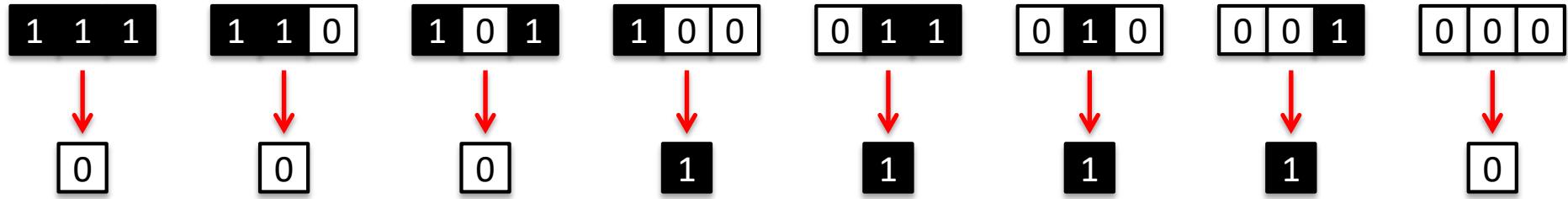
- ◊ They are based around a neighbourhood of size 3:



- ◊ Hence, it maps  $2^3=8$  possible patterns to 0 or 1
  - ▷ Meaning there are  $2^8=256$  possible update rules

# Elementary Cellular Automata

- ◊ An example of a rule:



- ◊ This is known as Rule 30 ... can you see why?
  - Hint: binary!
  - Every elementary CA has a number between 0 and 255

# Rule 30

- ◊ One of Wolfram's famous rules
  - ▷ Leads to complex, aperiodic, chaotic behaviour
  - ▷ Space-time diagram resembles the shell of *conus textile*
  - ▷ Used as a random number generator in Mathematica



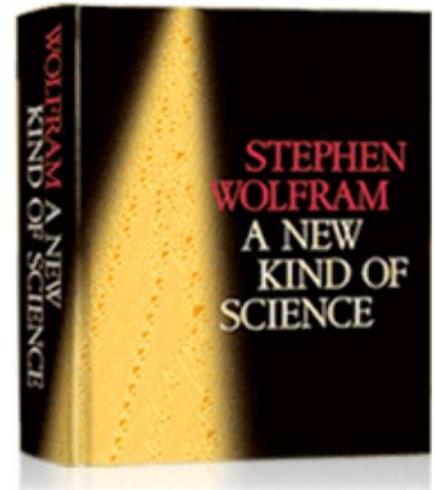
From A New Kind of Science, © Stephen Wolfram, LLC



[http://en.wikipedia.org/wiki/File:Textile\\_cone.JPG](http://en.wikipedia.org/wiki/File:Textile_cone.JPG)

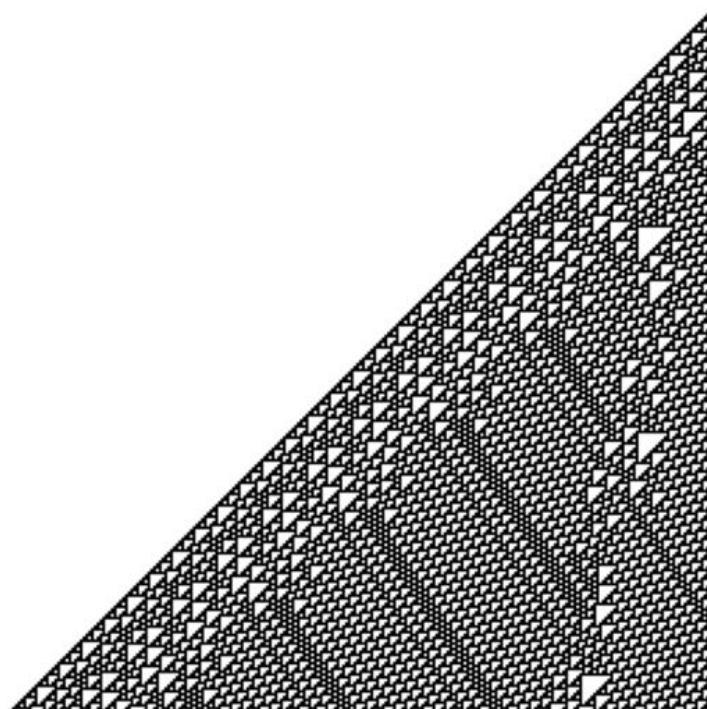
# “Simple Programs”

- ◊ Stephen Wolfram's book in a nutshell
  - Simple programs (such as CAs) can generate complex behaviours
  - They can generate a lot of the patterns we see in natural systems
  - They might therefore be used as a means for studying natural systems
  - Rather than using top-down models
  - e.g. in physics, chemistry, biology, ...
- ◊ Often misunderstood as saying the universe is a CA

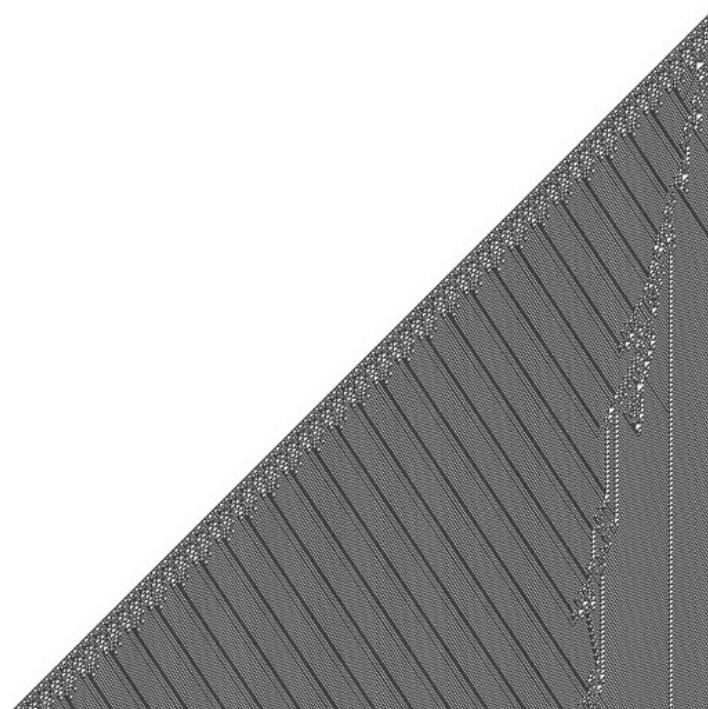


# Rule 110

- ◊ This one is known to be Turing complete
  - The simplest known Turing complete system
  - Very simple definition, complex behaviour
  - Behaviour appears to take place on the “edge of chaos”



From A New Kind of Science, © Stephen Wolfram, LLC

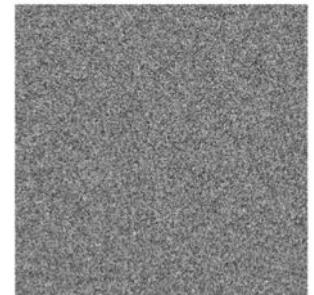


# Evolving Cellular Automata

- ◊ Rules 30 and 110 were discovered by exhaustively enumerating and simulating the rule space
  - Feasible for these elementary CAs
  - Quickly becomes infeasible for more complex CAs
- ◊ Is it possible to use EAs to find useful rules?
  - Yes, this has been done.
- ◊ Is this a form of genetic programming?
  - If the aim is to perform computation, then yes!
  - If the aim is modeling, then probably not

# Computing with CAs

- ◊ How do I compute using a CA?
  - ▷ 1) Find a suitable rule
  - ▷ 2) Encode the problem instance in the initial state
  - ▷ 3) Execute the CA for a certain number of steps
  - ▷ 4) Read the result from the final state of the CA
  
- ◊ Lots of cryptography applications
  - ▷ Google ‘cellular automata encryption’
  - ▷ Lots of different CA models used
  - ▷ Have a look!



# Things you should know

- ◊ Know how to execute a CA, and their uses
- ◊ Know about the Game of Life:
  - Know it is computationally universal, and understand why
  - Know its emergent phenomena: gliders, methuselah
  - *You don't need to remember patterns*
- ◊ Know about Elementary CAs:
  - Know what they are and why they are numbered
  - Know they are a simple model of universal computation
  - *You don't need to remember individual rules*

# Suggested Reading

- ◊ Have a look at Wolfram's "A New Kind of Science"
  - ▷ Book at  
<https://www.wolframscience.com/nksonline/toc.html>
  - ▷ Or his overview talk at  
<https://www.youtube.com/watch?v= eC14GonZnU>

# Things to try out

- ❖ Try out some CAs:
  - ▶ Forest fire  
<http://www.macs.hw.ac.uk/~dwcorne/mypages/apps/ca.html>
  - ▶ Elementary cellular automata  
<http://www.macs.hw.ac.uk/~dwcorne/mypages/apps/1dca.html>
  - ▶ Golly (game of life, Langton's loops, WireWorld)  
<http://golly.sourceforge.net/>
  
- ❖ Read about using CAs for modelling:
  - ▶ See papers on course website
  - ▶ Urban growth, traffic simulation, flu infection, brain tumour growth, HIV infection

# Next Lecture

- ◊ Models of how biological cells actually “compute”
  - ▷ Gene regulatory networks (GRNs)
  - ▷ Boolean networks
  - ▷ Artificial genome models

