

# CITS4403

## Computational Modelling

### Lecture 4: Cellular Automata I

**Dr. Siwen Luo**

Semester 2, 2024  
School of Computer Science,  
University of Western Australia



## Lecture 4: Cellular Automata I

1. Recap
2. Cellular Automata
  - 1-D CA
  - Implementation
3. Wolfram's Rules
4. Cellular Automata Classes

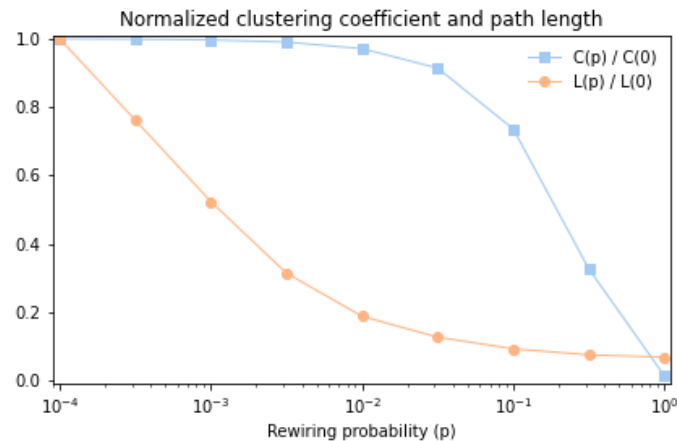
## Watts & Strogatz (WS) Graph:

$$G(n, k, p)$$

$n$ : the number of nodes

$k$ : degree of each node

$p$ : the probability to rewire the edge between a pair of nodes



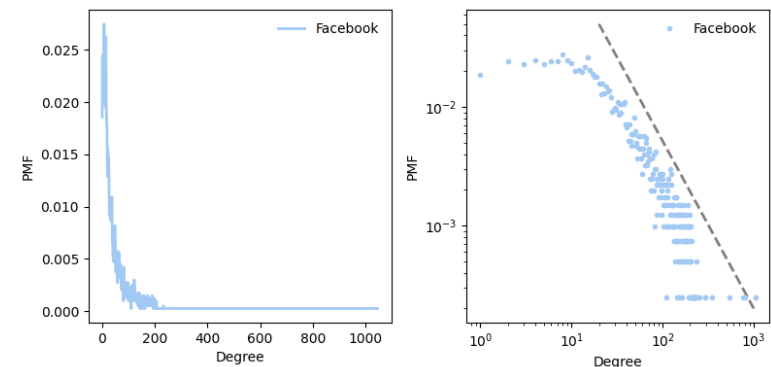
There is a wide range of  $p$  where a WS graph has the properties of a small world graph, high clustering and low path lengths.

A **scale-free network** is a network whose degree distribution follows a **power law**, at least asymptotically.



**Barabasi & Albert (BA) model:** generate random graphs with the scale-free property.

Different from WS Graph:

- **Growth** -- Instead of starting with a fixed number of vertices, the BA model starts with a small graph and adds nodes one at a time.
- **Preferential attachment** -- When a new edge is created, it is more likely to connect to a node that already has large number of edges.



## WS model vs. BA model

WS model	BA model
<ul style="list-style-type: none"><li>• High clustering</li><li>• Short path length</li><li>• Power law distribution </li></ul>	<ul style="list-style-type: none"><li>• Low clustering</li><li>• Short path length</li><li>• Power law distribution </li></ul>

## Lecture 4: Cellular Automata I

1. Recap
2. **Cellular Automata**
  - 1-D CA
  - Implementation
3. Wolfram's Rules
4. Cellular Automata Classes

## Cellular Automata

A **discrete** model of computation studied in automata theory.

**Automaton:** simple machine (mathematical abstraction or a computer simulation) designed to automatically follow a sequence of operations or respond to predetermined instructions.

## Cellular Automata

A **discrete** model of computation studied in automata theory.

**Automata:** simple machines (mathematical abstraction or a computer simulation) designed to automatically follow a sequence of operations or respond to predetermined instructions.

Cellular Automata is **discrete** in both ***space*** and ***time***:

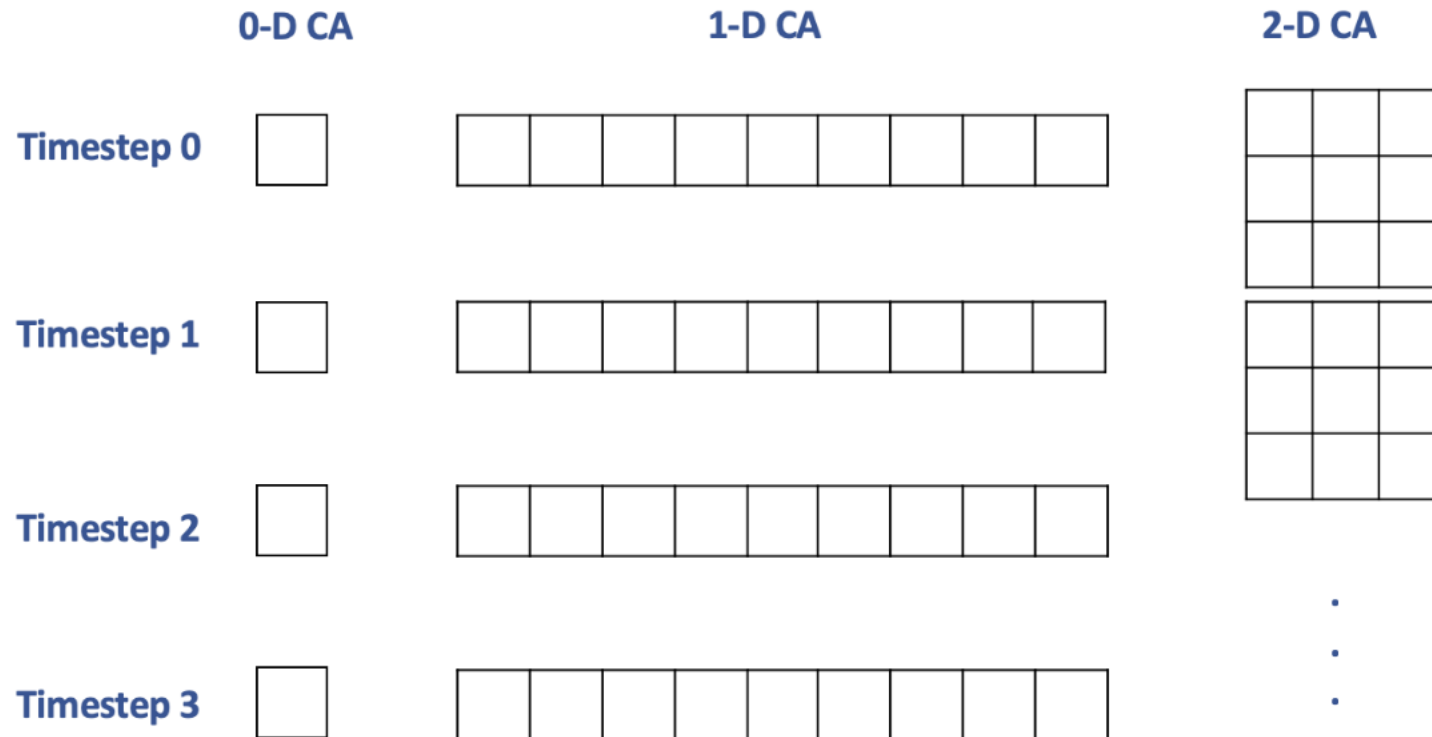
- Cellular: divide a space into small and simple discrete chunks – cells;
- All the cells evolve together in discrete time steps to undergo **transitions** based on some **rules**.



## A Cellular Automaton:

- consists of a grid of cells; the grid can be in any finite number of dimensions.
- each cell has a **state**, normally the number of possible states is finite.
- the simplest state form is usually **binary**: True/False, ON/OFF or 0/1.
- each cells have a neighbourhood: a set of cells that determine the next state of the given cell at next time step.
- evolves through discrete time steps based on rules that determine how the state of the cells changes over time.
- Typically, the rule does not change over time.

Cellular Automata come in different varieties with **different dimension**



Cellular Automaton evolves through discrete time steps, and the states of cells will be updated based on the fixed rule.

## 0-D CA

Timestep 0	1
Timestep 1	0
Timestep 2	1
Timestep 3	0

Cellular Automaton evolves through discrete time steps, and the states of cells will be updated based on the fixed rule.

## 1-D CA

Timestep 0	0	0	0	0	1	0	0	0	0
Timestep 1	0	0	0	1	1	1	0	0	0
Timestep 2	0	0	1	0	1	0	1	0	0
Timestep 3	0	1	1	0	1	0	1	1	0

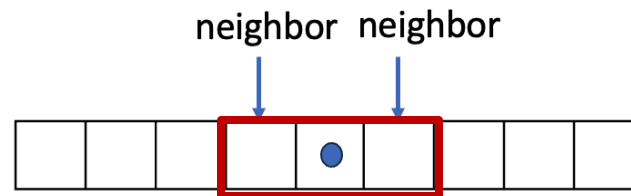
***Check the implementation of 1D CA based on  
simple rules in python***

## Lecture 4: Cellular Automata I

1. Recap
2. Cellular Automata
  - 1-D CA
  - Implementation
3. **Wolfram's Rules**
4. Cellular Automata Classes

In the early 1980s Stephen Wolfram published a series of papers presenting a systematic study of 1-D CAs.

- A finite sequence of cells arranged in a lattice, where each cell is connected to 2 neighbours.



- The rules that determine how the system evolves in time are based on the notion of a “neighbourhood”, which is the set of cells that determines the next state of a given cell.
- Wolfram’s experiments use a **3-cell neighbourhood**: the cell itself and its two neighbours.
- the cells have two states, denoted 0 and 1 or “off” and “on”

## Wolfram's 3-cell neighborhood

The state of each cell in next generation (time step) based on the value of:

- The cell to its left
- The cell itself
- The cell to its right

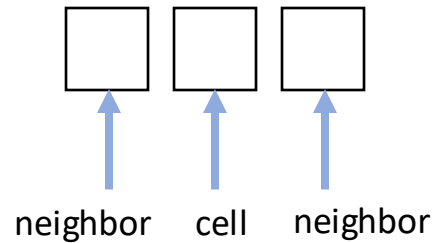
at the current step.

***How many possible neighborhood states combinations can be?***

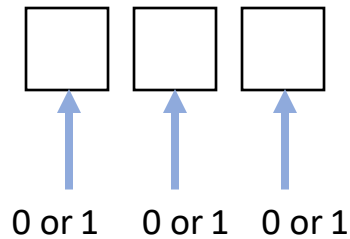




*How many possible neighborhood states combinations can be?*

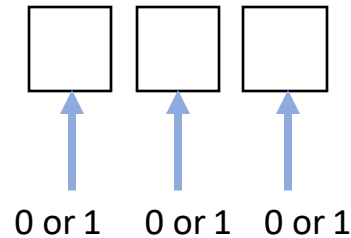


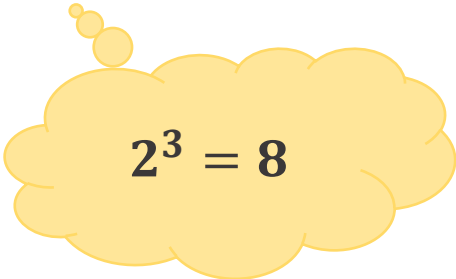
*How many possible neighborhood states combinations can be?*



*Each cell has two states: 0 and 1*

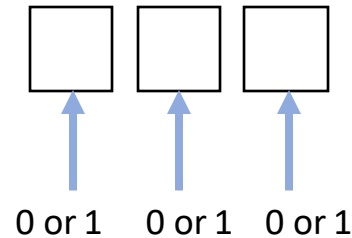
*How many possible neighborhood states combinations can be?*




$$2^3 = 8$$

*Each cell has two states: 0 and 1*

*How many possible neighborhood states combinations can be?*



$$2^3 = 8$$

*Each cell has two states: 0 and 1*

1	1	1
---	---	---

1	1	0
---	---	---

1	0	1
---	---	---

1	0	0
---	---	---

0	1	1
---	---	---

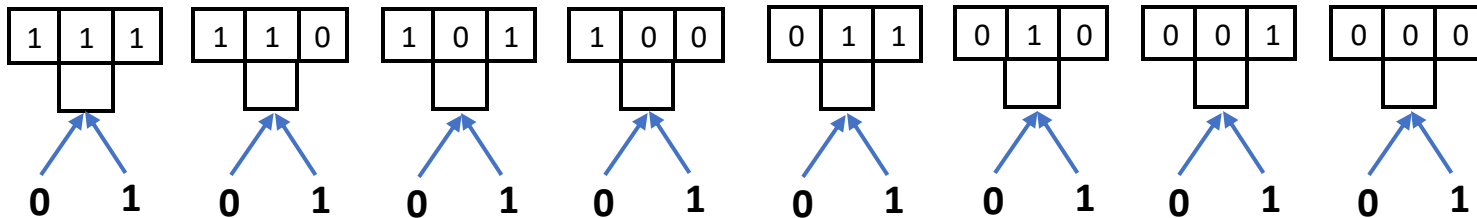
0	1	0
---	---	---

0	0	1
---	---	---

0	0	0
---	---	---

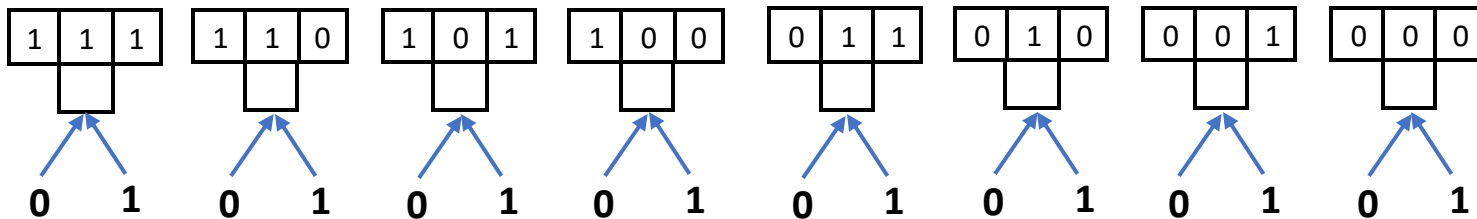
## Wolfram's 3-cell neighborhood

A rule can be summarized that maps from the state of the neighborhood (a tuple of three states) to the next state of the center cell.



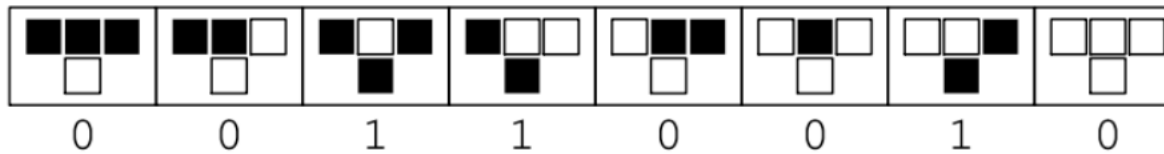
## Wolfram's 3-cell neighborhood

A rule can be summarized that maps from the state of the neighborhood (a tuple of three states) to the next state of the center cell.



- The top line shows the current states of the left neighbor, the cell itself and the right neighbor (a 3-tuple).
- Below is the new state updated based on the previous state of itself and its two neighbors in the previous generation.

## Wolfram's *Rule 50*



### *Why call this Rule 50?*

Wolfram suggested reading the bottom row as a binary number, and 00110010 in binary is 50 in decimal.

## Binary to Decimal

*How to know 00110010 is binary for 50?*

0	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
-------	-------	-------	-------	-------	-------	-------	-------



## Binary to Decimal

*How to know 00110010 is binary for 50?*

0	0	1	1	0	0	1	0
x	x	x	x	x	x	x	x
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

## Binary to Decimal

*How to know 00110010 is binary for 50?*

0	0	1	1	0	0	1	0
x	x	x	x	x	x	x	x
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
=							
0	0	32	16	0	0	2	0

## Binary to Decimal

*How to know 00110010 is binary for 50?*

0	0	1	1	0	0	1	0
<b>x</b>	<b>x</b>	<b>x</b>	<b>x</b>	<b>x</b>	<b>x</b>	<b>x</b>	<b>x</b>
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

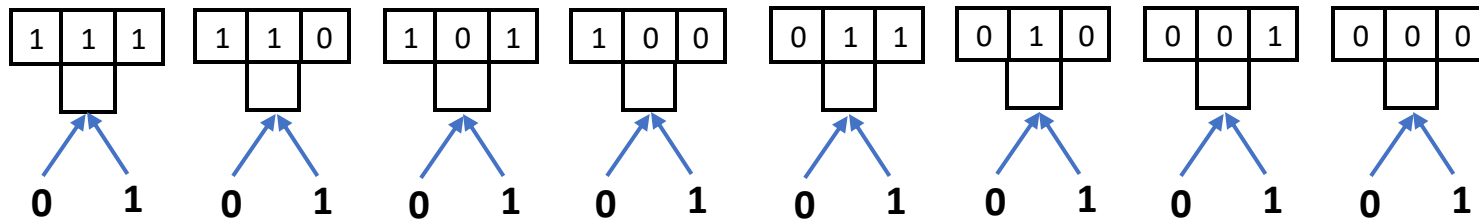
**=**

0	<b>+</b>	0	<b>+</b>	32	<b>+</b>	16	<b>+</b>	0	<b>+</b>	0	<b>+</b>	2	<b>+</b>	0
---	----------	---	----------	----	----------	----	----------	---	----------	---	----------	---	----------	---

**= 50**

## Total number of Rules

*In Wolfram's 3-cell neighborhood, we can specify  $2^8 = 256$  rules.*



*For initial conditions of a single black cell, rule 50 is equivalent to rules 58, 114, 122, 178, 186, 242, and 250, which are precisely those rules with binary representation xx11x010.*

current automaton contents



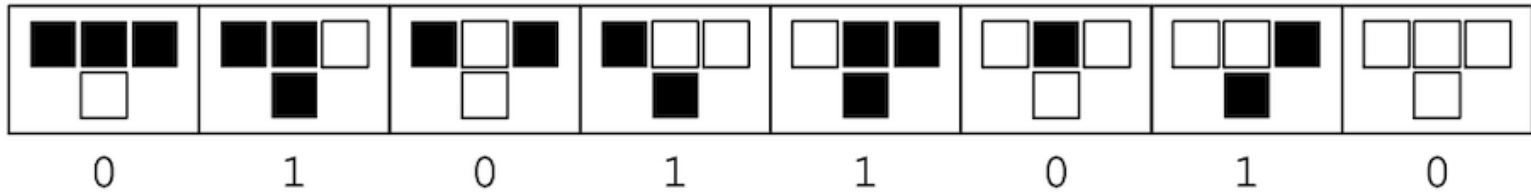
rule 30 (00011110)

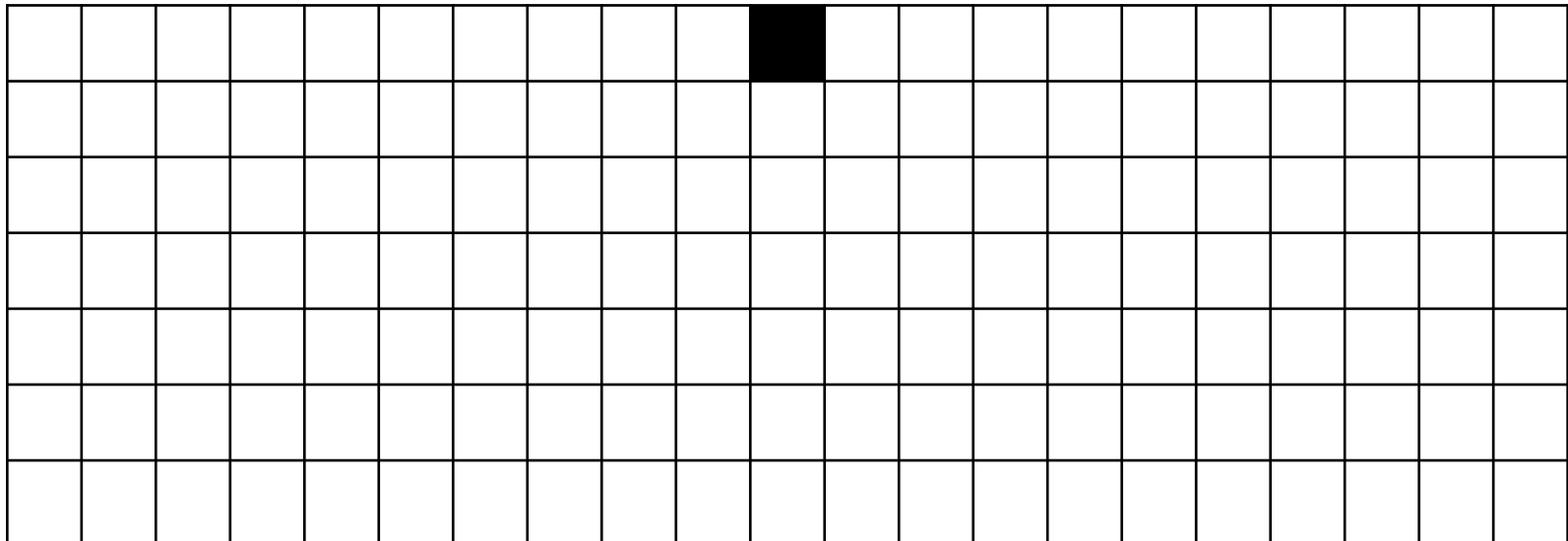
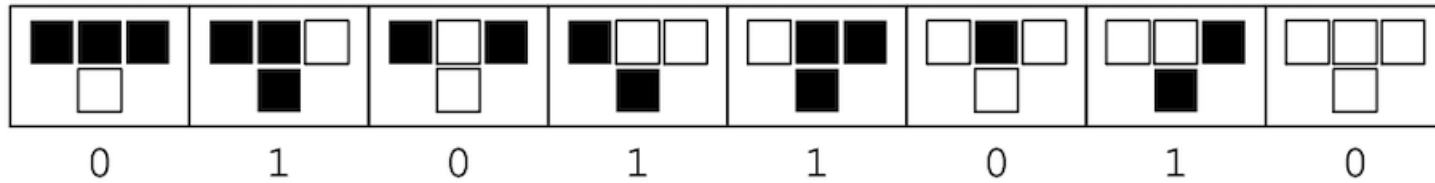


the next generation of the automaton



*Let's try with some drawing for Rule 90!*

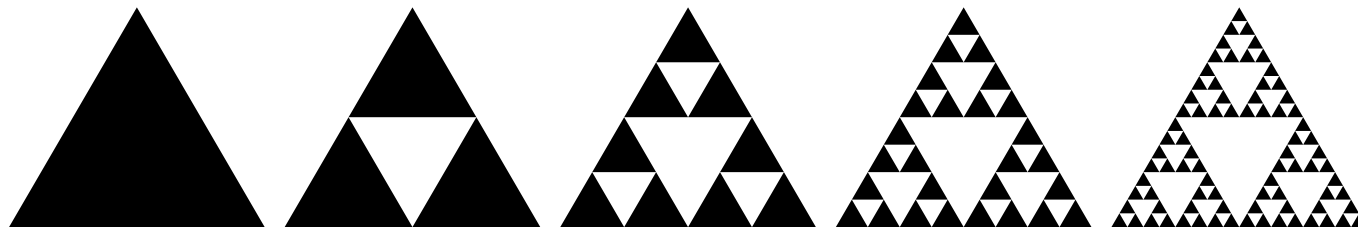




## The Sierpinski triangle

The Sierpinski triangle may be constructed from an equilateral triangle by repeated removal of triangular subsets:

- Start with an equilateral triangle.
- Subdivide it into four smaller congruent equilateral triangles and remove the central triangle.
- Repeat step 2 with each of the remaining smaller triangles infinitely.





## Lecture 4: Cellular Automata I

1. Recap
2. Cellular Automata
  - 1-D CA
  - Implementation
3. Wolfram's Rules
4. **Cellular Automata Classes**

## ***Class 1***

Nearly all initial patterns evolve quickly into a stable, homogeneous state.

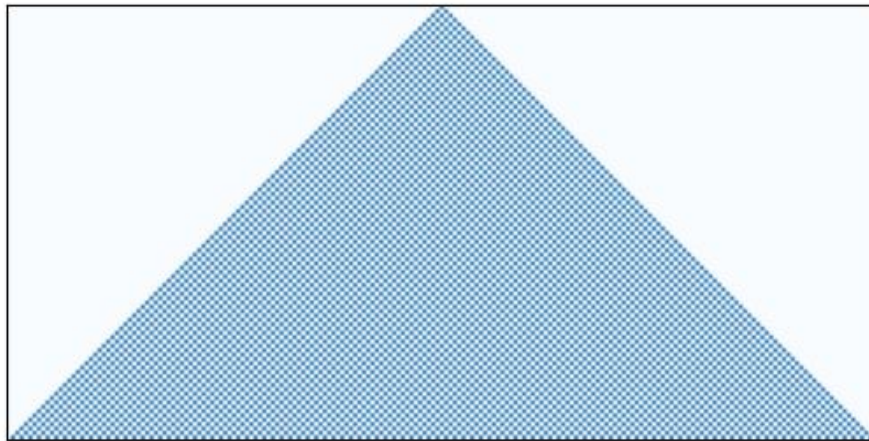
Class 1 contains the simplest (and least interesting) CAs, the ones that evolve from almost any starting condition to the same uniform pattern.



Rule 4, after 100 steps

## Class 2

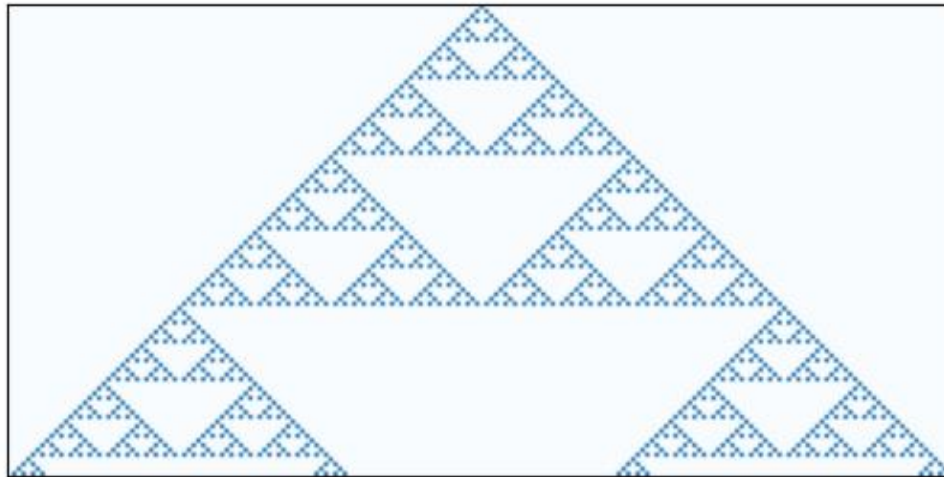
Generates simple pattern with intricate nested repeats – stable repetition and a fractal nature



Rule 50, after 100 steps

## Class 2

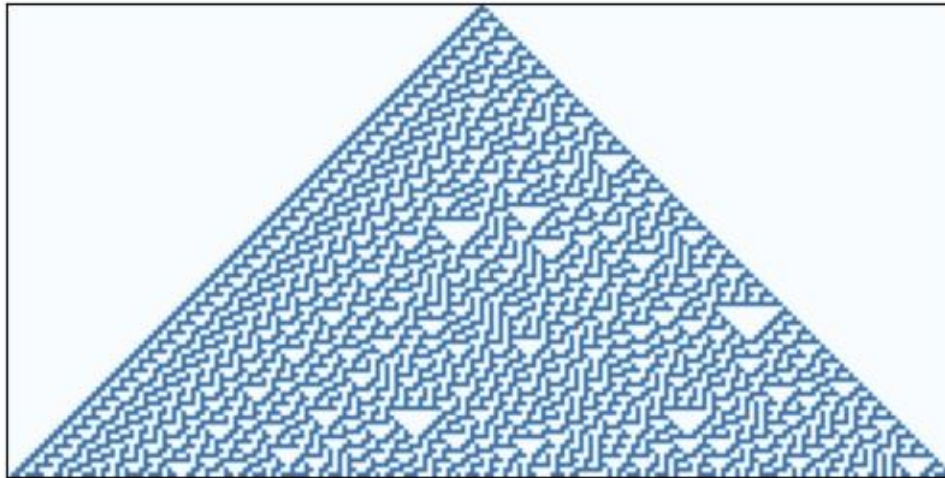
Generates simple pattern with intricate nested repeats – stable repetition and a fractal nature



Rule 90, after 100 steps

### ***Class 3***

Nearly all initial patterns evolve in a pseudo-random or chaotic manner. Any stable structures that appear are quickly destroyed by the surrounding noise.



Rule 30, after 100 steps

## ***Pseudo-Random Number Generators (PRNGs)***

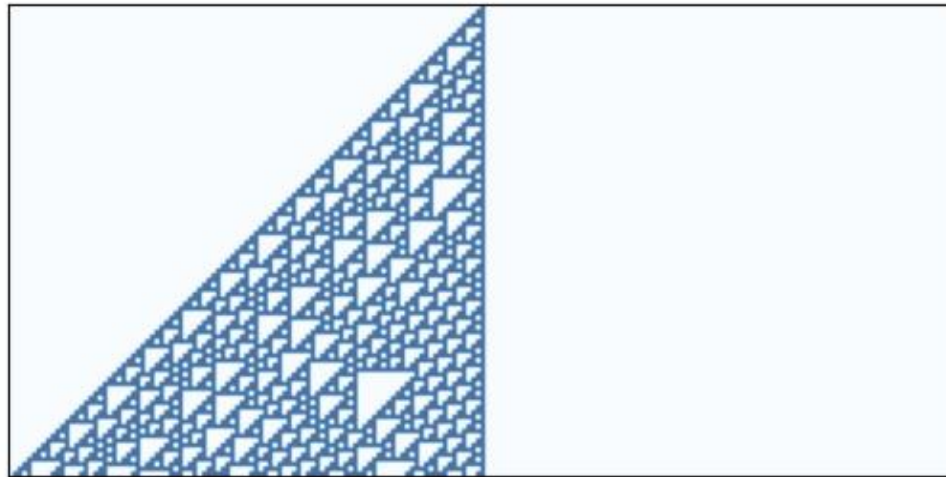
PRNGs are not considered truly random:

- Many of them produce sequences with regularities that can be detected statistically.
- Any PRNG that uses a finite amount of state will eventually repeat itself.
- The underlying process is fundamentally deterministic, unlike some physical processes, like radioactive decay and thermal noise, that are fundamentally random.

Modern PRNGs produce sequences that are statistically indistinguishable from random, and they can be implemented with periods so long that the universe will collapse before they repeat. Wolfram argues that no difference between a good PRNG (with a very long period before repetition) and real randomness

## Class 4

Nearly all initial patterns evolve into structures that interact in complex and interesting ways, with the formation of local structures that can survive (remain repetitive or stable) for long periods of time



Rule 110, after 100 steps

## ***Class 4***

Nearly all initial patterns evolve into structures that interact in complex and interesting ways, with the formation of local structures that can survive (remain repetitive or stable) for long periods of time



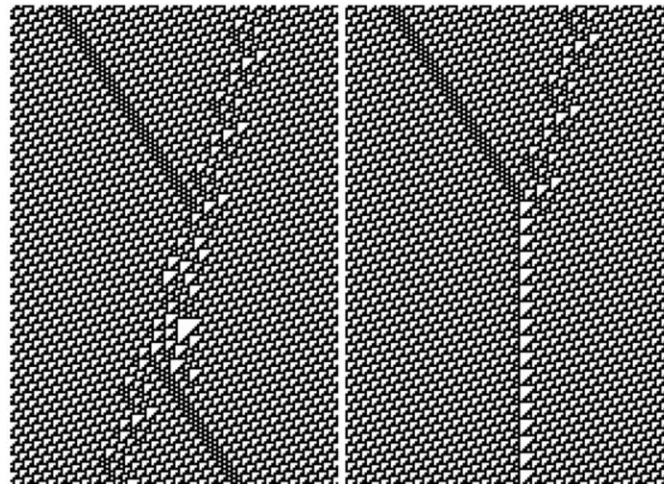
Rule 110, after 600 steps



## Class 4

Nearly all initial patterns evolve into structures that interact in complex and interesting ways, with the formation of local structures that can survive (remain repetitive or stable) for long periods of time

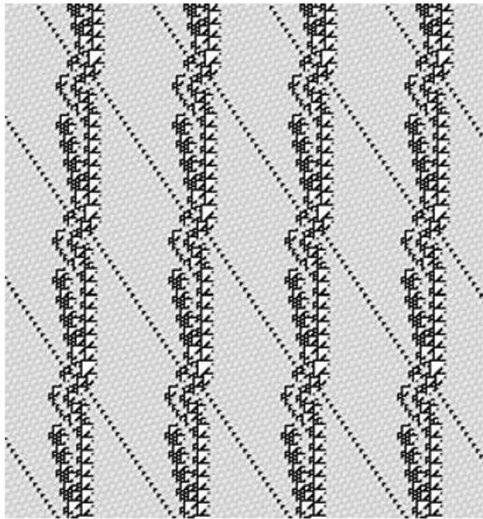
*Below is an image showing the first two structures passing through each other without interacting other than by translation (left) and interacting to form the third structure (right).*



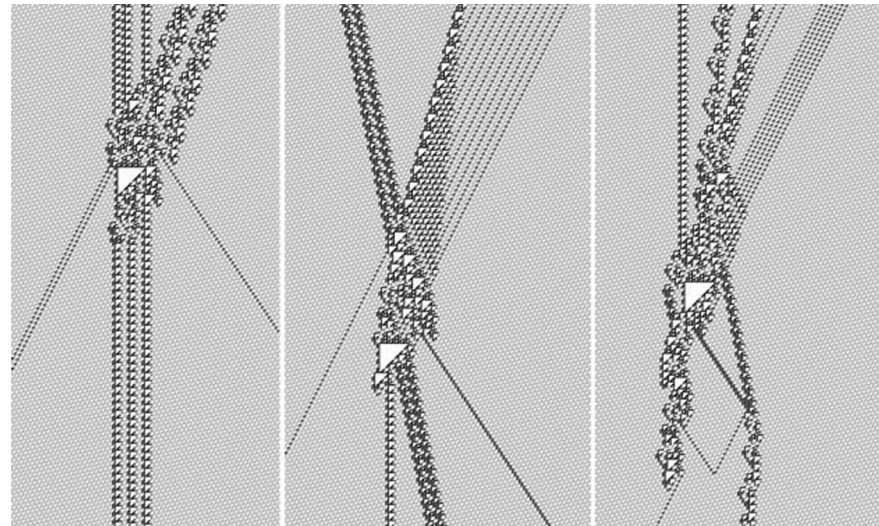
*If we think of gliders as signals that propagate through space, and collisions as gates that compute logical operations like AND and OR, we can see what it means for a CA to perform a computation.*

## Class 4

Nearly all initial patterns evolve into structures that interact in complex and interesting ways, with the formation of local structures that can survive (remain repetitive or stable) for long periods of time



Meta-glider in Rule 110



Collision-based Triangle in Rule 110

*Collisions between gliders yield different results depending on the types of the spaceships and the phase they are in when they collide.*

Cellular Automata are abstract (not detailed or realistic) models of physical systems, the elements of the model are not realistically corresponded to the elements of a physical system.

E.g. some species of cone snail produce a pattern on their **shells that resembles the patterns generated by cellular automata**.

It is not clear how the *cells, interaction between neighbours, rules* of the CA correspond to the *real cells, chemical signals, protein interaction networks* of a growing snail.



Simple, abstract models offer a different kind of explanation than detailed models.

*If an observation of a physical system resembles the behaviours exhibited by a simple model, this shows the physical system have the set of features sufficient to produce those behaviours.*

## Lecture 5: Cellular Automata II

- 2D Cellular Automata
- Game of Life
- W6 in-lecture test information

