Cellular Automata Model

1 Introduction

Cellular Automata (CA) Model is a type of mathematical model used to simulate complex systems and processes in a discrete space and time framework.

2 Elements

The basic elements of CA model include:

2.1 Cells

The basic units of a cellular automaton are cells. Typically, these are arranged in a grid, which can be 1D (a line), 2D (like a chessboard), 3D, or even higher dimensions. Each cell has a state, which can change over discrete time steps.

2.2 Neighborhood

Each cell in the grid has a set of neighboring cells. The most common neighborhoods in 2D are the **Moore neighborhood** (which includes all 8 surrounding cells) and the **Von Neumann neighborhood** (which includes only the 4 direct neighbors: up, down, left, and right).

2.3 Rules

The state of a cell in the next time step is determined by a set of rules that take into account the current state of the cell and the states of its neighbors. These rules are applied simultaneously to all cells in the grid at each time step.

2.4 Initialization

The system starts from an initial configuration. This might be random, a single seed cell, or some predefined pattern, depending on the context or the problem being addressed.

2.5 Boundary Condition

The boundary condition of a cellular automata (CA) model defines how the edges of the grid are treated. Since a CA model consists of a grid of cells that evolve based on rules considering the states of neighboring cells, what happens at the edges of the grid (where some neighbors are missing) needs to be specified. There are several common types of boundary conditions:

2.5.1 Periodic Boundary Conditions (Wrap-around)

The grid is considered to wrap around on itself like a torus. The cells on the left edge are neighbors with the cells on the right edge, and the cells on the top edge are neighbors with the cells on the bottom edge.

2.5.2 Fixed Boundary Conditions

The edge cells are influenced by fixed values, often set to a particular state like 0 or 1. The cells outside the boundary are considered to be in this fixed state.

2.5.3 Reflective (or Neumann) Boundary Conditions

The edge cells have their missing neighbors reflected back into the grid. This means that the neighbors of an edge cell are the edge cell itself.

2.6 Evolution

Over discrete time steps, the cellular automaton evolves. This means that, at every time step, the rules are applied to each cell, determining their next states based on their current states and those of their neighbors.

3 Infectious Disease Spread Case Study

3.1 Case Setup

Similar with the Markov Chain model, we have the following parameters:

1. k: Duration of infection. In this case, k = 2.
2. t: Number of days
3. τ: Infection probability
4. Node label 0: Susceptible
5. Node label -1: Recovered
6. Node label 1 and 2: Infection date 1 and 2.

3.2 Transition

At day 0, one patient is infected, every other patients are susceptible. The N-S-E-W neighbors are exposed.

Figure 1: Day 0.

At day 1, the illness propagates randomly to the exposed and susceptible neighbors, and spreads to other neighbors:

Figure 2: Day 1.

At day 2, the first case recovers, but the infection continues to spread:

Figure 3: Day 2.

4 Conway's Game of Life

Conway's Game of Life is a cellular automaton devised by mathematician John Horton Conway in 1970. It is a zero-player game, meaning its evolution is determined by its initial state, requiring no further input. The game consists of a grid of cells, each of which can be in one of two states: alive or dead. The evolution of the grid follows these simple rules:

1. **Survival:** A living cell with two or three living neighbors stay alive
2. **Death by Isolation:** A living cell with fewer than two living neighbors dies.
3. **Death by Overcrowding:** A living cell with more than three living neighbors dies.
4. **Birth:** A dead cell with exactly three living neighbors becomes alive.

Some classical evolving patterns include:

- **Stable Pattern:** These patterns do not change over time. Once they reach a stable configuration, they remain unchanged through successive generations.
- **Oscillatory Pattern:** These patterns cycle through a fixed sequence of states. They return to their initial configuration after a certain number of generations, known as the period of the oscillator.
- **Motion:** These patterns, also known as "spaceships," travel across the grid over time. They periodically return to their initial configuration but in a different location.

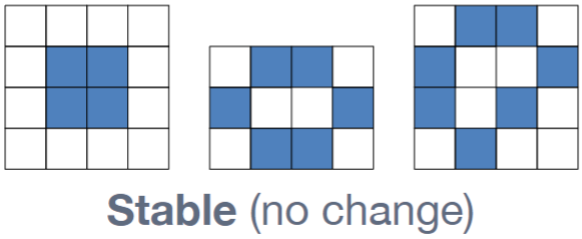

Figure 4: Stable Pattern.

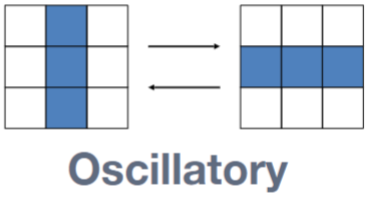

Figure 5: Oscillatory Pattern.

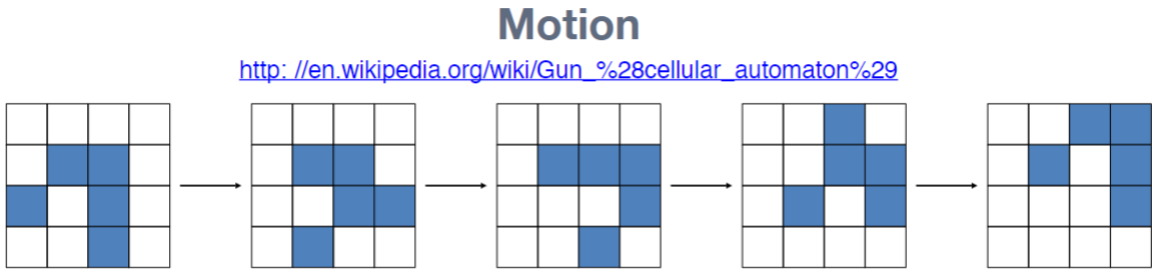

Figure 6: Motion Pattern.

5 Wolfram Elementary Cellular Automata

5.1 Definition

Wolfram's 1-D nearest neighbor patterns refer to a specific type of cellular automaton rule set, known as elementary cellular automata. These are one-dimensional arrays of cells, each of which can be in one of two possible states (often represented as 0 or 1). The state of each cell in the array at the next time step is determined by its current state and the state of its immediate neighbors to the left and right. The basic components of this problem include:

1. **Three-cell neighborhood:** Each cell has a left neighbor, itself, and a right neighbor. This makes for $2^3 = 8$ possible configurations of states for these three cells.

2. **Rule table**: Each of the 8 configurations is mapped to a result (0 or 1), creating a rule table. Notice that the corner values will follow the boundary conditions.

3. **Binary to decimal conversion:** The rule table can be represented as an 8-bit binary number, which can then be converted to a decimal number between 0 and 255. This decimal number is called the rule number. For example, rule 90 is shown below:

"**Rule 90**" (90_{10} == 01011010_2)

In	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 (0)
Out	0	1	0	1	1	0	1	0

Figure 7: Rule 90.

5.2 Rule Table

Different rule tables will create different patterns:

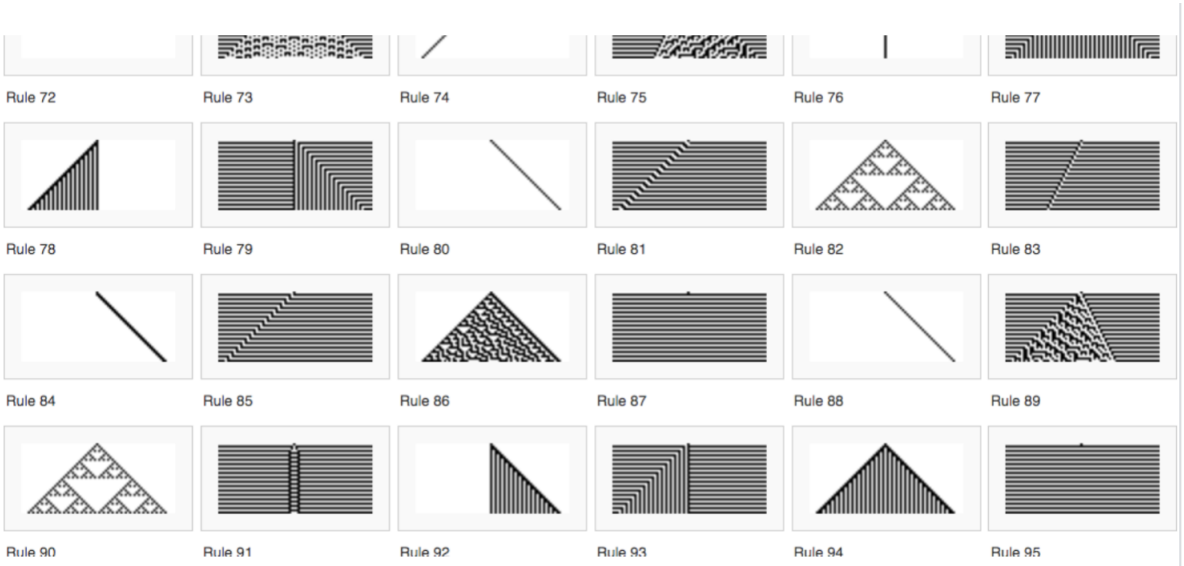

Figure 8: Rules Overview

In general, there are 4 classes of rules:

1. **Fixed Point:** Most initial states go to all ones or zeros.

2. **Periodic:** Most initial states go to stable or oscillatory states.

3. **Chaotic:** Most initial states lead to pseudorandom states.

4. **Complex:** Random yet ordered.

6 Cellular Automata Calculation

6.1 General Rules

- D: Dimensions
- L: Number of cells in one dimensional
- k: States per cell
- r: Radius of interaction
- n: Size of neighborhood, $n = (2r + 1)^D$
- p: Number of input states, $p = k^n$
- q: Number of rules, $q = k^p$

For example, for the elementary CA problem, which is 1D, binary state, 3-cell neighborhood:

- $D = 1$
- $k = 2$
- $r = 1$
- $n = (2 * 1 + 1)^1 = 3$
- $p = k^n = 2^3 = 8$
- $q = k^p = 2^8 = 256$

6.2 Elementary Rule 90 'on' Probability Calculation

	111	110	101	100	011	010	001	000
Recall: Rule 90 =	0	1	0	1	1	0	1	0
Let: $p \equiv$ initial prob.	(c)	(d)	(a)	(b)	(d)	(c)	(b)	(a)

	Initial	Next	Prob?
a)	0	0	$(1-p)^3 + (1-p)p^2 \equiv a(p)$
b)	0	1	$2(1-p)^2 p \equiv b(p)$
c)	1	0	$p[p^2 + (1-p)^2] \equiv c(p)$
d)	1	1	$p^2(1-p) + (1-p)p^2 \equiv d(p)$

At time $t+1$, p_{t+1} = prob "on" $= b(p_t) + d(p_t)$

[31] $b(p) + d(p) = 2p(1-p)^2 + 2p^2(1-p) = 2p(1-p)[1-p+1] = 2p(1-p)$

Figure 9: Rule90 Probability Calculation

7 Data Analysis

Because there are so many possible results coming from this model, the better way to is using **Monte Carlo** simulation to conduct large amount of simulations to find the probabilities.

The basic processes of a Monte Carlo Simulation include:

1. **Define a Model:** First, you must have a model of the system you want to analyze. This could be anything from a financial model predicting the future value of an investment, to a physical model predicting the behavior of a natural system.

2. **Identity Inputs:** Determine the uncertain factors or inputs in the model. For **each uncertain factor**, you assign a **probability distribution** based on your current knowledge, historical data, or expert judgment.

3. **Random Sampling:** For each uncertain factor, draw a random sample from its probability distribution. This sample represents a possible scenario or state of the world.

4. **Run the Model and Repeat:** With the set of random samples for each of your uncertain factors, run your model to compute an outcome and repeat it millions of times.

8 Useful Videos

1. Wolfram Elementary Cellular Automata