Conway's Game of Life

July 2023

By Mohammad Yousefiyan & Laya Amirian

Prof. Larijani

Check out MohammadYSF.github.io/ConwayGameOfLife to see the some visualization of this concept .
And you can find the source code here

Introduction

The **Game of Life**, also known simply as **Life**, is a cellular automaton devised by the British mathematician John Horton Conway in 1970.

The "game" is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves or, for advanced players, by creating patterns with particular properties.

One of the applications of this game is to analyze the complex systems ; systems in which its particles behavior is dependent on the behavior of other particles (in the game of life , that is neighbors)

What Is a Cellular Automaton?

A cellular automaton is a model of a system of "cell" objects with the following characteristics.

- The cells live on a ***grid*** (this grid can be in any finite dimensions).
- Each cell has a ***state***. The number of state possibilities is typically finite. The simplest example has the two possibilities of 1 and 0 (otherwise referred to as "on" and "off" or "alive" and "dead").
- Each cell has a ***neighborhood***. This can be defined in any number of ways, but it is typically a list of adjacent cells.

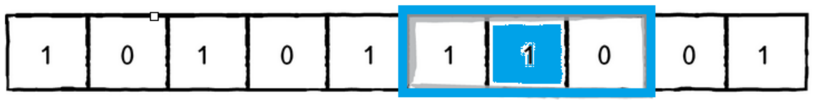
One Dimensional Cellular Automaton

1) ***Grid***. The simplest grid would be one-dimensional: a line of cells.

2) ***States***. The simplest set of states (beyond having only one state) would be two states: 0 or 1.

3) ***Neighborhood***. The simplest neighborhood in one dimension for any given cell would be the cell itself and its two adjacent neighbors: one to the left and one to the right.

The figures above show us the CA at time equals 0 or generation 0

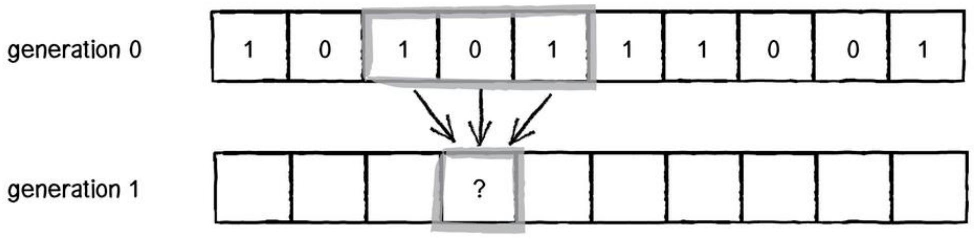


The most important detail of how cellular automata work is time. We're not really talking about real-world time here, but about the CA living over a period of time, which could also be called a generation.

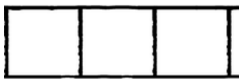
The questions we have to ask ourselves are: How do we compute the states for all cells at generation 1? And generation 2? And so on and so forth.

CELL state at time t = f(CELL neighborhood at time t - 1)

We calculate a new state value by looking at all the previous neighbor states.



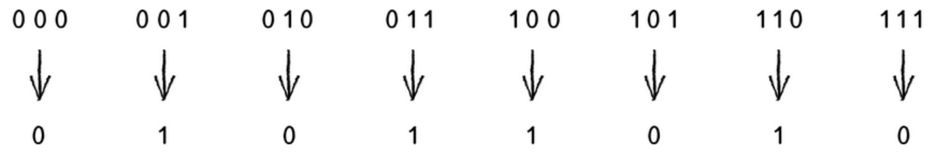
Consider the following neighborhood

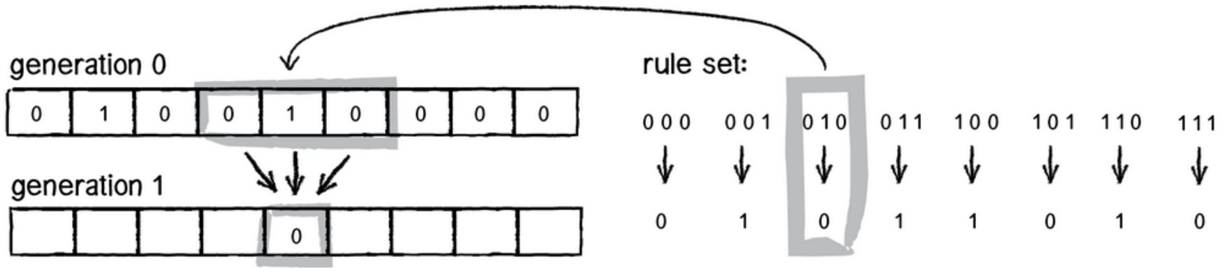


There are 8 possible set of states for this neighborhood

0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1

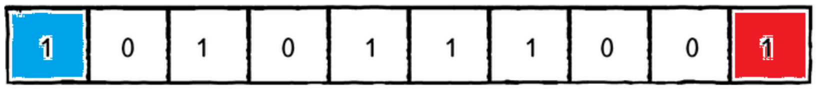
Once we have defined all the possible neighborhoods, we need to define an outcome (new state value: 0 or 1) for each neighborhood configuration.





As you can see , for each neighborhood , we reach to 8 bit binary number . We call this number a **RuleSet.** So the number of all possible rule sets is 2^8 = 256

A question you might ask it that what about the edges of the grid? It's up to you . One answer is that the edge cells has only one neighbor (instead of two) . The other (and the better) answer is that you think of the grid as a ring . See the figure below . In this approach , red and blue cells are neighbors of each other;while in the first approach , they are not.



you can see a simple graphical visualization here

Conway's Game of Life and Two Dimensional Automaton

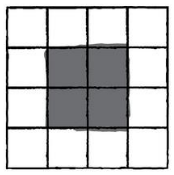
First, instead of a line of cells, we now have a two-dimensional matrix of cells. As with the elementary CA, the possible states are 0 or 1. Only in this case, since we're talking about "life," 0 means dead and 1 means alive.

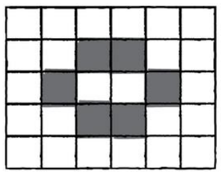
The cell's neighborhood has also expanded. If a neighbor is an adjacent cell, a neighborhood is now nine cells instead of three.

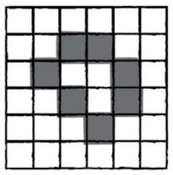
With three cells, we had a 3-bit number or eight possible configurations. With nine cells, we have 9 bits, or 512 possible neighborhoods. In most cases, it would be impractical to define an outcome for every single possibility. The Game of Life gets around this problem by defining a set of rules according to general characteristics of the neighborhood. In other words, is the neighborhood overpopulated with life? Surrounded by death? Or just right? Here are the rules of life.

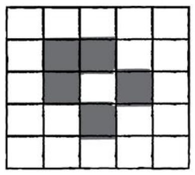
1. ***Death.*** If a cell is alive (state = 1) it will die (state becomes 0) under the following circumstances.
   - ***Overpopulation:*** If the cell has four or more alive neighbors, it dies.
   - ***Loneliness:*** If the cell has one or fewer alive neighbors, it dies.
2. ***Birth.*** If a cell is dead (state = 0) it will come to life (state becomes 1) if it has exactly three alive neighbors (no more, no less).
3. ***Stasis.*** In all other cases, the cell state does not change. To be thorough, let's describe those scenarios.
   - ***Staying Alive:*** If a cell is alive and has exactly two or three live neighbors, it stays alive.
   - ***Staying Dead:*** If a cell is dead and has anything other than three live neighbors, it stays dead.

One of the exciting aspects of the Game of Life is that there are initial patterns that yield intriguing results. For example, some remain static and never change.

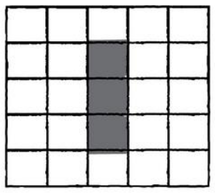

block

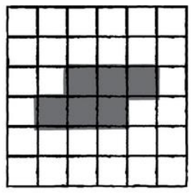

beehive

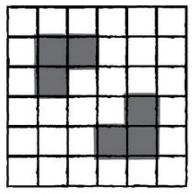

loaf


boat

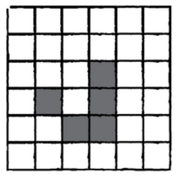
There are patterns that oscillate back and forth between two states.

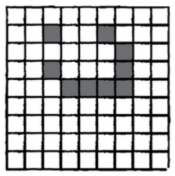

blinker


toad


beacon

And there are also patterns that from generation to generation move about the grid. (It's important to note that the cells themselves aren't actually moving, although we see the appearance of motion in the result as the cells turn on and off.)


glider


lightweight spaceship

you can play this game here

Undecidability

Many patterns in the game of life eventually become a combination of still lives, oscillators and spaceships; other patterns may be called chaotic. A pattern may stay chaotic for a very long time until it eventually settles to such a combination.

It can be asked whether the game of life is decidable: whether an algorithm exists, so that given an "initial" pattern and a "later" pattern, the algorithm can tell whether, starting with the initial pattern, the later pattern is ever going to appear. This turns out impossible: no such algorithm exists. This is in fact a corollary of the halting problem.

Indeed, since the game of life includes a pattern that is equivalent to a UTM (universal Turing machine), this "deciding" algorithm, if existed, could have been used to solve the halting problem, by taking the initial pattern as the one corresponding to a UTM+input and the later pattern as the one corresponding to a halting state of the machine with an empty tape (as one can modify the Turing machine to always erase the tape before halting). However the halting problem is provably undecidable and so such an algorithm does not exist.

It also follows that some patterns exist that remain chaotic forever: otherwise one could just progress the game of life sequentially until a non-chaotic pattern emerges, and then easily compute whether the later pattern is going to appear.

Resources

- https://conwaylife.com/
- The coding train youtube channel
- https://gambiter.com/tabletop/Conways_game_life.html
- Nature Of Code By by Daniel Shiffman