Cellular Automata

Lucas Rodrigues[1]

[1]Undergraduate Student, Mathematical Sciences
Florida Atlantic University

Nonlinear Dynamical Systems, Fall 2020

Basis of Cellular Automata

Cellular automata are structures created by operations that iterate through generation done on an initial set of cells with the operation being a predetermined rule set. Each generation is obtained by performing the rule set on the cell's corresponding predecessor, and its neighbors. In this lab, we are only working with one dimensional elementary CA which uses a single line of grid as a generation and the two nearest cells as the neighborhood of a cell. In the general case for this CA, each "neighborhood set" including a particular cell and its 2 neighbors may only have $2^3 = 8$ different possible values, since each particular cell only has 2 different possible values. We are interested in a rule set that maps each of these possible neighborhood sets onto a binary number, indicating whether that particular cell will be colored. Since we have 8 possible neighborhood sets, we will have $2^8 = 256$ possible binary rule sets for this kind of CA. Two methods of figuring out a value of a cell are listed below:

- Empirical Formula
- Neighbour States

Neighbour Cell State Rule

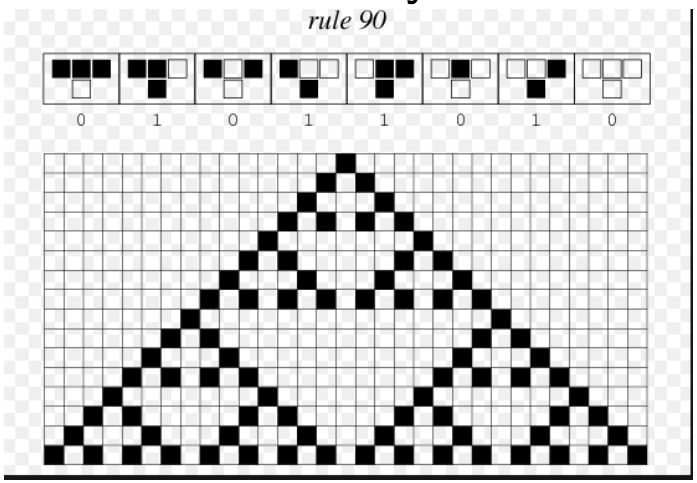
current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	0	1	1	0	1	0

The above image describes the rule 90 based on the above cell states. This rule is equivalent to performing the XOR logical operation on the outer two cells, which gives rise to the triangle shaped fractal.
Using this binary rule, one can create a loop that iterates through the corresponding previous generation cell and its neighbors, apply the rule, and obtain the binary value, which will tell us if the cell should be colored or not.

Empirical Formula

```
X[i,j] = (rule/(2**(4*X[i-1,j-1] + 2*X[i-1,j] + X[i-1,j+1]))) % 2
```

This formula can be applied generally to each of the 256 possible CAs of this kind, which condenses the previous process into a single computation, with no necessity of checking the previous values.



The algorithm gives rise to this triangle.