Cellular Automata Lab 3 Report

Nickolas Arustamyan

September 2020

Abstract

In this experiment we were exposed to early cellular automata and how they can generate interesting patterns. We have examined 3 different methods to generate a specific shape designated by rule 90. This shape is similar to Sierpinski's triangle we saw in the previous lab.

Introduction and Theory

A cellular automaton is a grid that colors or uncolors certain cells based on the previous levels values and a predetermined rule set. In our particular lab, we only are working with a 1 dimensional input, the line above. And within that line we only care about the cell directly above, and the two cells directly diagonal to the cell. Because there are only 3 cells in consideration and each cell has 2 possible values, we have a total of 8 possible values. In total that means for this sort of CA, we have $2^8 = 256$ possible rule sets to choose from.

Rule 90

For this lab, we are looking at Rule 90, defined by:

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

Where 1 indicates a colored cell and 0 an empty white cell.
One might notice this is the same as using the XOR function on the first and last number. This may also be interpreted as addition mod 2. Using that rule will give us a mod 2 version of pascals triangle!

Methods

Method 1:

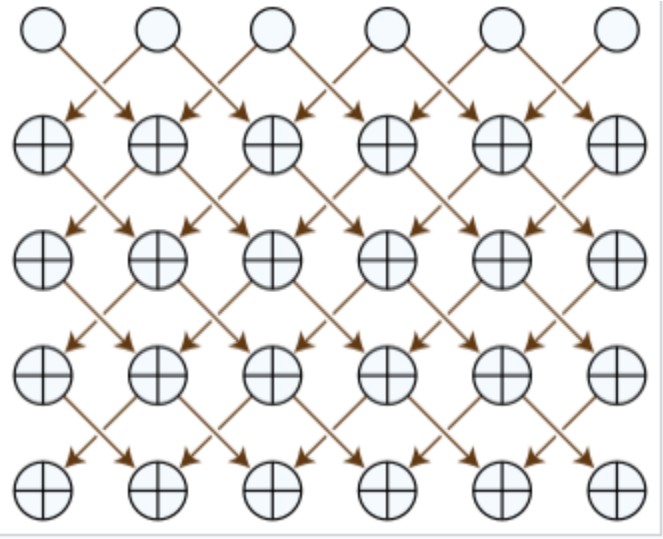
We start with a single colored block in the first generation. The excel sheet can then either be coded using the direct formula, which we will look at more closely in Method 3, or by simple evolutionary iteration and the XOR property of the rule 90. Using these rules, we take a cell in the first generation and look at it's neighbors. We then use the rule set on each set of 3 blocks in a line and then when the line is finished, go down to the next line.

Method 2:

This method does essentially the same thing of following a rule set and going down a grid. In this case, we have defined a 100x100 grid which is totally filled with 0's except for our initial value in the top middle slot. Then, we run each cell through the rule set we have defined in essentially a table and match the values.

Method 3:

The method uses a smaller but equivalent version of the previous algorithms. As seen before, the rule set can be expressed as the XOR function in respect to its neighbors, meaning it does not matter what the value of the cell is. Because of this, we can arrive at the following relationship graph:



Which can expressed in a formula like this

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

Conclusions

In conclusion, we have looked at 3 separate but related ways to generate a CA. Each rule set has a different formulation but many result in interesting patterns. In our case, we have looked at once that ends up generating a modulo 2 version of Pascals triangle. All from one singular input, we have found a pattern.