Computational Physics, Project 4 -
One dimensional cellular automata

Ena Salihovic
Jacobs University Bremen

Due November 3rd, 2021

Background

A boolean cellular automata is a collection of cells that can be in one of two states, on and off, 1 or 0. The states of each cell varies in time depending on the connections, called rules, between the cells.

There is a grid of cells. Each cell has a state (binary: 0 or 1) and a neighborhood. How a cell state changes over time is determined by analyzing its neighbor states (generations over time). the evolution of an elementary cellular automaton can completely be described by a table specifying the state a given cell will have in the next generation based on the value of the cell to its left, the value the cell itself, and the value of the cell to its right. There is a set of rules for how cell states change over time for one dimensional CA.

A ruleset requires 8 binary numbers. There are $2^8 = 256$ states an elementary CA can be defined. There are 4 possible outcomes: a) uniformity, b) repetition, c) random, d) complexity. As a result of all of this, a pseudo-random intelligent behavior is produced.

a) 1D Boolean CA

Goal: Write a program which solves the iteration of a 1D Boolean CA. The code works for N sites and one is able to select between finite grids or periodic boundary conditions. Code can be found in 4-a.py. Rule 30 was chosen to test the code (Figure 1).

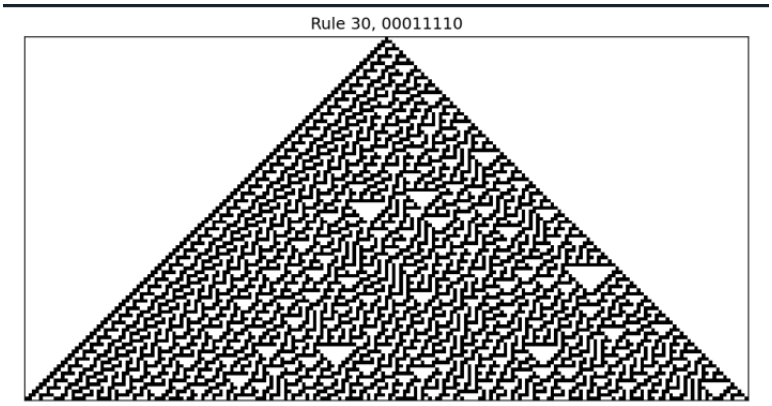

Figure 1: Visualization of Rule 30

b) Rule 150

Goal: In this part, the properties of a rule 150 or rule 10010110 were determined. This is a rule for which the value of a site at step $t+1$ is the sum modulo 2 of the values of its neighbors, plus its own value at step t.

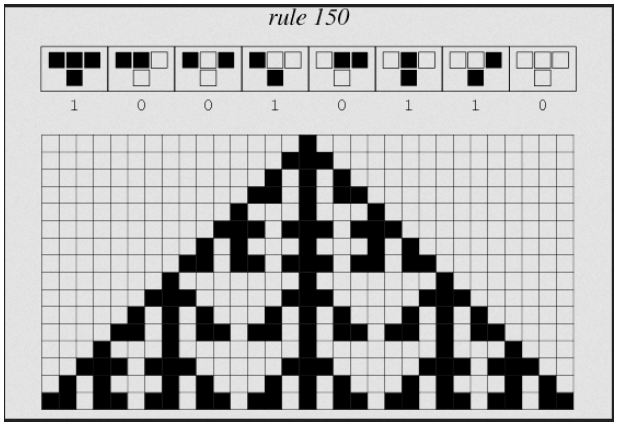

Figure 2: Visualization of rule 150 from Wolfram Alpha

Pattern produced by the rule 150 was plotted using code in 4-b.py, and can be seen in Figure (3).

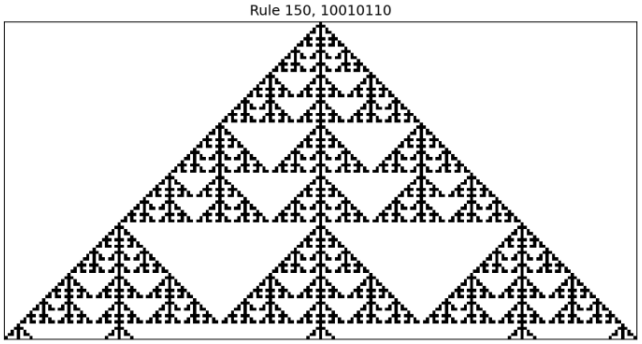

Figure 3: Visualization of rule 150 using code in 4-b.py

The main property of rule 150: the sequence generated by this rule can be obtained by taking the coefficients of the successive powers of $(1+x+x^2)$ mod 2 and interpreting them as integers in binary.

c) Evolution rule 90, 150, 18, 73, and 136

Goal: To determine the evolution rule 90, 150, 18, 73, and 136, using random initial configuration, for which the probability of site to have either value 1 or 0 is $\frac{1}{2}$. Additionally, it was evaluated how sensitive the patterns produced by the aforementioned rules are to the initial conditions, and whether the nature of the patterns depends on the use of periodic boundary conditions.

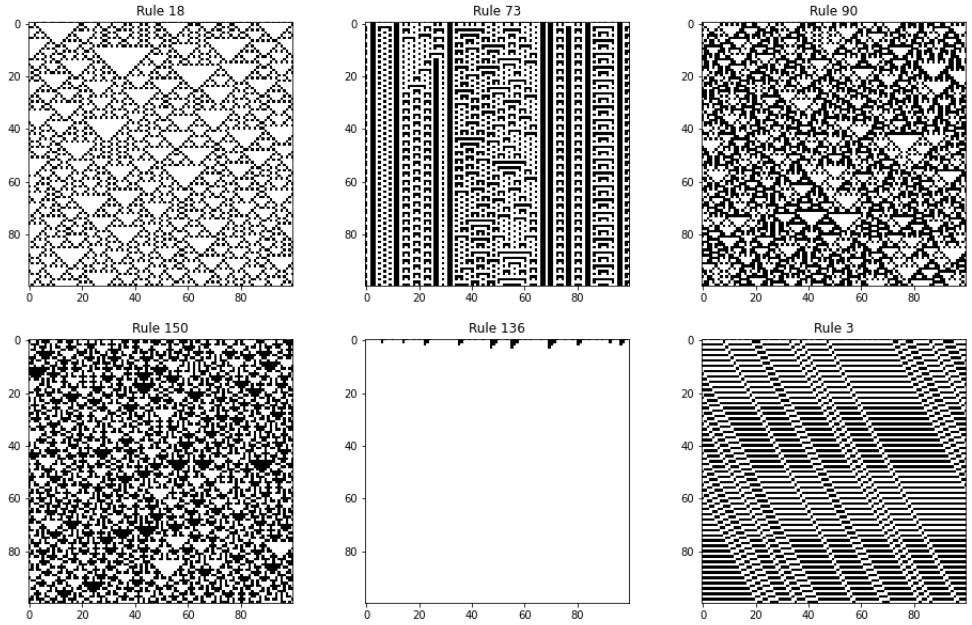

Figure 4: Visual representation of different rule sets

Random initial conditions were generated using $np.random.rand(size)$ (which can be seen in 4-c.py). It was observed that the patters were different every time the program is compiled, indicating that the patters are sensitive to the initial configuration. Furthermore, it was found that the nature of the pattern does not depend on the usage or non-usage of periodic boundary conditions, as for rule 90 a fractal/self-repeating pattern is produced.

b) Traffic rule 184

Goal: to implement the traffic rule 184 using periodic boundary conditions, and to simulate a traffic jam with 6, 12 and 18 out of 18 cars, for a ring-shaped street of 100 cells.

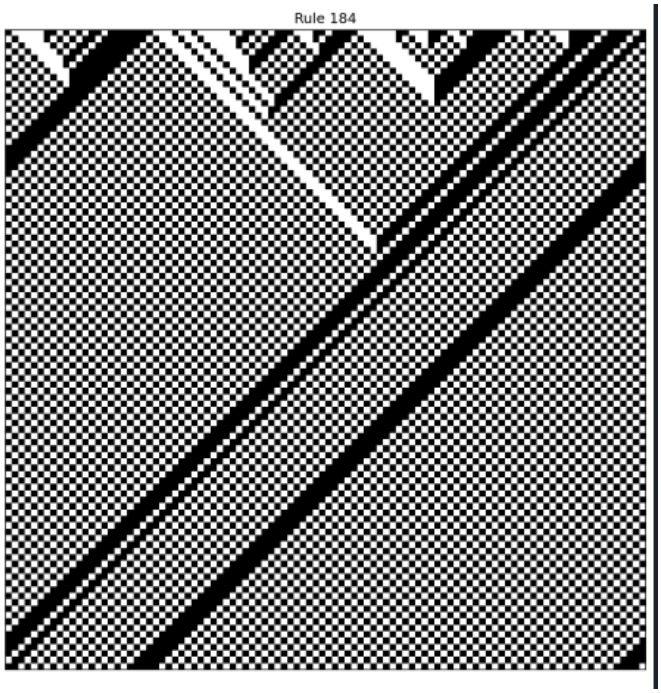

Figure 5: Rule 184 with periodic boundary conditions

I wasn't exactly sure how to implement the ring shape street of 100 cells and therefore the traffic jam was not fully simulated. However, the rule was still plotted with random initial configuration and periodic boundary conditions. Depending on the initial conditions generated, the plot changes, and simulates different scenarios of a traffic jam.

References

[1] Weisstein, Eric W. "Elementary Cellular Automaton." From MathWorld–A Wolfram Web Resource. *https://mathworld.wolfram.com/ElementaryCellularAutomaton.html*