Elementary Cellular Automata

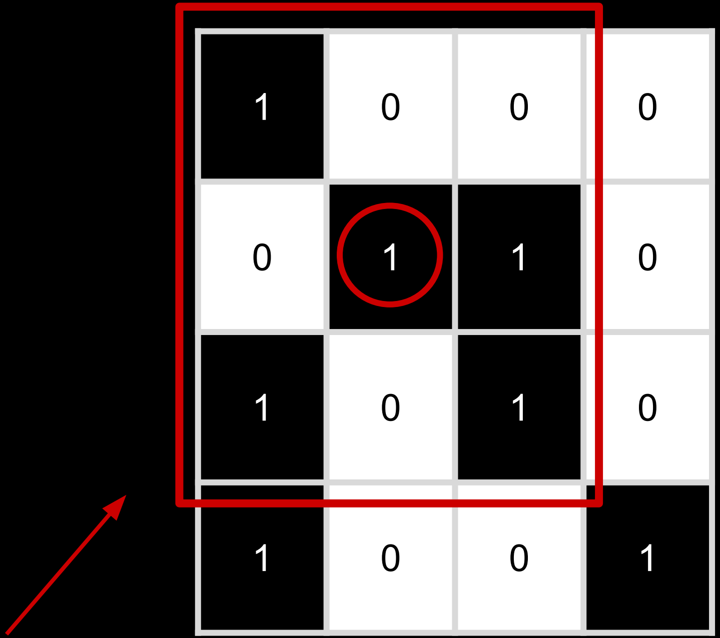
Sehajpreet Kaur, Shweta Prasad

Introduction

What are Cellular Automata?

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

- The cells live on a **grid**
- Each cell has a **state**. The number of states is usually finite, eg. binary states 0 or 1.
- Each cell has a **neighbourhood** i.e. it has adjacent cells which can be defined in a number of ways depending on the grid



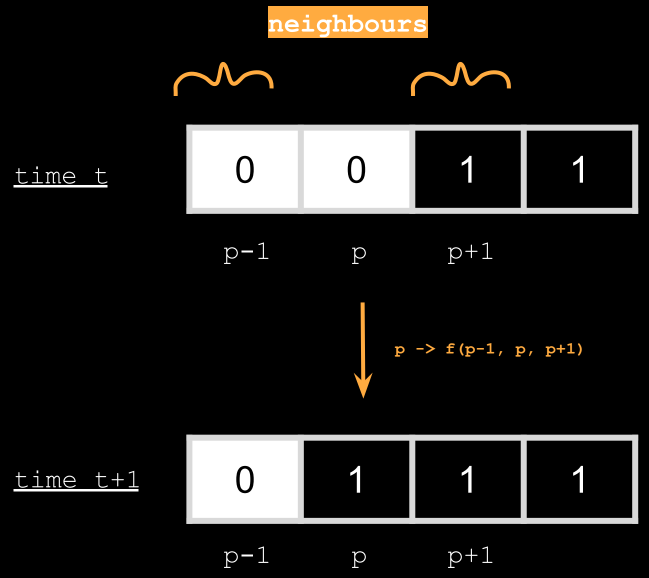
A 2-D grid of cells, each in state 0 or 1

Properties of Elementary Cellular Automata

1. **Grid**
   Elementary CA exist on a one-dimensional grid: they can be visualised as a horizontal array of cells

2. **States**
   Either of two possible states for each cell: denoted by 0 or 1

3. **Neighbourhood**
   Nearest neighbour scheme: state of cell p at time t+1 depends on the states of cells p−1, p, and p+1 at time t



Defining Rules for CA

Step 1 Define the neighbourhood:

For any given cell c, the neighbourhood consists of cells whose states at timestep t influences the state of c at timestep t+1. Refer to image shown in slide 3. The maximum distance at which neighbours can influence the cell is denoted by the number r. Let the ordered set of the neighbours' states be called *G*.

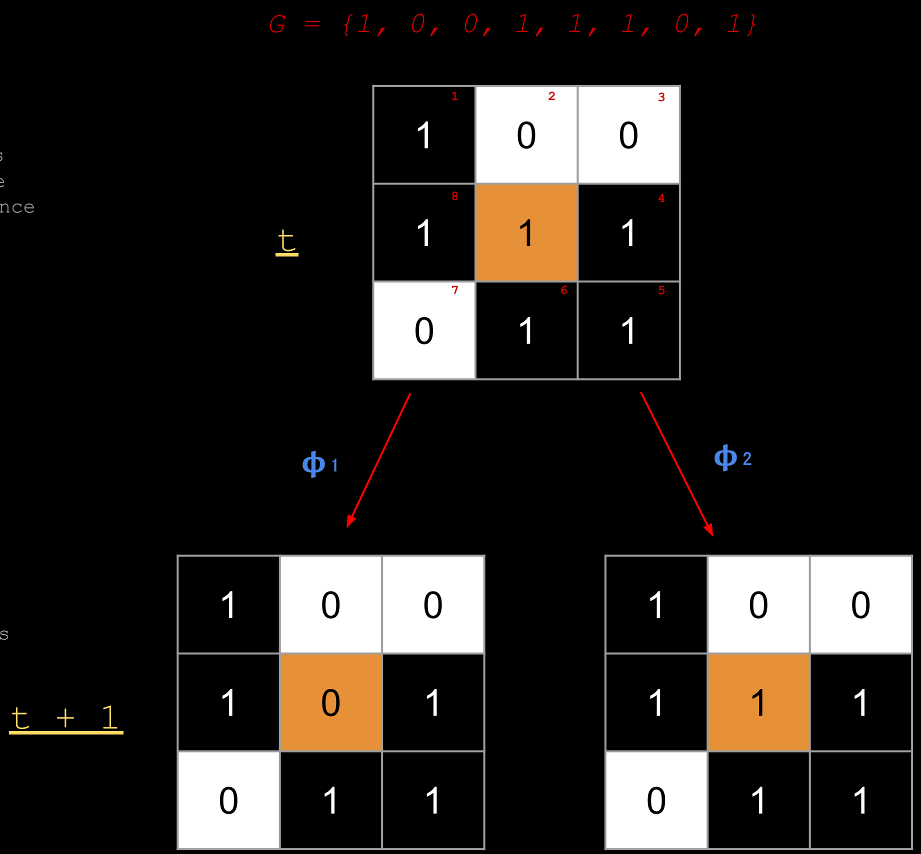
Step 2 Define the number of states that c can assume:

A cell can be in any number of states from the finite set of possible states, *M* . The cardinality of this set is denoted by k.

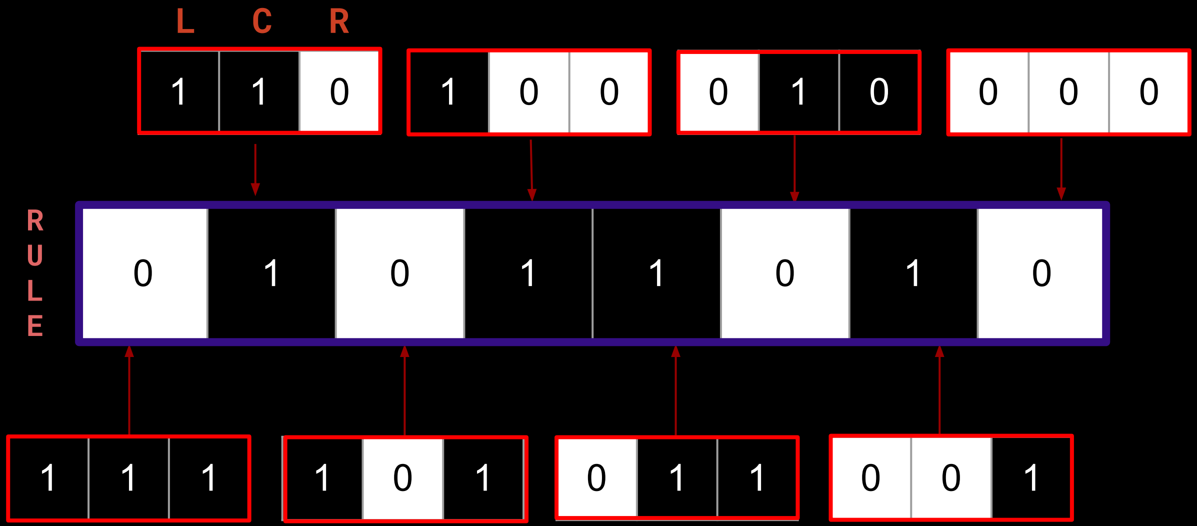
Step 3 Define the 'rule' of interaction between c and its neighbourhood:

The transition rule ϕ for each cell in a given state S(c,t) defines its state in the next time step, S(c,t+1). ϕ is applied on *G* to give S(c,t+1).

$$S(c,t+1) = \phi(G \cup S(c,t))$$



Rules for Elementary CA



Boolean representation of rule 90:

$$S(c,t) = S(L_c, t-1) \bigoplus S(R_c, t+1)$$

Elementary CA are those 1-D CA with $r = 1$ (only the neighbours adjacent left and right to the cell form the neighbourhood) and $k = 2$ (only 2 possible states for a given cell in the grid).

Shown here is rule 90.

Everytime a triple of one of the 8 configurations is encountered, it is used to index into the appropriate position in the RULE array. The value in the corresponding cell in the RULE array tells us what the next state of the center cell will be.

One can represent the relationship in terms of boolean expressions, as arrays (as shown here) or as dictionaries.

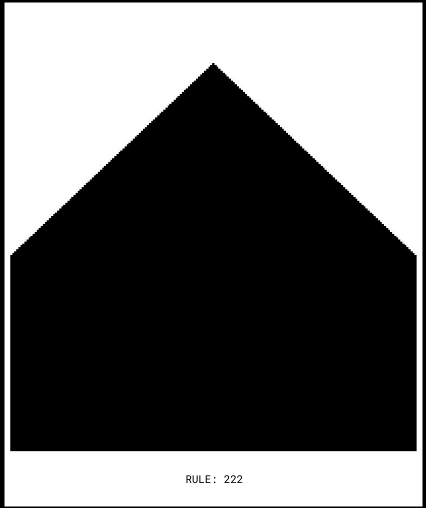
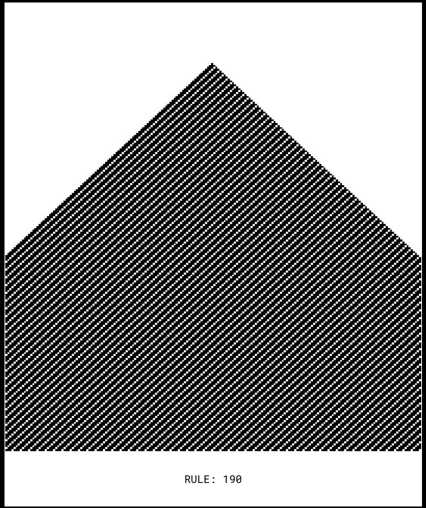
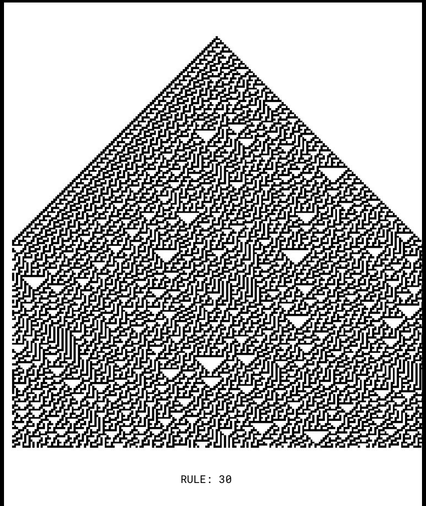
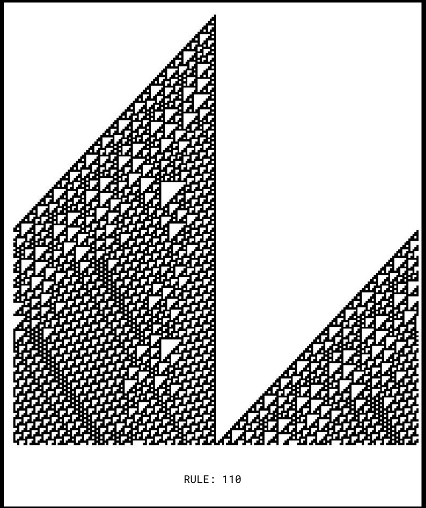
The rule is labelled 90 based on the decimal value of the rule when read as a byte.

One can thus see that the total number of possible rules is :

$$2^{2^3} = 256$$

Classes of Cellular Automata

For an arbitrary initial condition, it seems that the patterns produced by each of the 256 rules are remarkably distinct from each other. However, a closer look suggests that the number of fundamentally different patterns is actually just four. **This classification holds across CA types.**

Class 1: Uniform	Class 2: Periodic	Class 3: Random	Class 4: Complex
			

Methods: Simulation and Visualization of CA Behavior

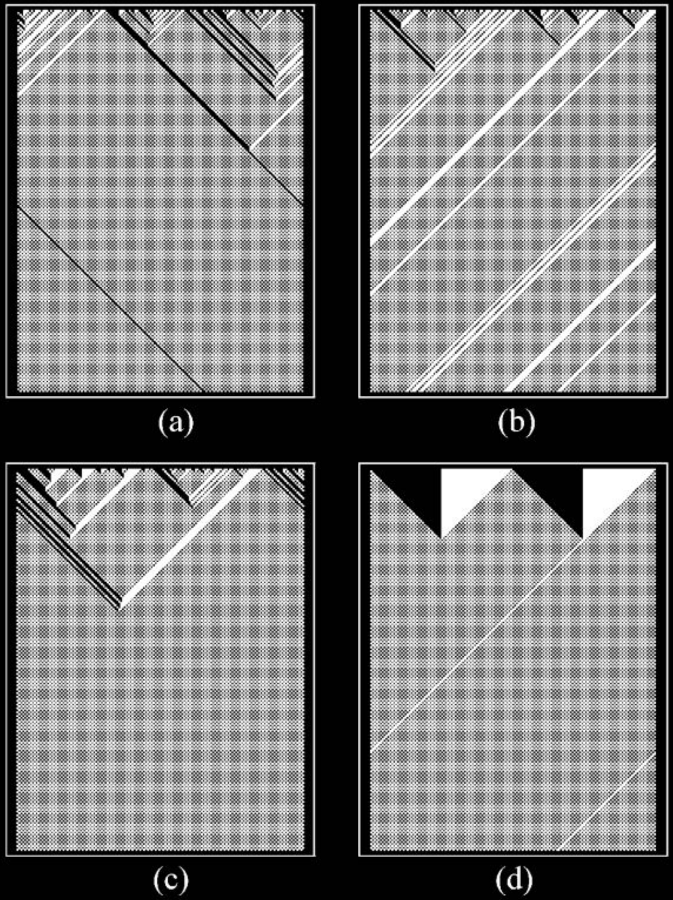
The Majority/Density Problem

A solution to the density problem:

"Two-state, r=1 Cellular Automaton that Classifies Density" (Capcarrere et.al, 1996)

Previous Approaches: The 1-D, two-state CA in an arbitrary initial configuration should converge in time to a state of all 1s if the density of 1s in the initial configuration > 0.5, and to all 0s if this density < 0.5; for an initial density of 0.5, the CA's behavior is undefined.

Novel Approach: Instead of finding a more efficient rule, change the way the final classification is represented (do not need to converge to a state of all 1s or 0s). The grid relaxes to a limit cycle within ⌈N/2⌉ time steps (no aperiodic behaviour is observed). Correct classification with 100% accuracy is achieved using Rule 184 of elementary CA. If the density of 1s in the initial configuration is > 0.5, the configuration at the end of ⌈N/2⌉ time steps has at least one block at least two consecutive 1s. In case the density is < 0.5, there is at least one block with at least two consecutive 0s.



(a) (b) (c) (d)

Rule 184 :

$$s_i(t+1) = \begin{cases} s_{i-1}(t), & \text{if } s_i(t) = 0 \\ s_{i+1}(t), & \text{if } s_i(t) = 1 \end{cases}$$

1	0	1	1	1	0	0	0

For a finite-size CA of size N, let $S(t) = \{s_0(t), ..., s_{N-1}(t)\}$ be the grid configuration at time step t, let $D(\{s_i(t), ..., s_{i+k-1}(t)\})$ be the density of 1s at time t over a block of k cells at positions $\{i, ..., i+k-1\}$, and let $T = \lceil N/2 \rceil$. Then, note the following:

(1) If $D(S(0)) > 0.5$, then (a) there exists a pair of adjacent cells $i, i+1$ such that $s_i(T) = 1$ and $s_{i+1}(T) = 1$, and (b) for all i, $s_i(T) = 0$ implies $s_{i+1}(T) = 1$

(2) If $D(S(0)) < 0.5$, then (a) there exists a pair of adjacent cells $i, i+1$ such that $s_i(T) = 0$ and $s_{i+1}(T) = 0$, and (b) for all i, $s_i(T) = 1$ implies $s_{i+1}(T) = 0$

(3) If $D(S(0)) = 0.5$, then for all i, $s_i(T) \neq s_{i+1}(T)$

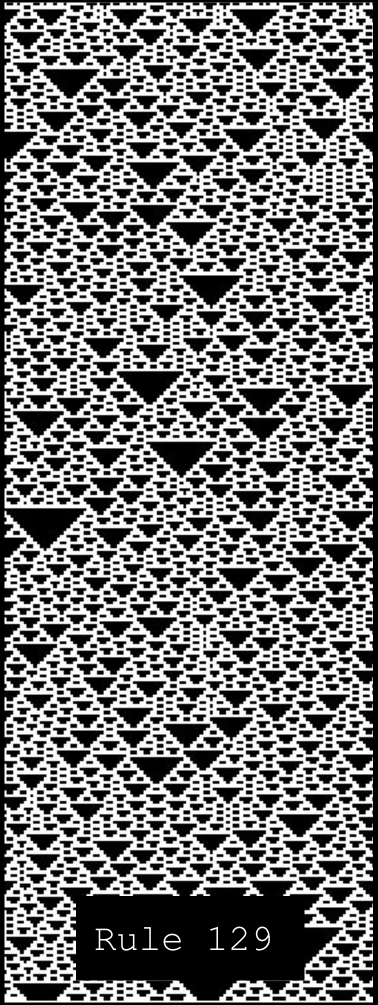
Theorem

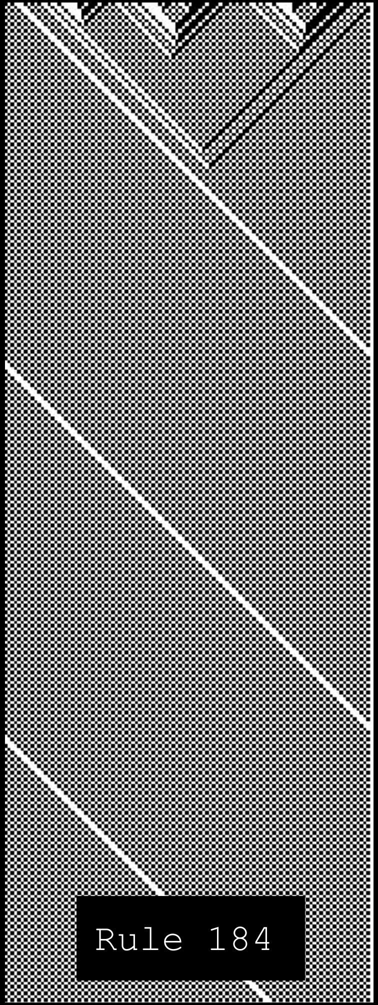
Complexity and Performance

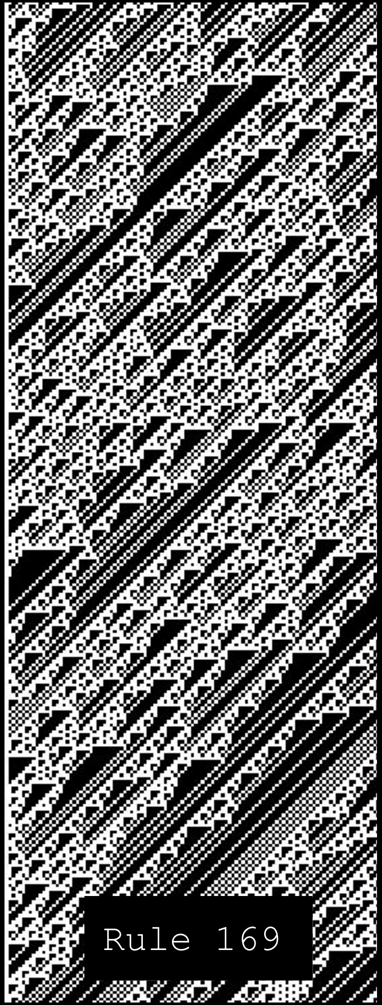
- **Kolmogorov Complexity:** For a finite string, this is concerned with the size of the shortest program that computes or produces the string. Since we start with an arbitrary initial configuration, this complexity is quite high. However, there is a notable reduction using both the fixed-point output and the novel "blocks" output.

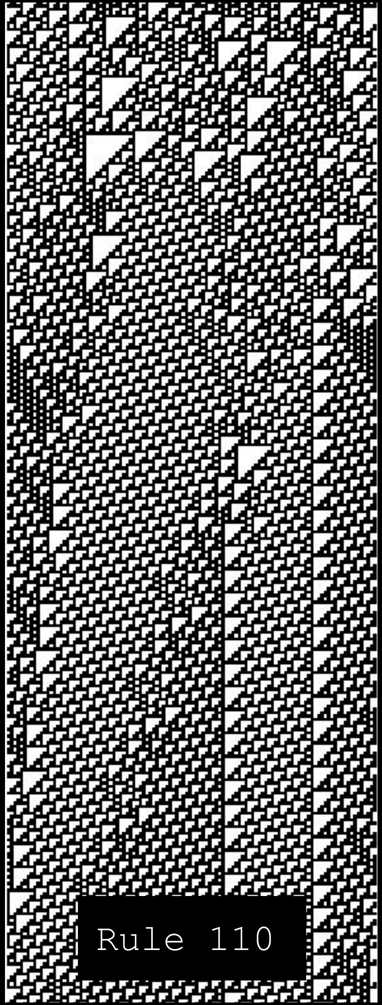
- **Computational Complexity:** The complexity of the final configuration in the fixed-point system is that of a simple regular language (all 0s or all 1s). Even in the novel approach, the complexity of the output is that of a regular language (identification of a block of two state-0 or state-1 cells).

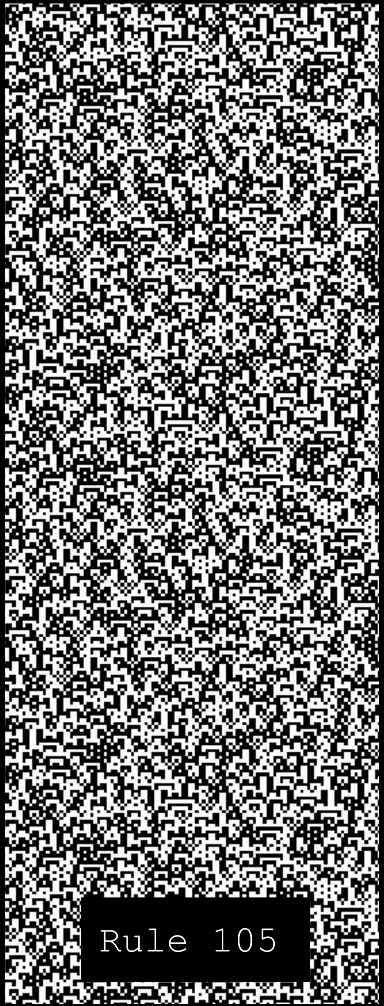
In Conclusion

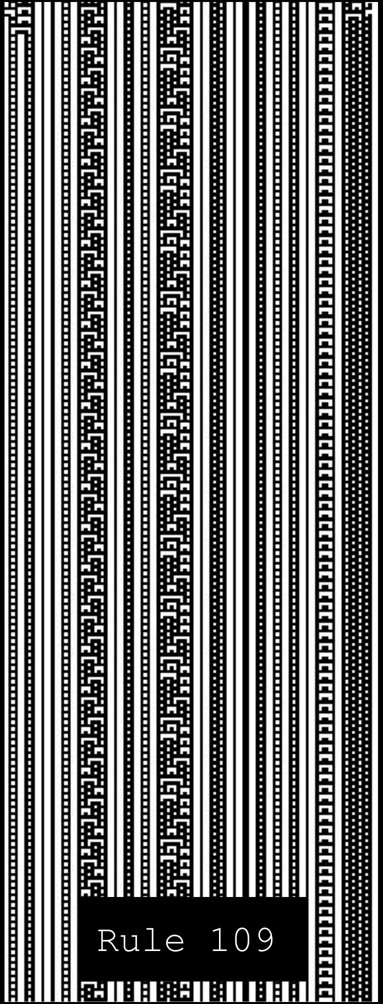

Rule 129


Rule 184

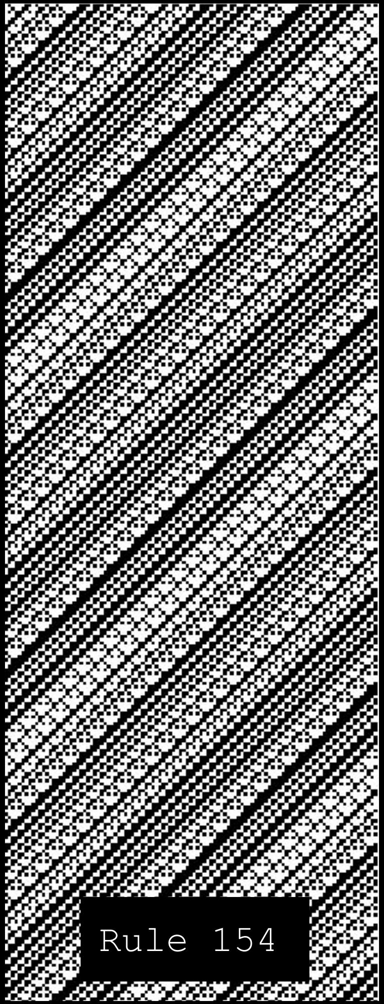

Rule 169


Rule 110


Rule 105


Rule 109


Rule 30


Rule 154

What we plan to explore further:

1. A systematic study of the 256 rules for elementary cellular automata, and their sensitivity to initial condition: by means of this, we will try to make our own observations about the properties of elementary CA, and compare with the extensive existing studies.

2. A proper understanding of the proof in Capcarrere 1996, to show that rule 184 does indeed solve the density classification problem

3. Applying this solution to studying and modelling stomatal patchiness

4. If time permits, trying to simulate CA using parallel computation instead of serial.

References:

1. M. S. Capcarrere, M. Sipper, and M. Tomassini,“*Two-state,r= 1 cellular automaton that classifies density,”* Phys. Rev. Lett., vol. 77, pp.4969–4971, Dec 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.77.4969

2. S. Wolfram, “*Statistical mechanics of cellular automata,”* Rev. Mod. Phys.,vol. 55,pp.601–644, Jul 1983. [Online]. Available:https://link.aps.org/doi/10.1103/RevModPhys.55.601

All images have been simulated/created locally.