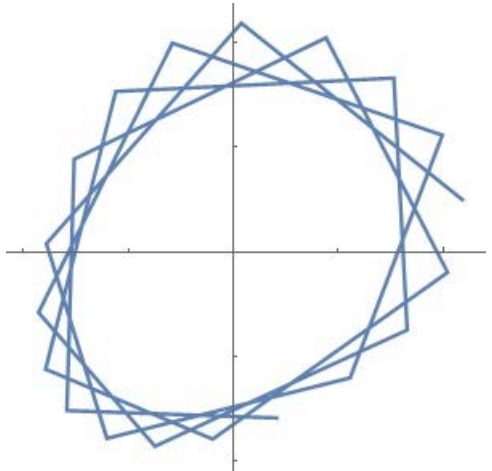
Cellular automata

theory

Notes for CAS

Feijs, Delbressine



Cellular automata:

- Model of interacting agents in fixed grid
- Very simple rules, yet complex behaviour
- Fundamental studies about computation and complexity
- Also useful for simulations in physics and social computation

Good overview: http://slideplayer.com/slide/5674720/

Mathworld: A cellular automaton is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells.

The rules are then applied iteratively for as many time steps as desired. von Neumann was one of the first people to consider such a model, and incorporated a cellular model into his "universal constructor." Cellular automata were studied in the early 1950s as a possible model for biological systems (Wolfram 2002, p. 48).

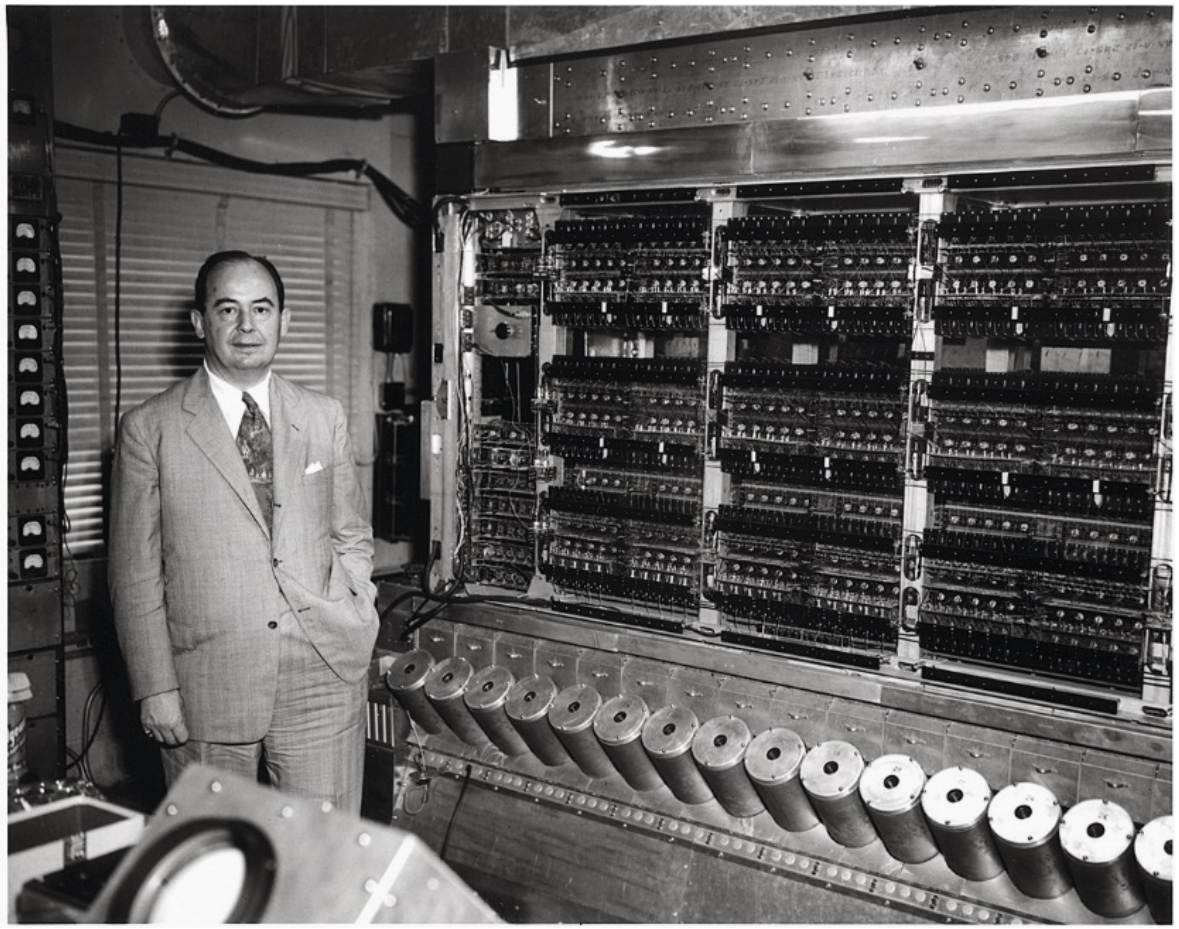
Comprehensive studies of cellular automata have been performed by S. Wolfram starting in the 1980s, and Wolfram's fundamental research in the field culminated in the publication of his book *A New Kind of Science* (Wolfram 2002) in which Wolfram presents a gigantic collection of results concerning automata, among which are a number of groundbreaking new discoveries.

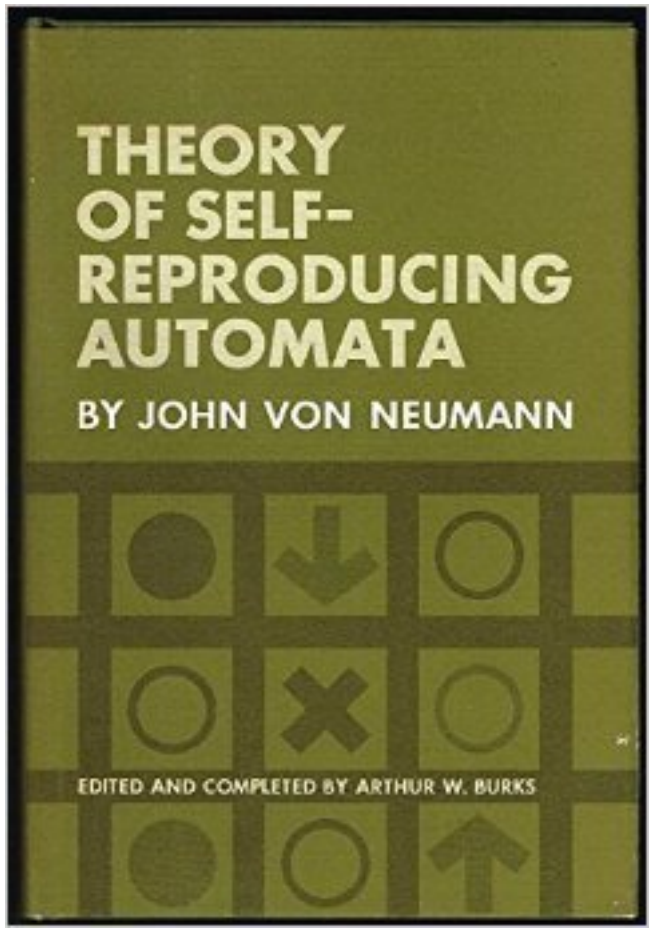
Cellular automata:

- every cell (agent) has finite-state machine behaviour
- neighbour states + own state determine new state
- all cells are updated simultaneously
- the rules are the same for all cells

History:

- John Von Neumann (1948, self-replication)
- John Conway (1970, universality)
- Steven Wolfram (2002, complexity)

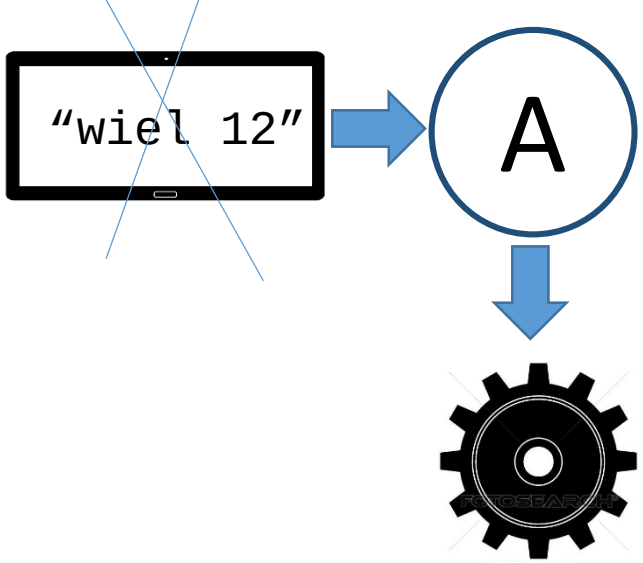


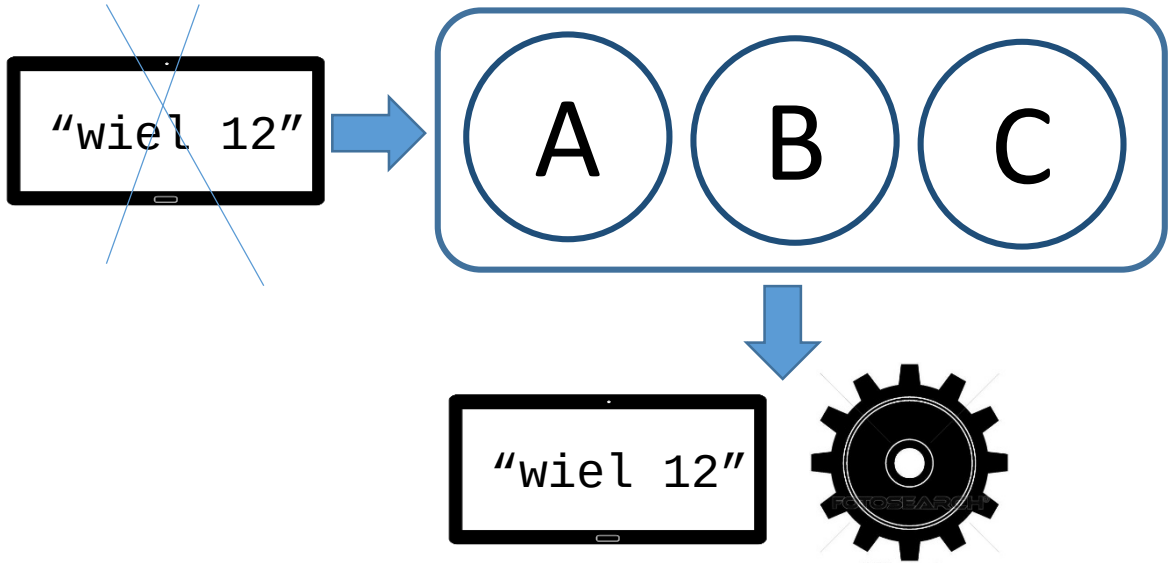

THEORY
OF SELF-
REPRODUCING
AUTOMATA
BY JOHN VON NEUMANN
EDITED AND COMPLETED BY ARTHUR W. BURKS

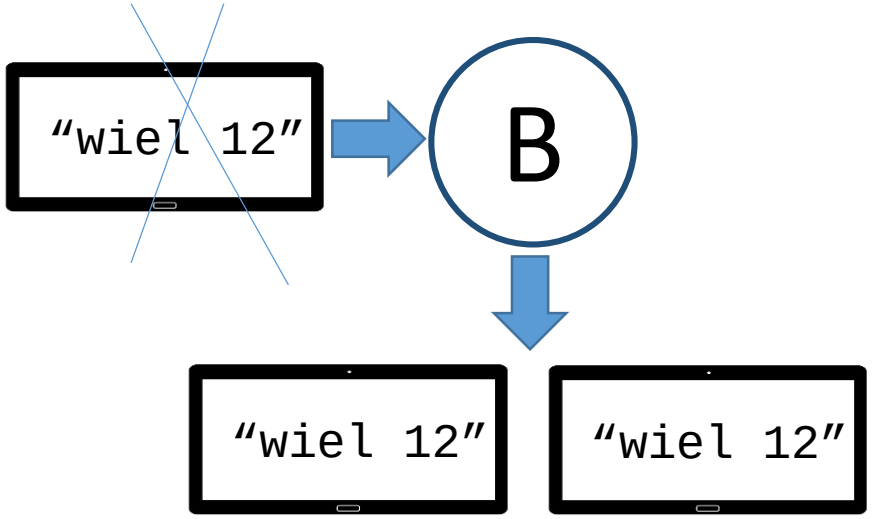
The general constructive automaton A produces only X when a complete description of X is furnished it, and on any reasonable view of what constitutes complexity, this description of X is as complex as X itself. The general copying automaton B produces two copies of $\phi(X)$, but the juxtaposition of two copies of the same thing is in no sense of higher order than the thing itself. Furthermore, the extra unit B is required for this copying.

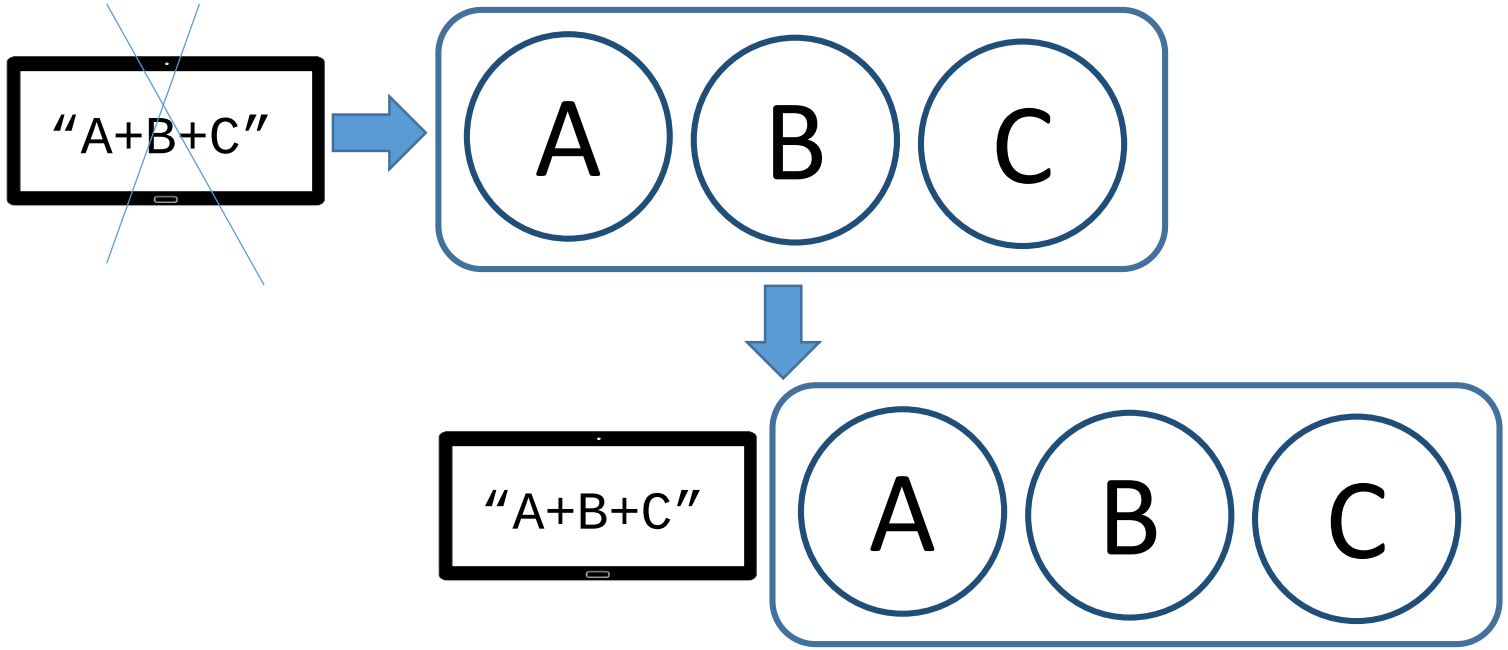
Now we can do the following thing. We can add a certain amount of control equipment C to the automaton $A + B$. The automaton C dominates both A and B, actuating them alternately according to the following pattern. The control C will first cause B to make two copies of $\phi(X)$. The control C will next cause A to construct X at the price of destroying one copy of $\phi(X)$. Finally, the control C will tie X and the remaining copy of $\phi(X)$ together and cut them loose from the complex $(A + B + C)$. At the end the entity $X + \phi(X)$ has been produced.

archive.org/details/theoryofselfrepr00vonn_0

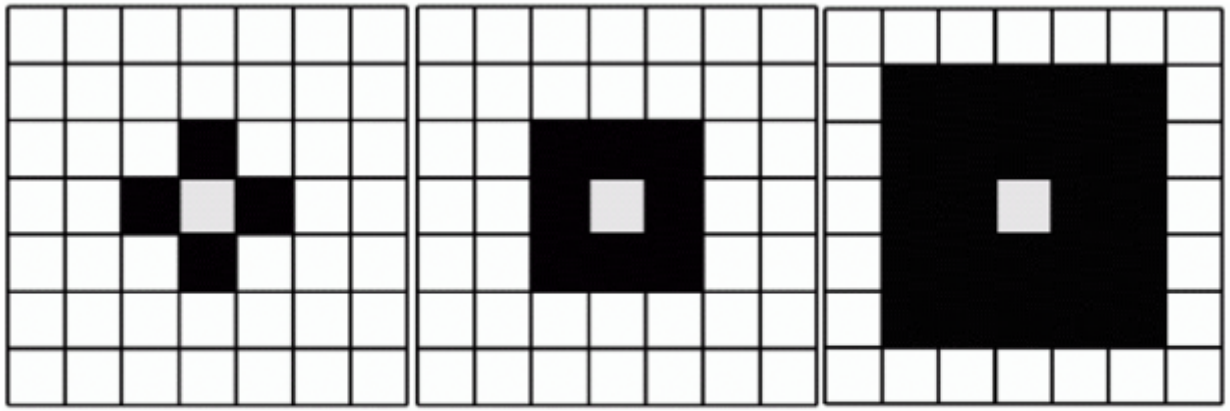

“wiel 12”
A


“wiel 12”
A
B
C
“wiel 12”


“wiel 12”
B
“wiel 12”
“wiel 12”


“A+B+C”
A
B
C
“A+B+C”
A
B
C

https://www.openabm.org/book/export/html/1949



von Neumann Neighborhood | Moore Neighborhood | Extended Moore Neighborhood

An Implementation of von Neumann's Self-Reproducing Machine

Umberto Pesavento
Princeton University
Class of 2000
Princeton, NJ 08544
pesavent@intercity.shiny.it

Keywords
cellular automata, self-reproduction, universal constructor

Abstract This article describes in detail an implementation of John von Neumann's *self-reproducing machine*. Self-reproduction is achieved as a special case of construction by a *universal constructor*. The theoretical proof of the existence of such machines was given by John von Neumann in the early 1950s [6], but was first implemented in 1994, by the author in collaboration with R. Nobili. Our implementation relies on an extension of the state-transition rule of von Neumann's original cellular automaton. This extension was introduced to simplify the design of the constructor. The main operations in our constructor can be mapped into operations of von Neumann's machine.

a **ground** state **U** (48, 48, 48)

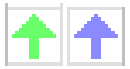
the **transition** or **sensitised** states (in 8 substates)

- **S** (newly sensitised) (255, 0, 0)
- $\mathbf{S_0}$ - (sensitised, having received no input for one cycle) (255, 125, 0)
- $\mathbf{S_{00}}$ - (sensitised, having received no input for two cycles) (255, 175, 50)
- $\mathbf{S_{000}}$ - (sensitised, having received no input for three cycles) (251, 255, 0)
- $\mathbf{S_{01}}$ - (sensitised, having received no input for one cycle and then an input for one cycle) (255, 200, 75)
- $\mathbf{S_1}$ - (sensitised, having received an input for one cycle) (255, 150, 25)
- $\mathbf{S_{10}}$ - (sensitised, having received an input for one cycle and then no input for one cycle) (255, 255, 100)
- $\mathbf{S_{11}}$ - (sensitised, having received input for two cycles) (255, 250, 125)

the **confluent** states (in 4 states of excitation)

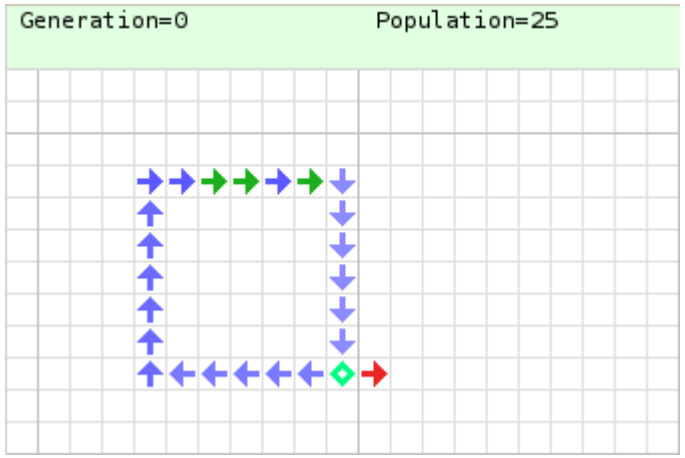
- $\mathbf{C_{00}}$ - quiescent (and will also be quiescent next cycle) (0, 255, 128)
- $\mathbf{C_{01}}$ - next-excited (now quiescent, but will be excited next cycle) (33, 215, 215)
- $\mathbf{C_{10}}$ - excited (but will be quiescent next cycle) (255, 255, 128)
- $\mathbf{C_{11}}$ - excited next-excited (currently excited and will be excited next cycle) (255, 128, 64)

the **ordinary transmission** states (in 4 directions, excited or quiescent, making 8 states)

- North-directed (excited and quiescent) 
- South-directed (excited and quiescent)
- West-directed (excited and quiescent)
- East-directed (excited and quiescent)

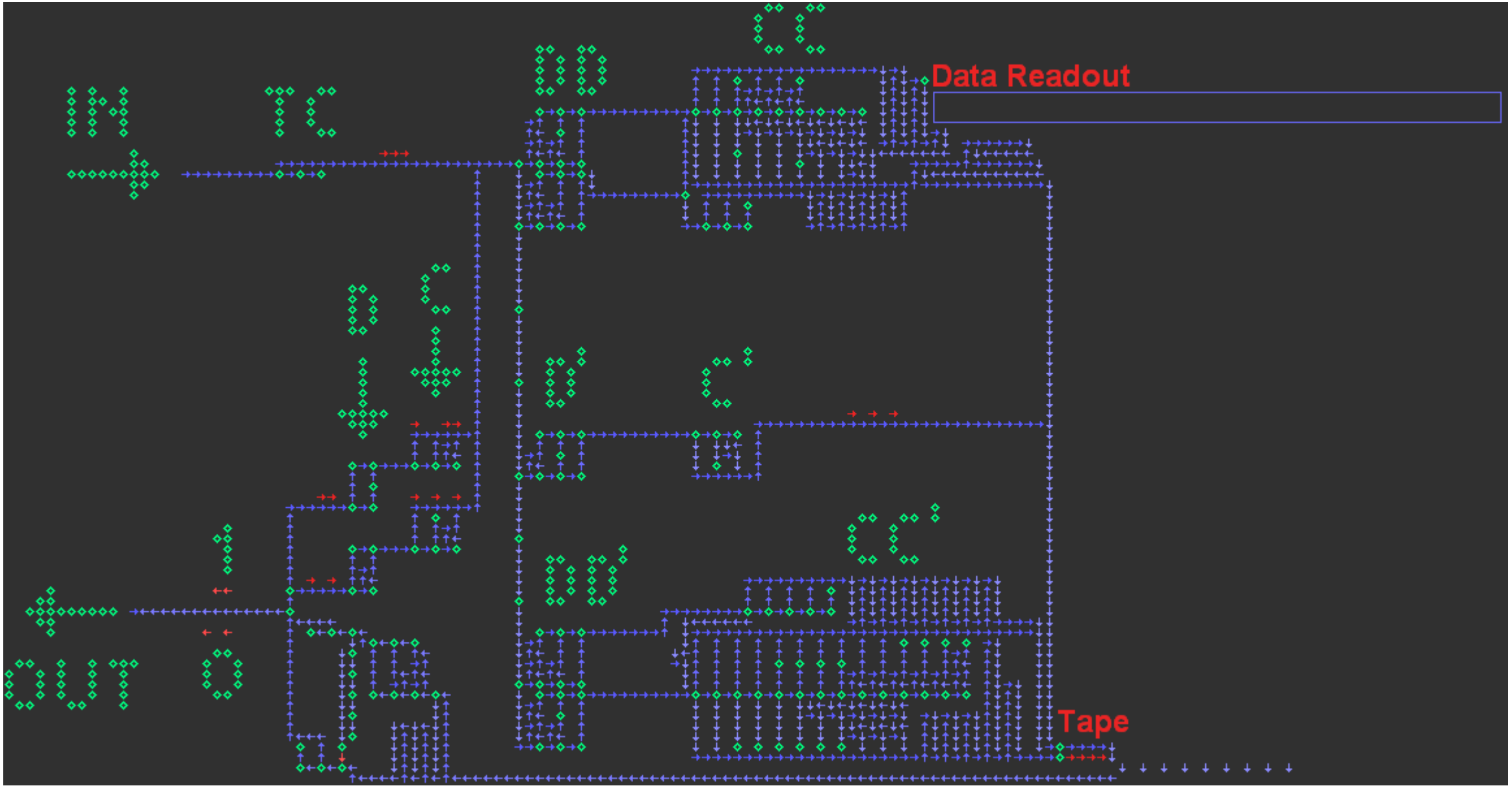
the **special transmission** states (in 4 directions, excited or quiescent, making 8 states)

- North-directed (excited and quiescent) (191, 73, 255) (255, 56, 56)
- South-directed (excited and quiescent) (203, 106, 255) (255, 89, 89)
- West-directed (excited and quiescent) (197, 89, 255) (255, 73, 73)
- East-directed (excited and quiescent) (185, 56, 255) (235, 36, 36)



A simple configuration in von Neumann's cellular automaton. A binary signal is passed repeatedly around the blue wire loop, using excited and quiescent *ordinary transmission states*. A confluent cell duplicates the signal onto a length of red wire consisting of *special transmission states*. The signal passes down this wire and constructs a new cell at the end. This particular signal (1011) codes for an east-directed special transmission state, thus extending the red wire by one cell each time. During construction, the new cell passes through several sensitised states, directed by the binary sequence

Source:
https://en.wikipedia.org/wiki/Von_Neumann_cellular_automaton



Source: https://en.wikipedia.org/wiki/Von_Neumann_cellular_automaton

QuineByPhiLho | Processing 2.2.1

File Edit Sketch Tools Help

Java

QuineByPhiLho

```
char q = 34, c = 44, n = 10;
String[] a =
{
"char q = 34, c = 44, n = 10;",
"String[] a =",
"{",
"",
"};",
"String B() { String s = new String(); for (int i = 0; i < a.length; i++) { s += q + a[i] + q + c + n; } return s; }",
"void D() { for (int i = 0; i < a.length; i++) println(a[i]); }",
"void setup() { a[3] = B(); D(); exit(); }",

};
String B() { String s = new String(); for (int i = 0; i < a.length; i++) { s += q + a[i] + q + c + n; } return s; }
void D() { for (int i = 0; i < a.length; i++) println(a[i]); }
void setup() { a[3] = B(); D(); exit(); }
```

```
char q = 34, c = 44, n = 10;
String[] a =
{
"char q = 34, c = 44, n = 10;",
"String[] a =",
"{",
"",
"};",
"String B() { String s = new String(); for (int i = 0; i < a.length; i++) { s += q + a[i] + q + c + n; } return s; }",
"void D() { for (int i = 0; i < a.length; i++) println(a[i]); }",
"void setup() { a[3] = B(); D(); exit(); }",

};
String B() { String s = new String(); for (int i = 0; i < a.length; i++) { s += q + a[i] + q + c + n; } return s; }
void D() { for (int i = 0; i < a.length; i++) println(a[i]); }
void setup() { a[3] = B(); D(); exit(); }
```

17

https://processing.org/discourse/beta/num 1243254687.html

Conway’s game of life

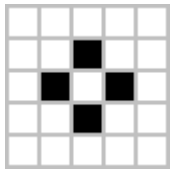
- Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any live cell with more than three live neighbours dies, as if by overpopulation.
- Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

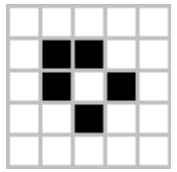
(https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)

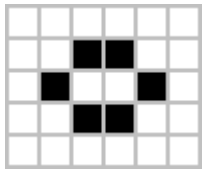
DIGITAL EQUIPMENT CORPORATIO

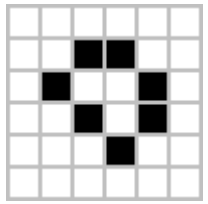
Static

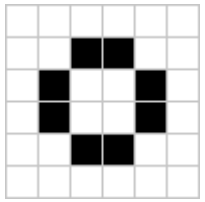
block



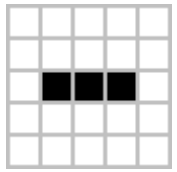


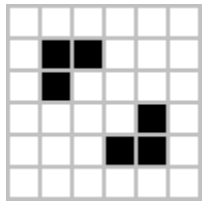

beehive

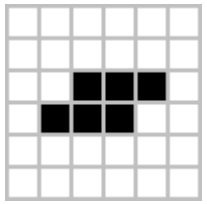


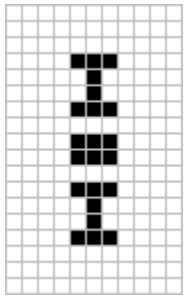

pond

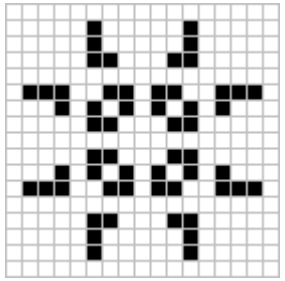
Pulsing


blinker

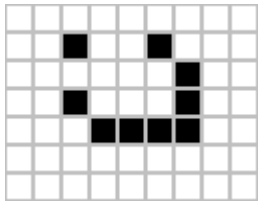


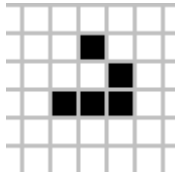


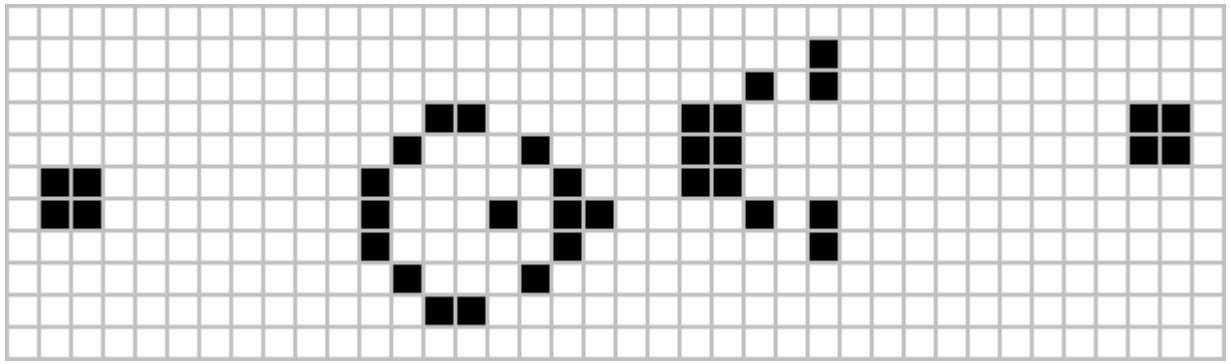




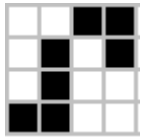
Crawling


lightweight spaceship (c/2)

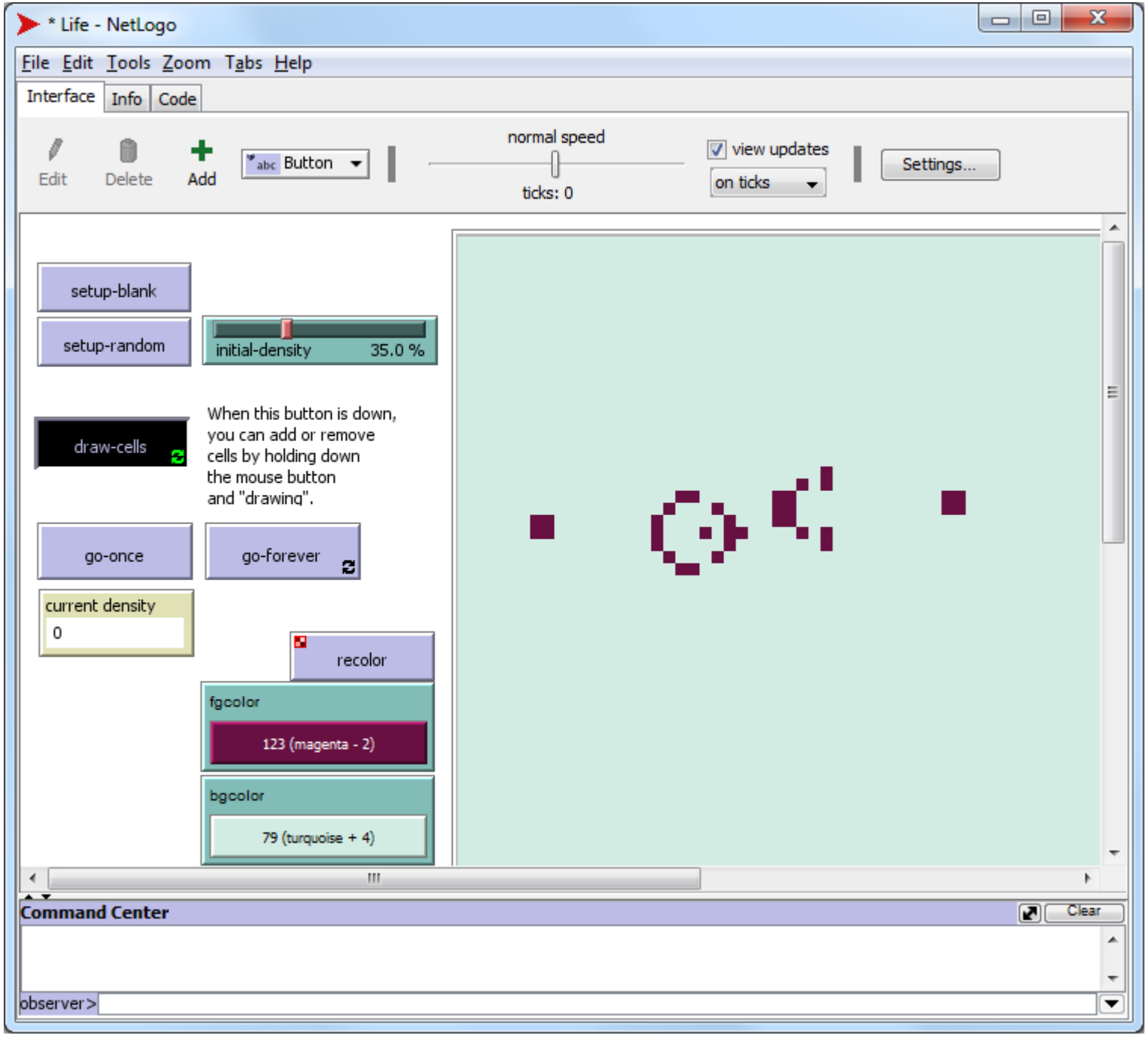

Glider (c/4)

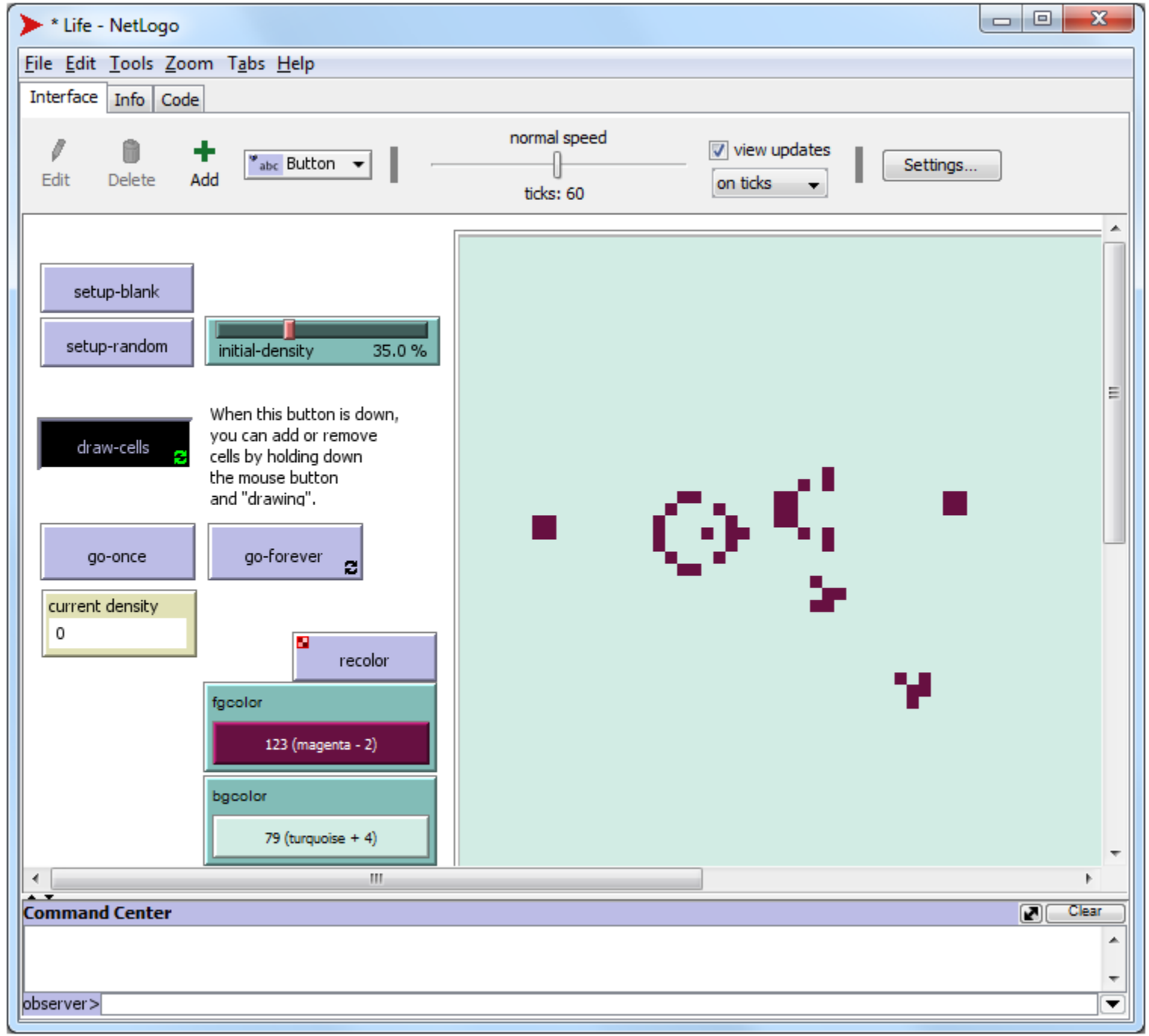


Gosper’s glider gun (wikimedia)

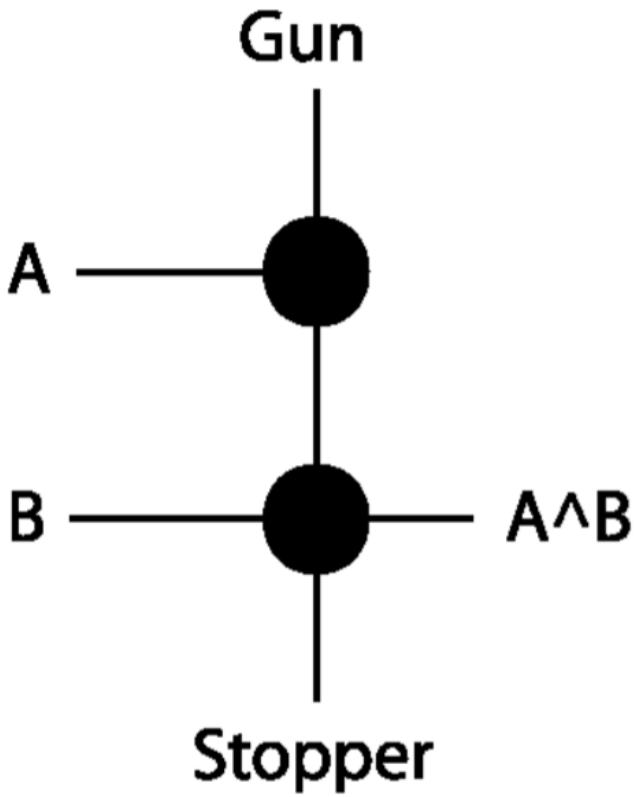


eater,
stopper

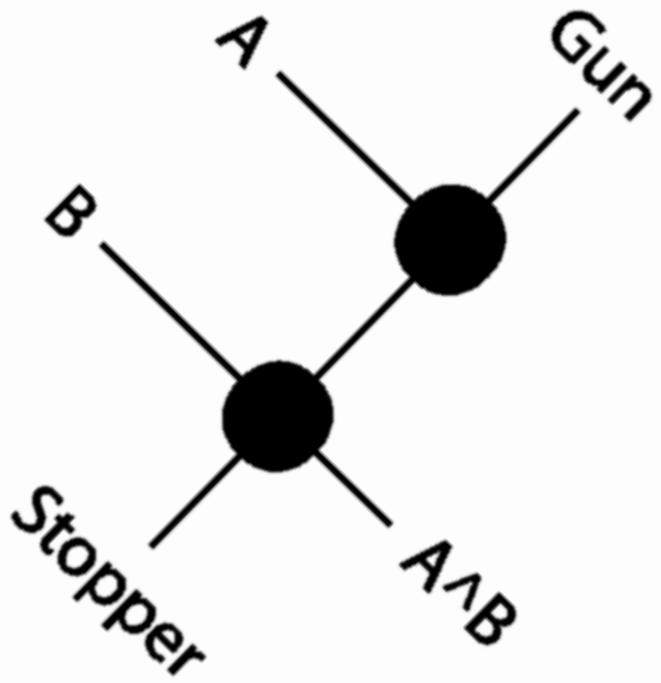

* Life - NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Edit
Delete
Add
Button
normal speed
ticks: 0
view updates
on ticks
Settings...
setup-blank
setup-random
initial-density 35.0 %
draw-cells
When this button is down, you can add or remove cells by holding down the mouse button and "drawing".
go-once
go-forever
current density
0
recolor
fgcolor
123 (magenta - 2)
bgcolor
79 (turquoise + 4)
Command Center
Clear
observer>

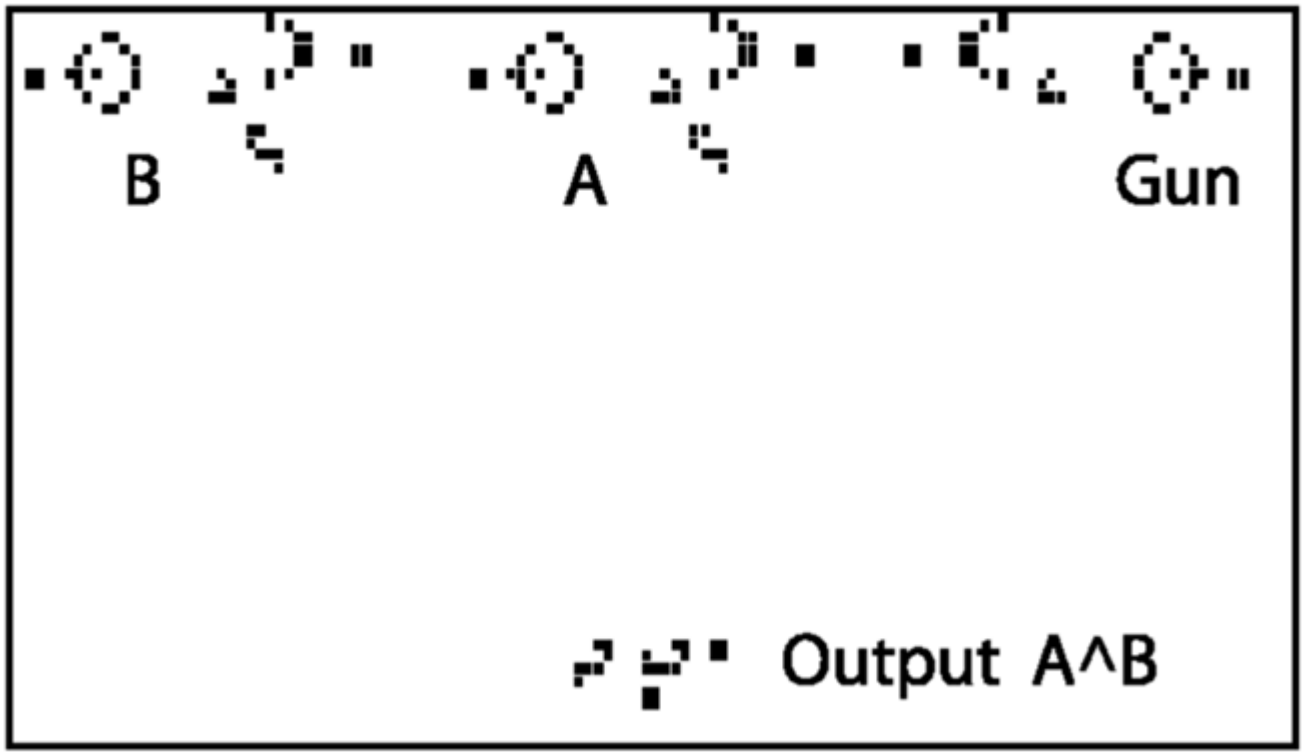

* Life - NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Edit
Delete
Add
Button
normal speed
ticks: 60
view updates
on ticks
Settings...
setup-blank
setup-random
initial-density 35.0 %
draw-cells
When this button is down, you can add or remove cells by holding down the mouse button and "drawing".
go-once
go-forever
current density
0
recolor
fgcolor
123 (magenta - 2)
bgcolor
79 (turquoise + 4)
Command Center
Clear
observer>

The AND-Gate



Source: Jean Philippe Rennard,
Implementation of Logical functions in the game of life,
Collision-based computing, 2002 - Springer,
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.9114&rep=rep1&type=pdf



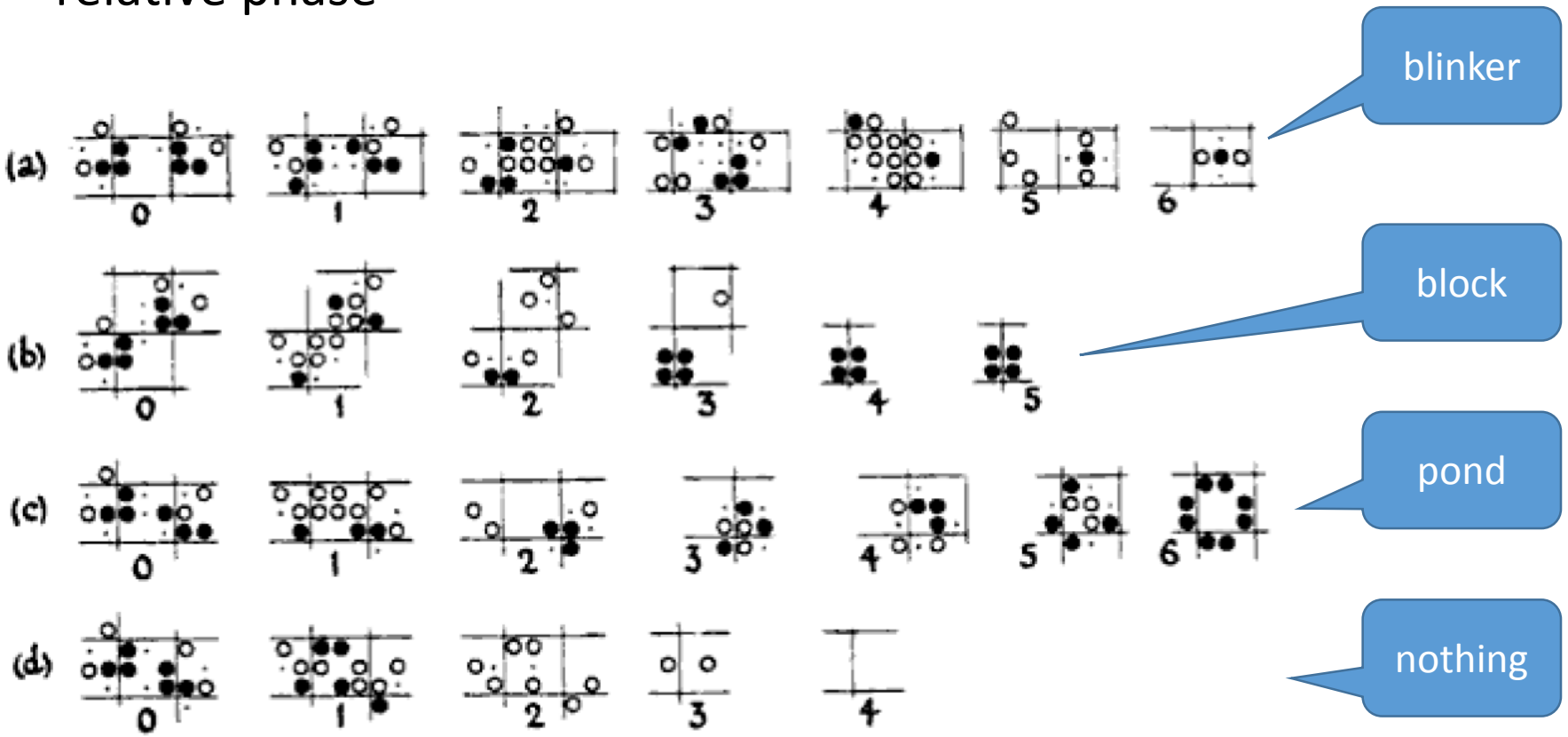


Source: Jean Philippe Rennard,
Implementation of Logical functions in the game of life,
Collision-based computing, 2002 - Springer,
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.9114&rep=rep1&type=pdf

Colliding gliders

outcome depends on:

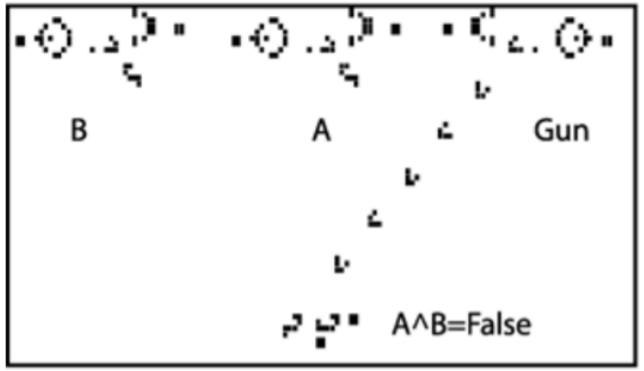
- relative position
- relative phase



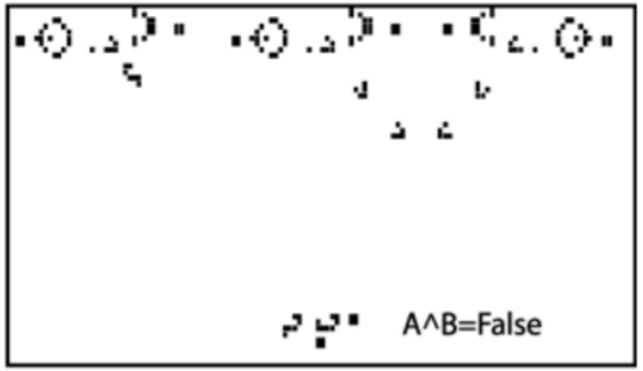
Source: Melissa Gymrek's lecture notes (MIT).
http://web.mit.edu/sp.268/www/2010/lifeSlides.pdf

No glider can activate the output. The result is False.

A False, B False



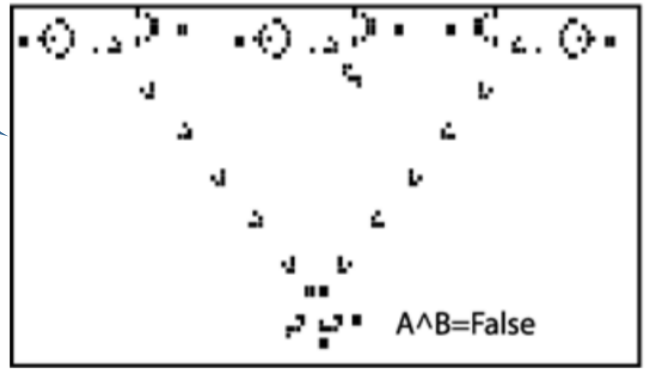
A True, B False



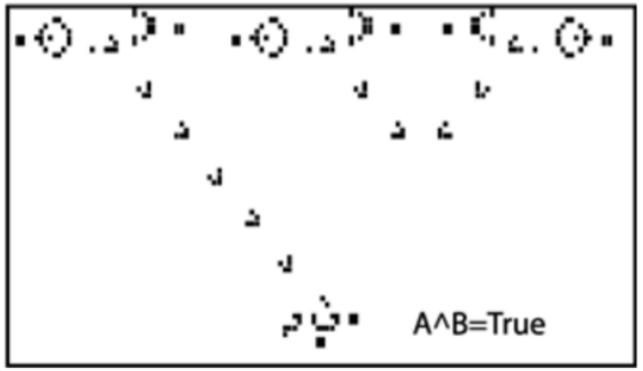
A stops the gun. Since B is false, it cannot activate the output. The result is False.

B cannot join the output since the gun stops it. The result is False.

A False, B True



A True, B True

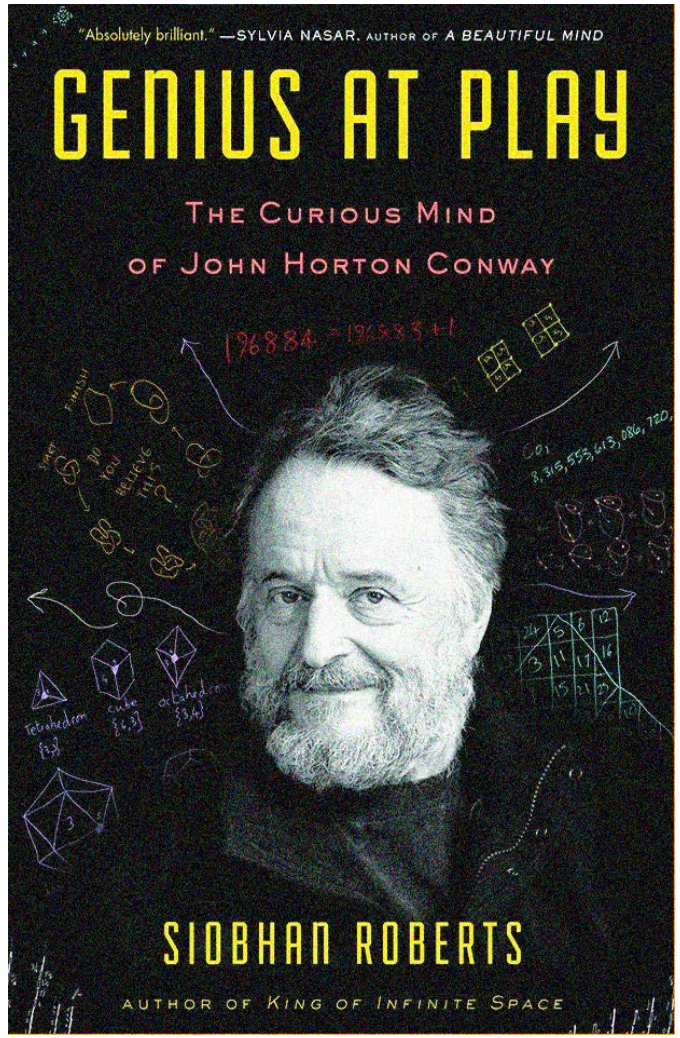


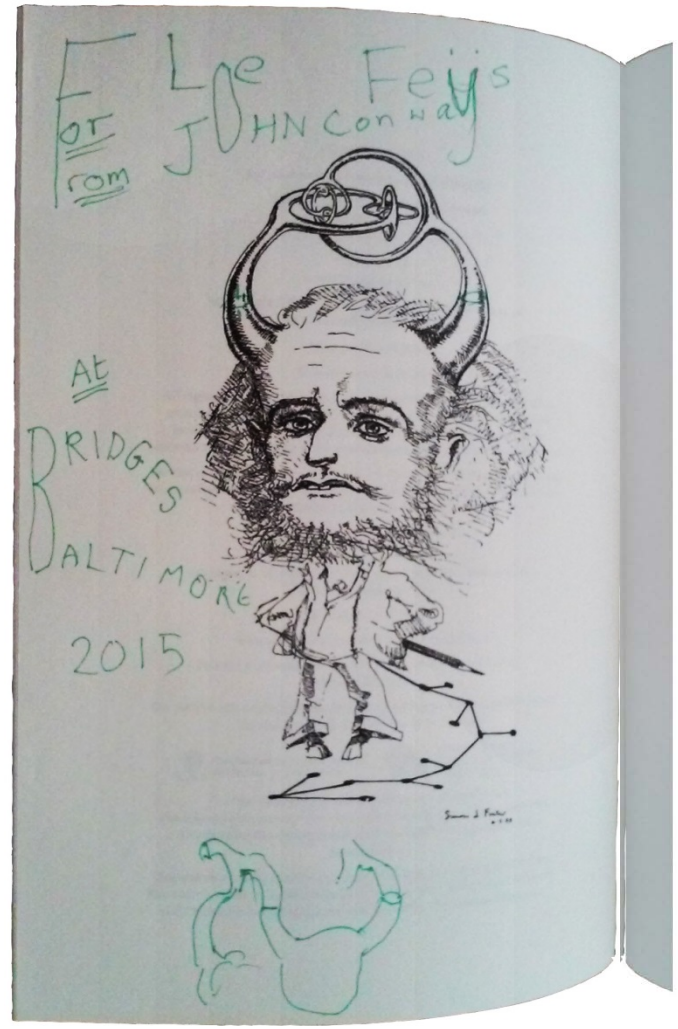
A stops the gun and B can activate the output. The result is True.

Source: Jean Philippe Rennard, Implementation of Logical functions in the game of life, Collision-based computing, 2002 - Springer, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.9114&rep=rep1&type=pdf

Conway’s game of life: conclusions

- There is a large number of life forms, many still to be discovered
- Boolean gates can be simulated
- Even a universal computer can be simulated (Turing machine)

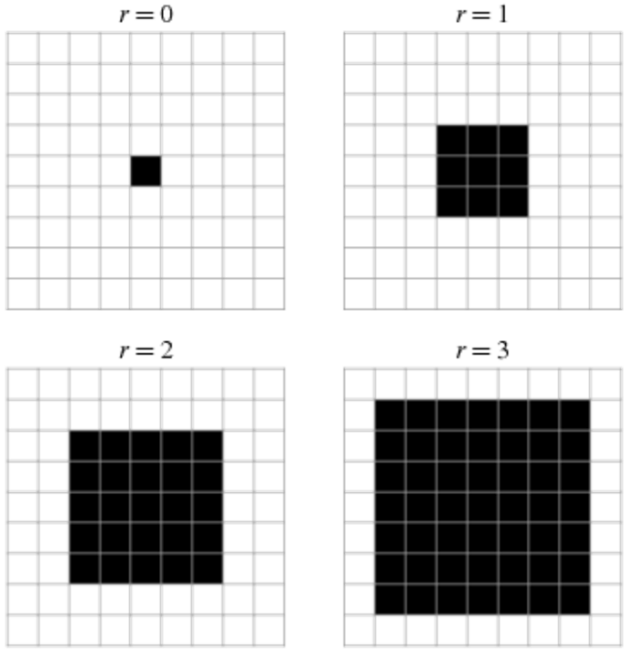




Meeting John Conway at Bridges 2015

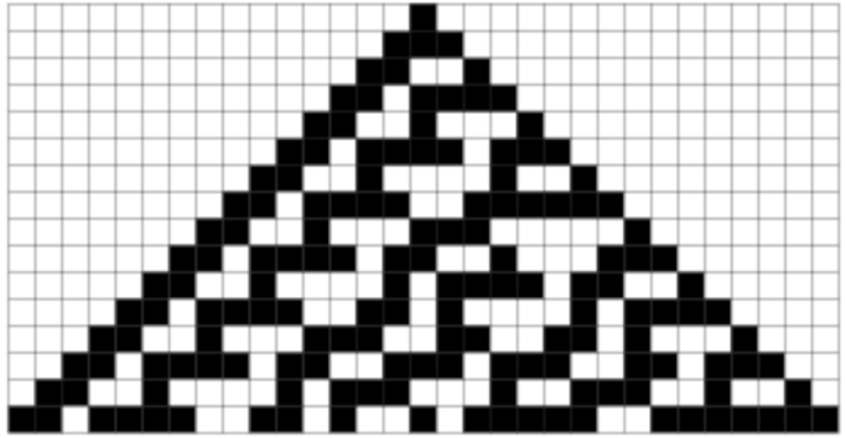
Towards elementary rules: one-dimension, r = 1

Mathworld: Moore neighbourhood (in two dimensions)



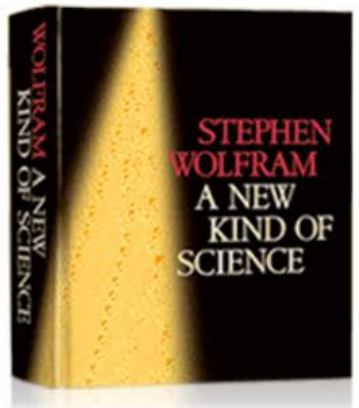
rule 30

r=1



https://www.wolframscience.com/

Now available for iPad
Download now »



Also available in the classic hardcover edition with art-quality printing »

Stephen Wolfram's
A NEW KIND OF SCIENCE | ONLINE

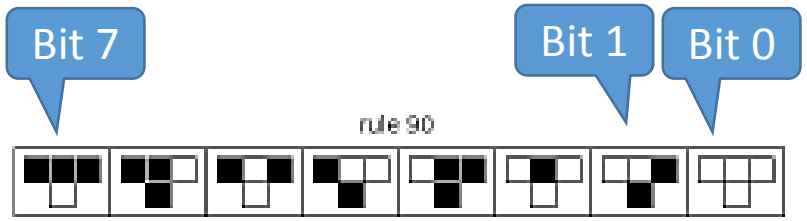
Jump to Page | Look Up in Index | Search

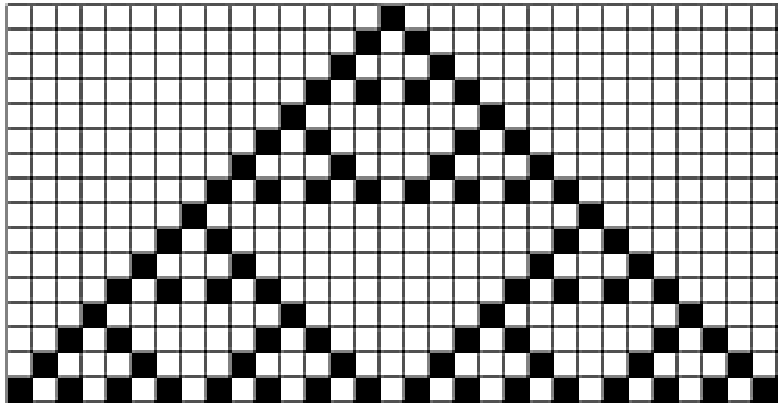
	Preface	ix
CHAPTER 1	The Foundations for a New Kind of Science	1
CHAPTER 2	The Crucial Experiment	23
CHAPTER 3	The World of Simple Programs	51
CHAPTER 4	Systems Based on Numbers	115
CHAPTER 5	Two Dimensions and Beyond	169
CHAPTER 6	Starting from Randomness	223
CHAPTER 7	Mechanisms in Programs and Nature	297

SETI INSTITUTE
SCIENCE COLLOQUIUM
Astronomy & Astrophysics - Astrobiology - Climate & Geoscience - Exoplanets - Planetary Exploration - SETI
Microsoft

Enumeration scheme for two-color r=1 one-dimensional automata

Example 01011010 (binary) = 64 + 16 + 8 + 2 = 90

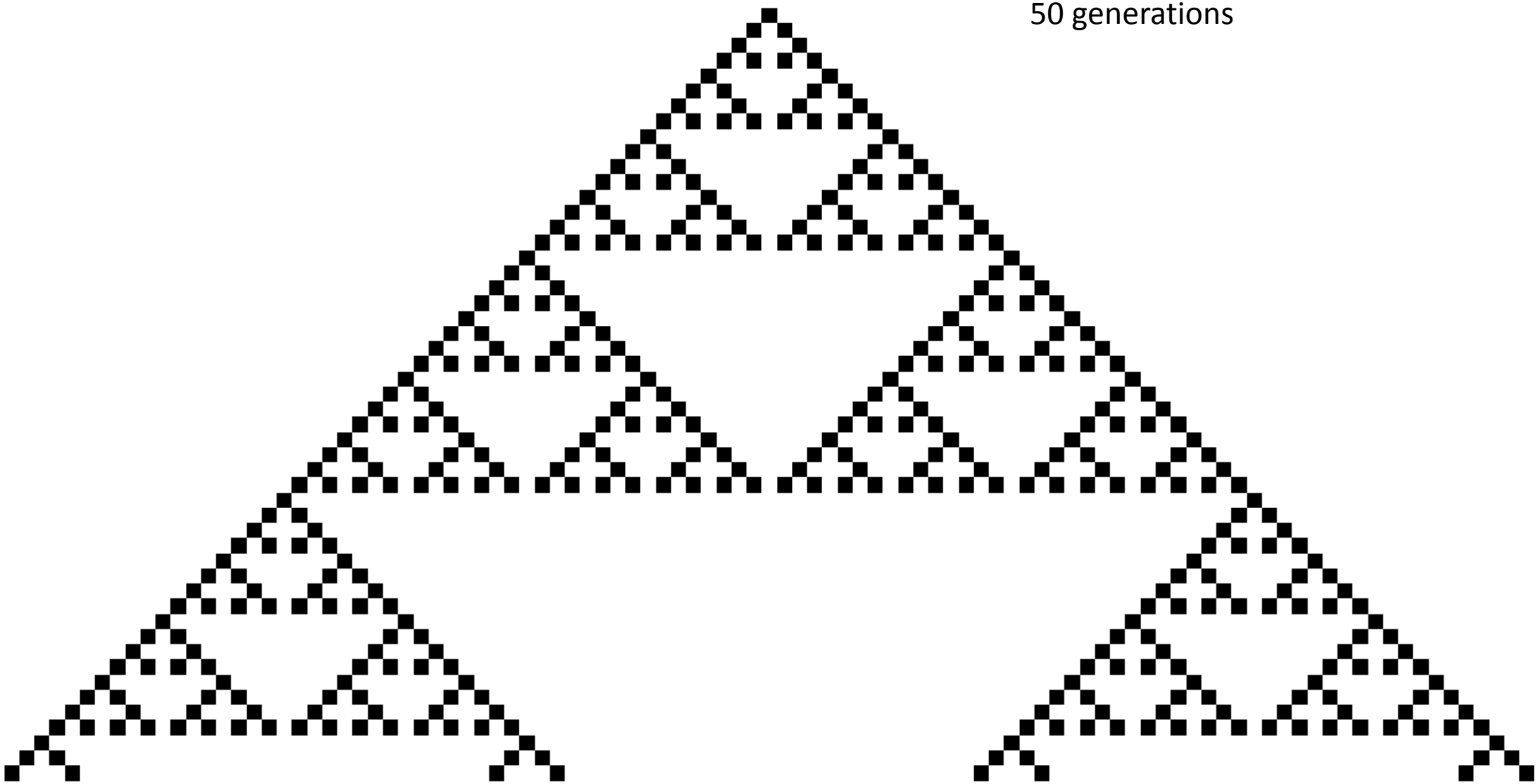




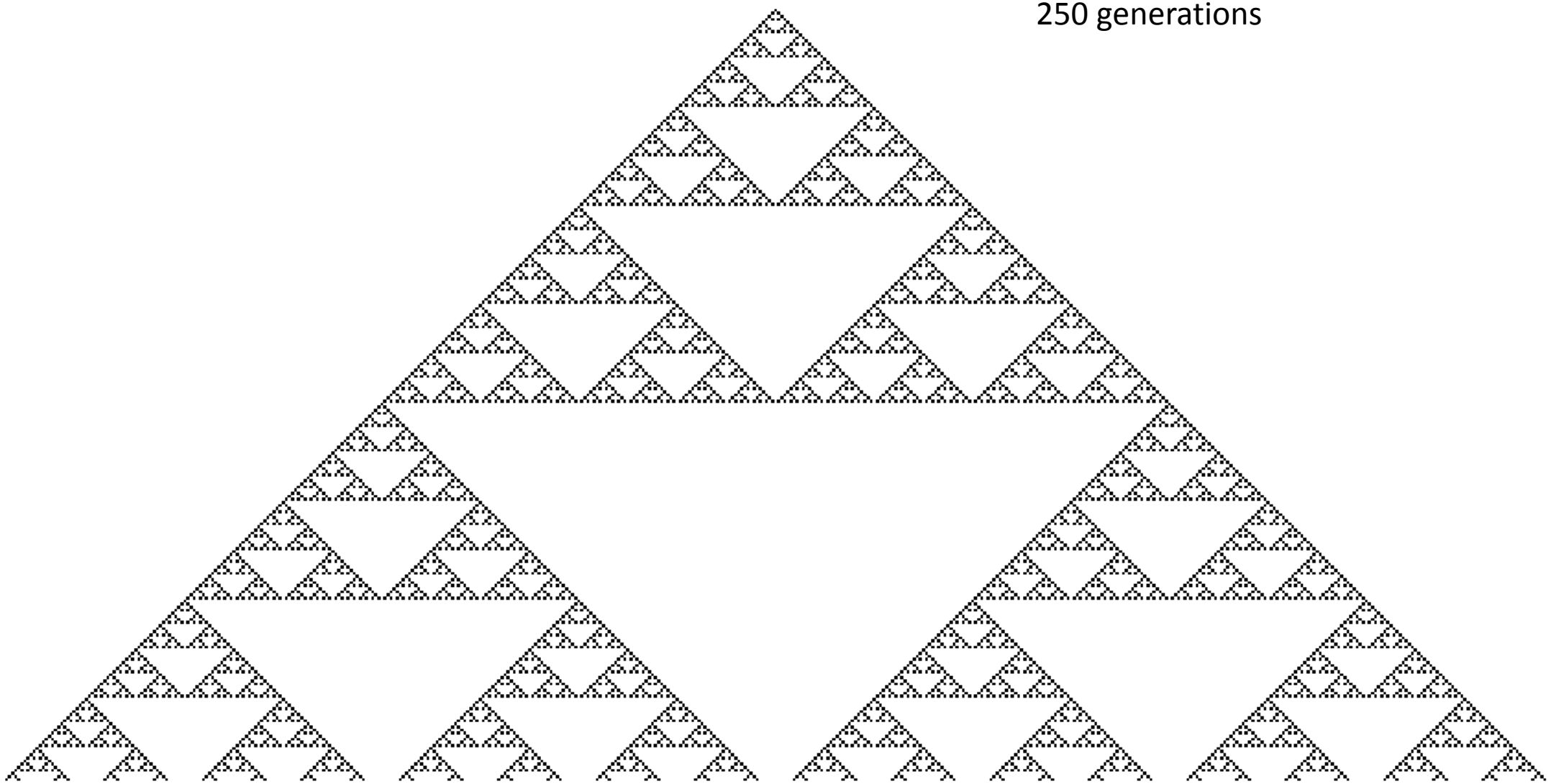
The 2D pattern formed by this rule has fractal dimension log 3 / log 2 ≈ 1.58

Source: NKS

50 generations



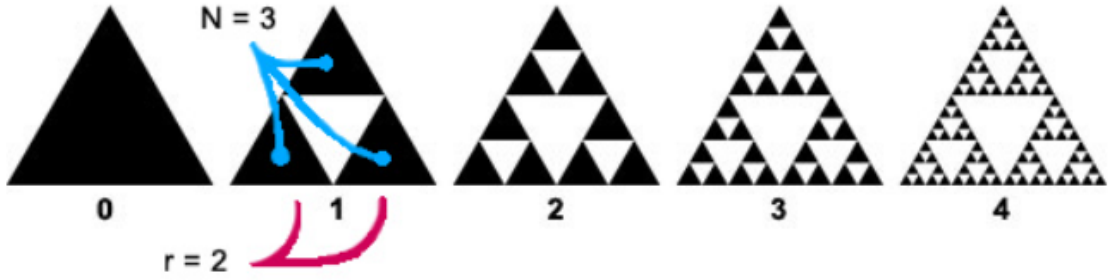
250 generations



This is a so-=called Sierpinski triangle, see https://en.wikipedia.org/wiki/Sierpinski_triangle

Fractal Dimension of the Sierpinski Triangle

Let's use the formula for scaling to determine the dimension of the Sierpinski Triangle fractal. First, take a rough guess at what you might think the dimension will be. Less than 1? Between 1 and 2? Greater than 2? Since the Sierpinski Triangle fits in plane but doesn't fill it completely, its dimension should be less than 2. Let's see if this is true.

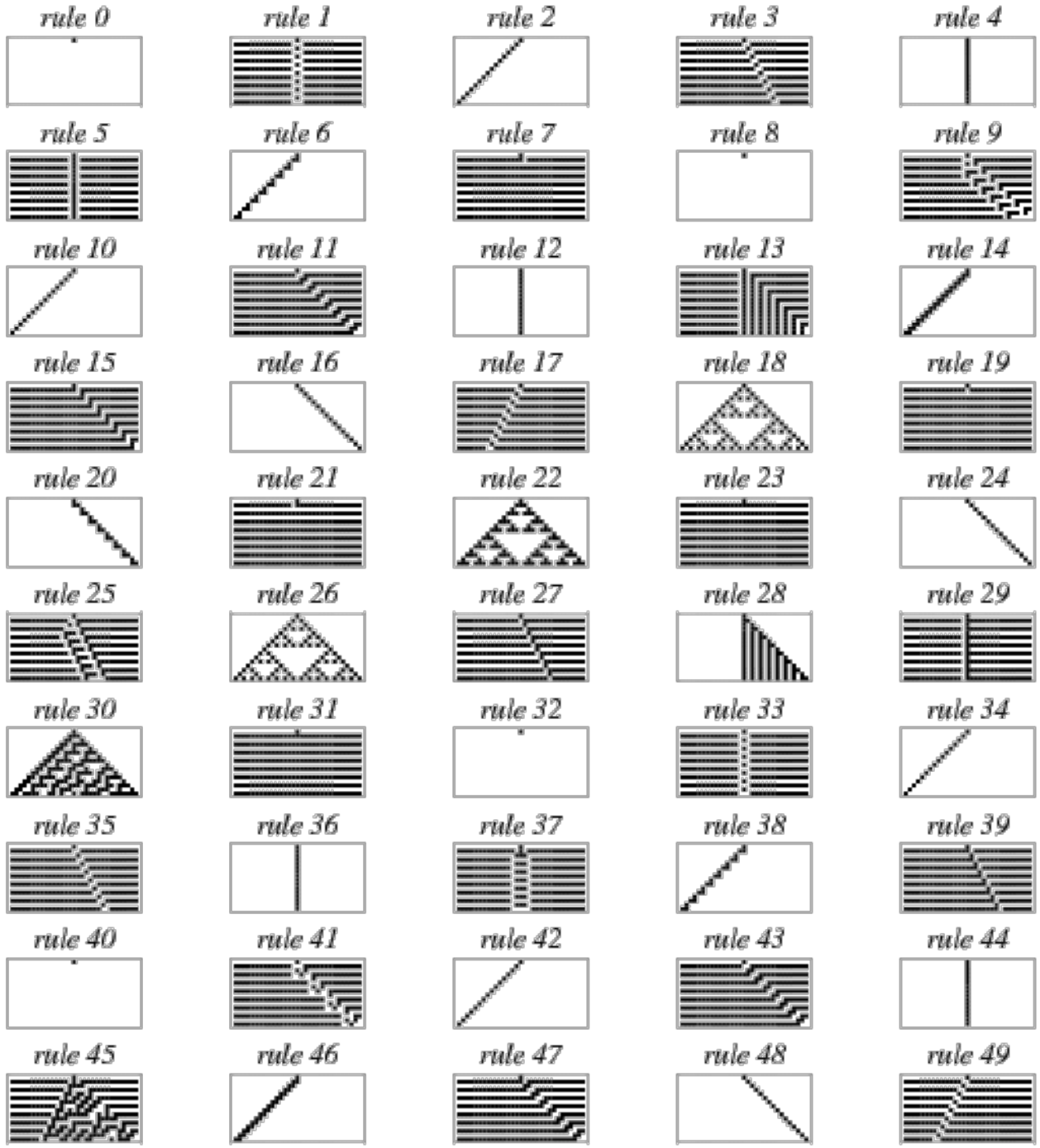


Start with the 0 order triangle in the figure above. The next iteration, order 1, is made up of 3 smaller triangles. And order 2 is made up of 9 triangles. So each iteration of the fractal has 3 times as many triangles, and **N=3**. Next we need to figure out the scaling factor, **r**. How much smaller is each triangle in order 1 than order 0? Look at the edge of each triangle in order 1, and you can see that the edge of each triangle is half the length of the edge of the triangle in order 0. So the scaling factor **r=2**.

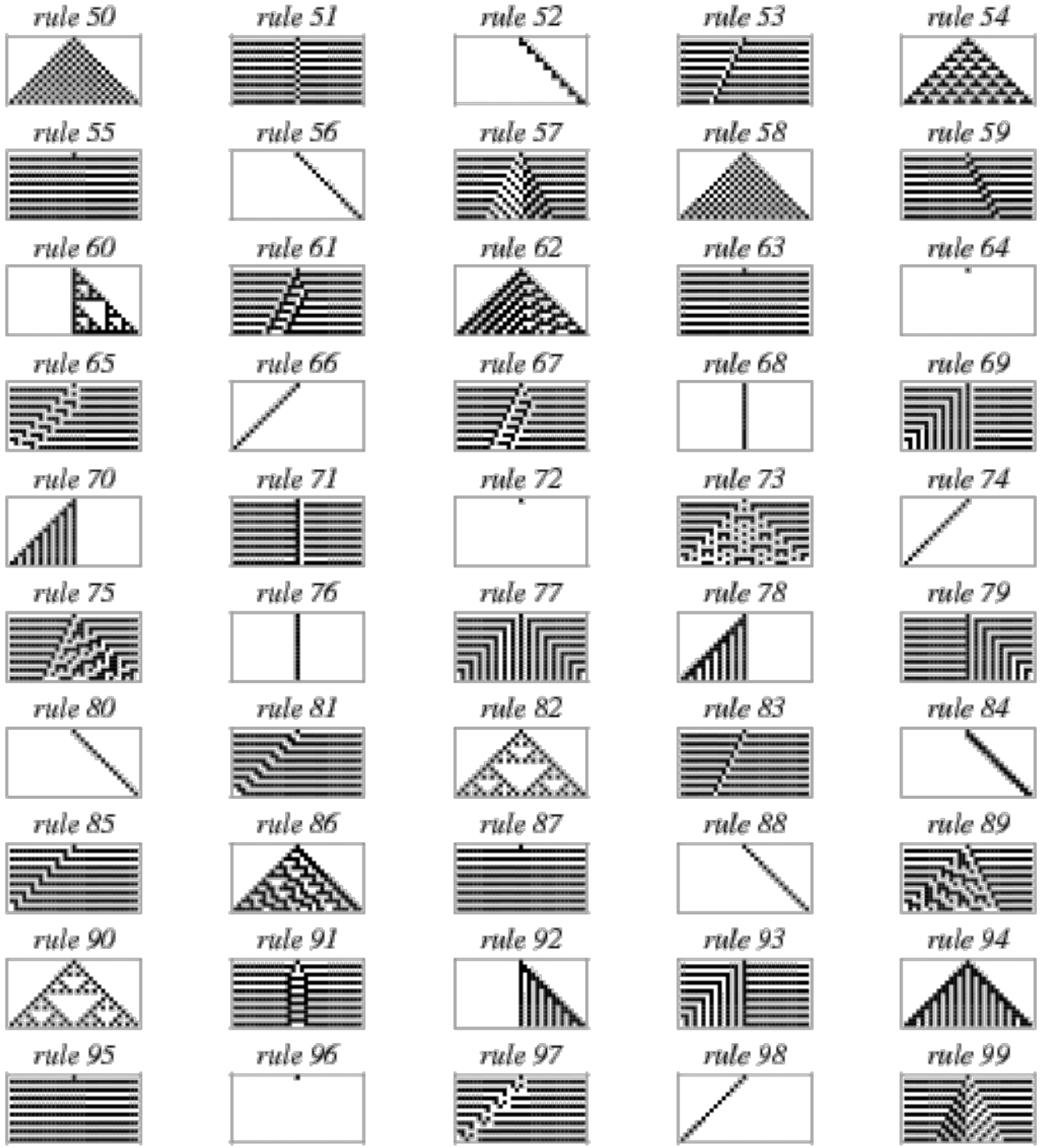
That's all we need to know, and we can find the dimension by using the formula:

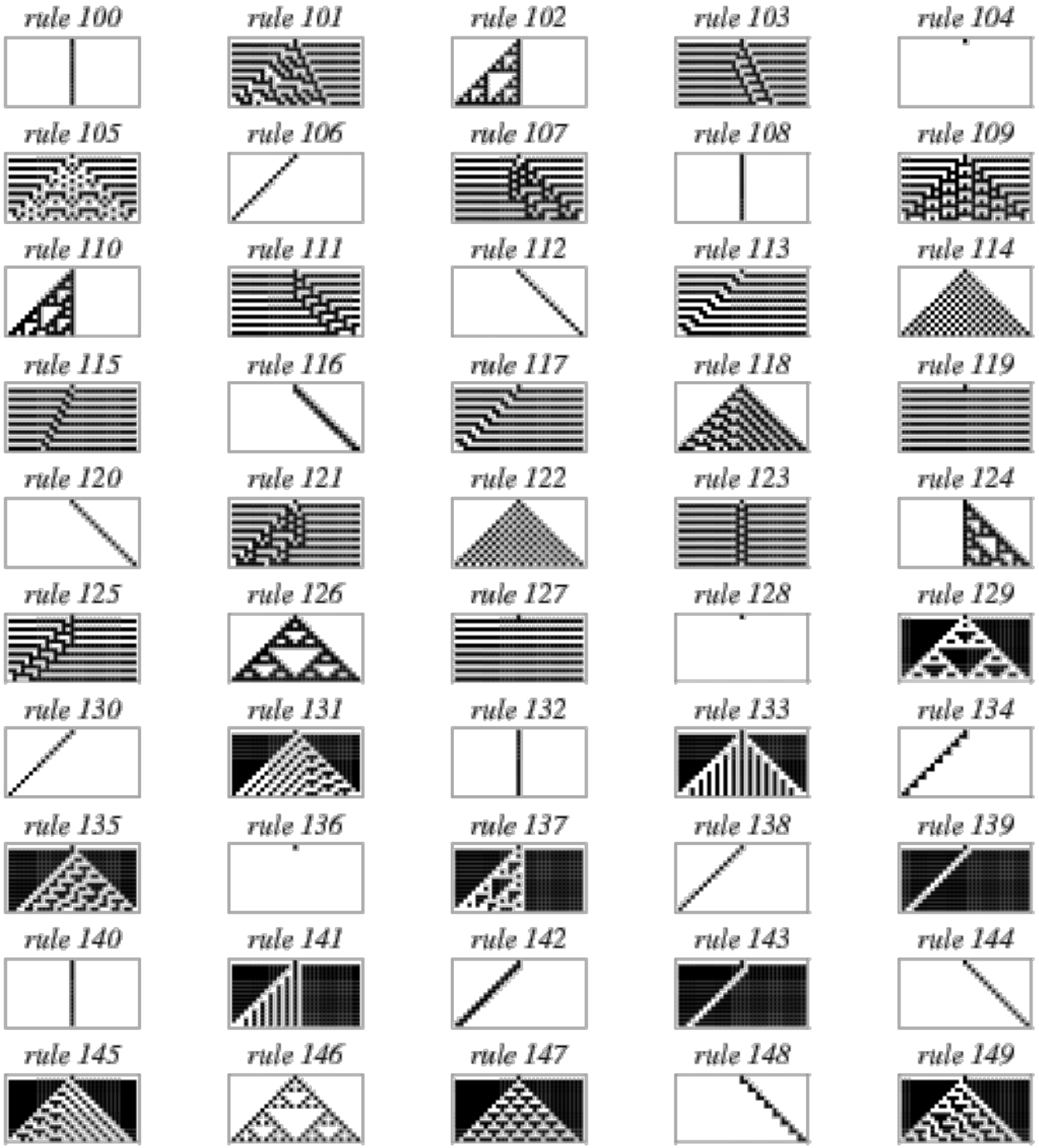
$$D = \frac{\log(N)}{\log(r)} = \frac{\log(3)}{\log(2)} = 1.585$$

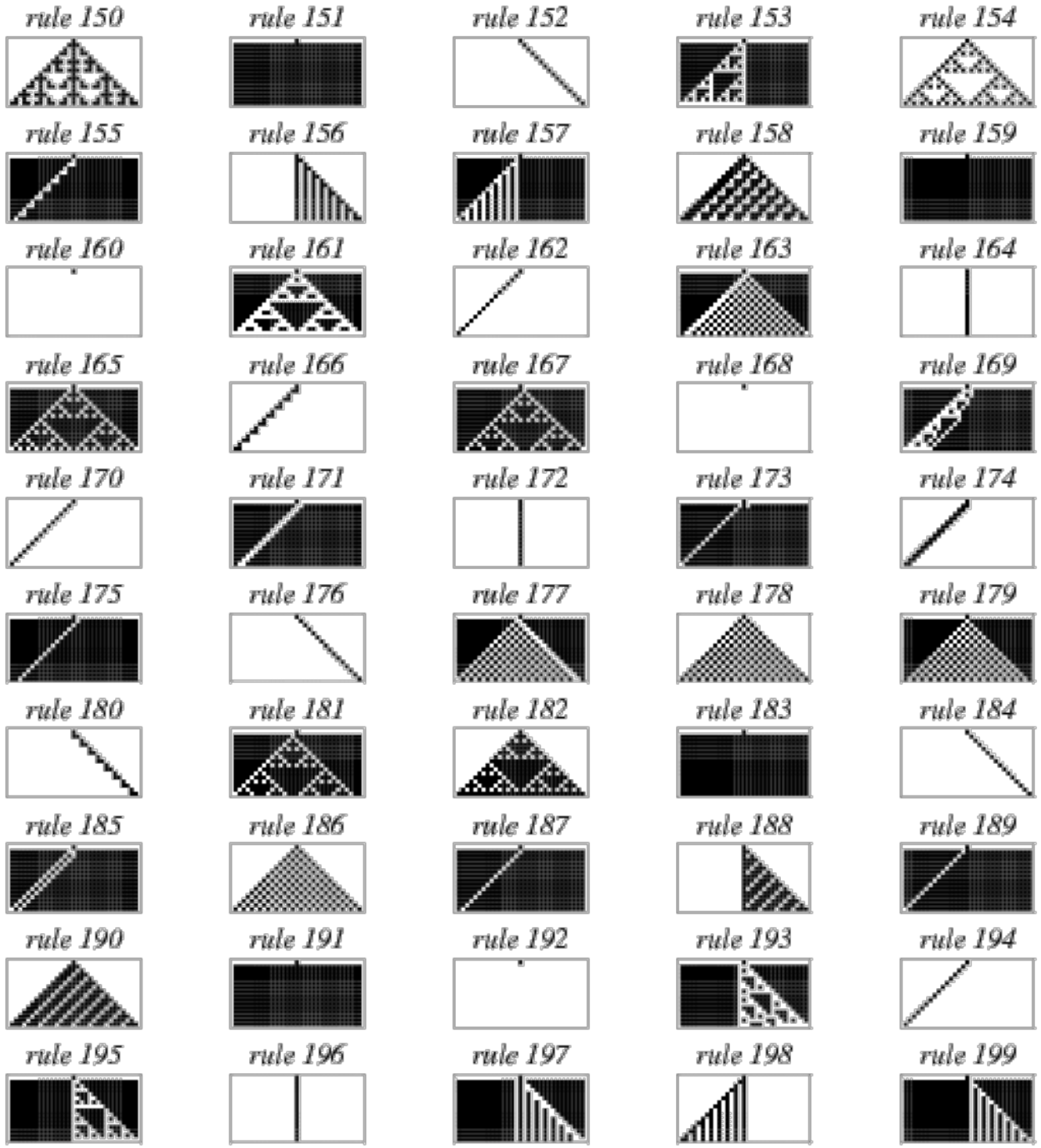
http://fractalfoundation.org/OFC/OFC-10-3.html

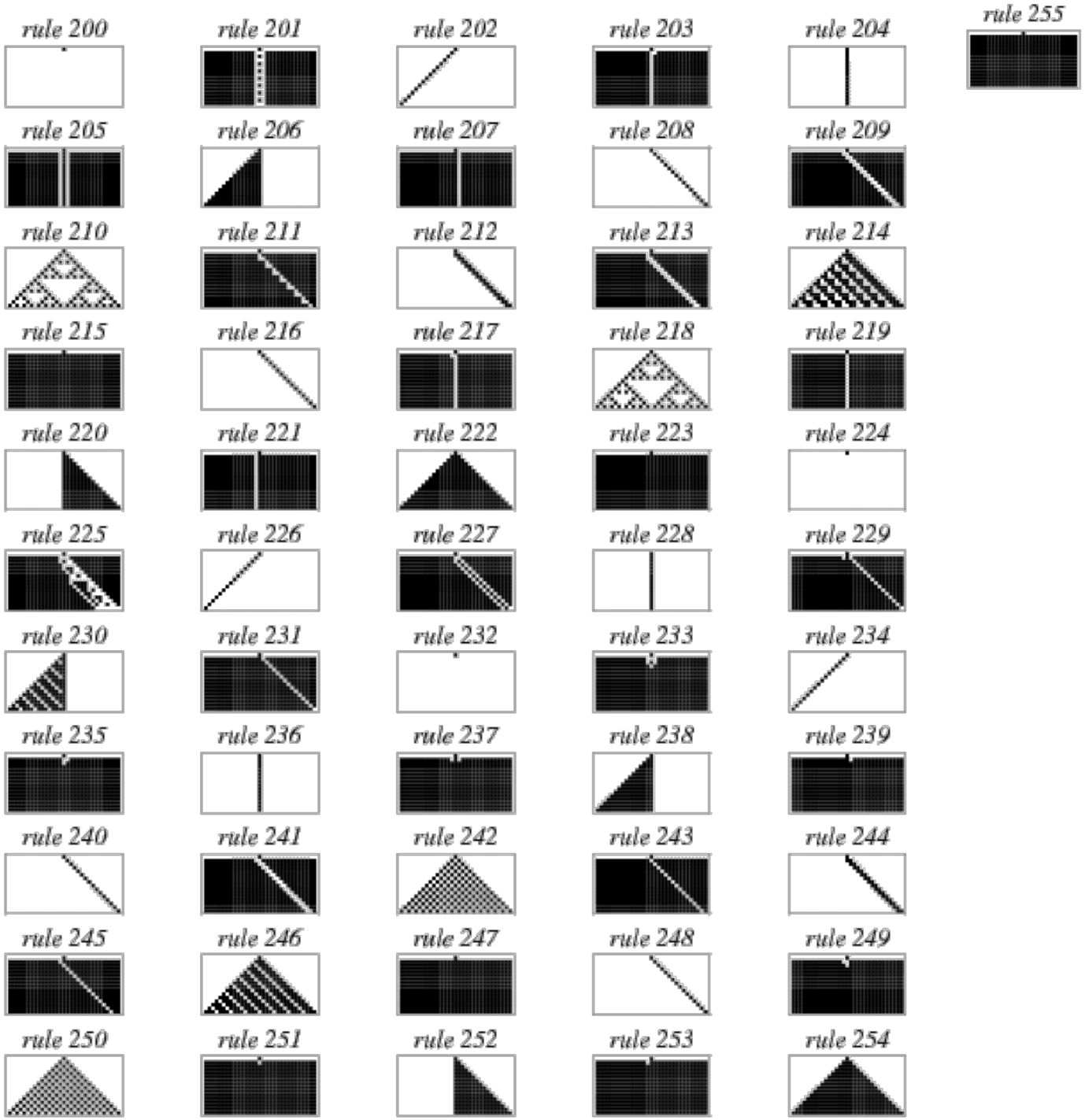

rule 0
rule 1
rule 2
rule 3
rule 4
rule 5
rule 6
rule 7
rule 8
rule 9
rule 10
rule 11
rule 12
rule 13
rule 14
rule 15
rule 16
rule 17
rule 18
rule 19
rule 20
rule 21
rule 22
rule 23
rule 24
rule 25
rule 26
rule 27
rule 28
rule 29
rule 30
rule 31
rule 32
rule 33
rule 34
rule 35
rule 36
rule 37
rule 38
rule 39
rule 40
rule 41
rule 42
rule 43
rule 44
rule 45
rule 46
rule 47
rule 48
rule 49

Source: NKS

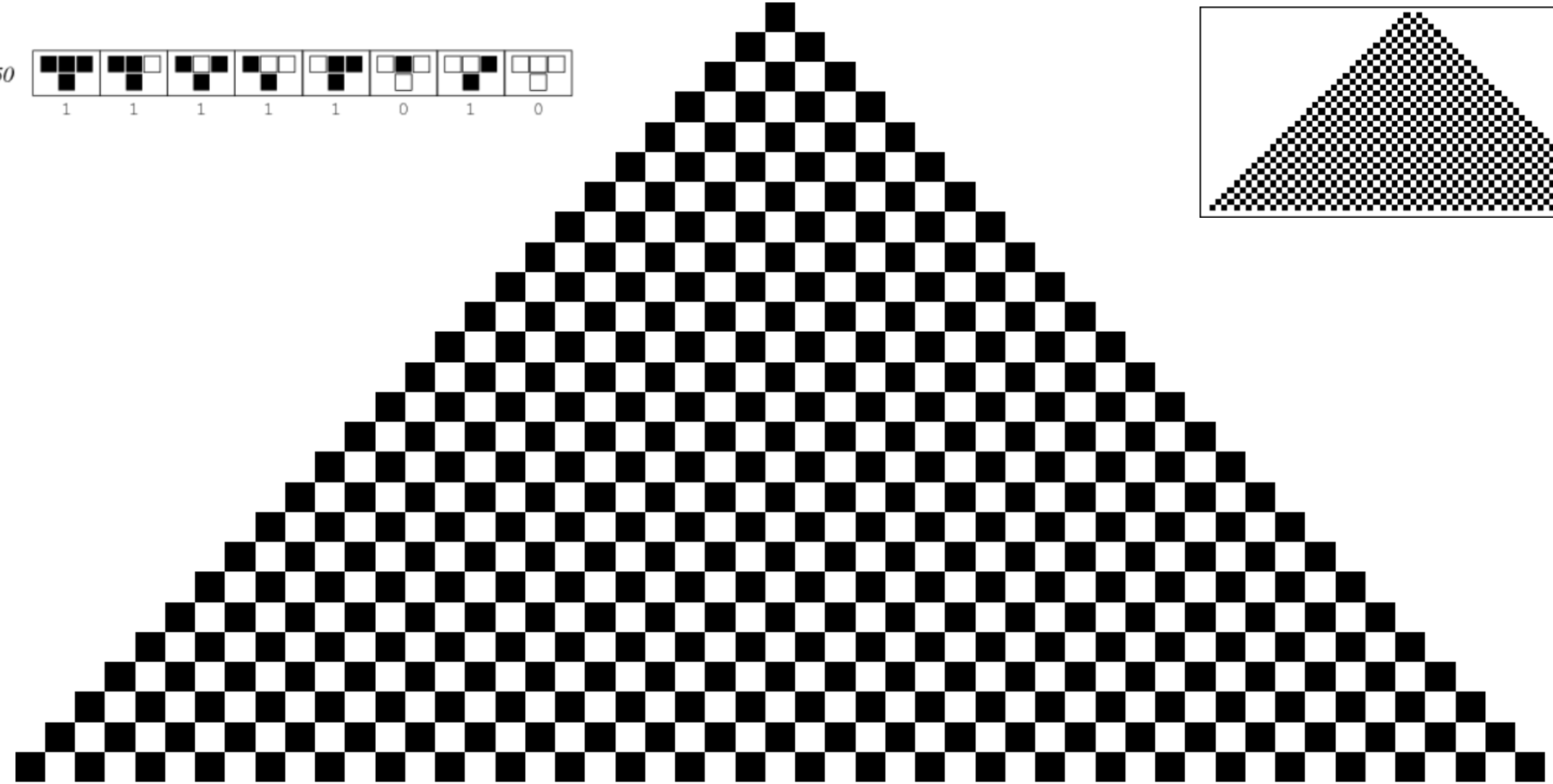

rule 50
rule 51
rule 52
rule 53
rule 54
rule 55
rule 56
rule 57
rule 58
rule 59
rule 60
rule 61
rule 62
rule 63
rule 64
rule 65
rule 66
rule 67
rule 68
rule 69
rule 70
rule 71
rule 72
rule 73
rule 74
rule 75
rule 76
rule 77
rule 78
rule 79
rule 80
rule 81
rule 82
rule 83
rule 84
rule 85
rule 86
rule 87
rule 88
rule 89
rule 90
rule 91
rule 92
rule 93
rule 94
rule 95
rule 96
rule 97
rule 98
rule 99

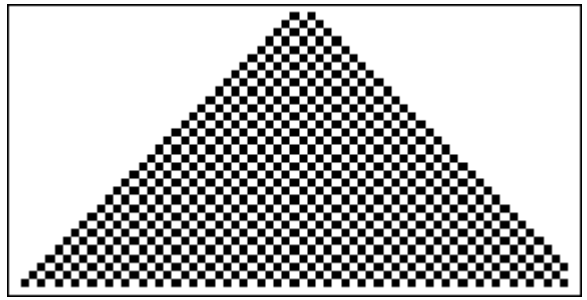

rule 100
rule 101
rule 102
rule 103
rule 104
rule 105
rule 106
rule 107
rule 108
rule 109
rule 110
rule 111
rule 112
rule 113
rule 114
rule 115
rule 116
rule 117
rule 118
rule 119
rule 120
rule 121
rule 122
rule 123
rule 124
rule 125
rule 126
rule 127
rule 128
rule 129
rule 130
rule 131
rule 132
rule 133
rule 134
rule 135
rule 136
rule 137
rule 138
rule 139
rule 140
rule 141
rule 142
rule 143
rule 144
rule 145
rule 146
rule 147
rule 148
rule 149


rule 150
rule 151
rule 152
rule 153
rule 154
rule 155
rule 156
rule 157
rule 158
rule 159
rule 160
rule 161
rule 162
rule 163
rule 164
rule 165
rule 166
rule 167
rule 168
rule 169
rule 170
rule 171
rule 172
rule 173
rule 174
rule 175
rule 176
rule 177
rule 178
rule 179
rule 180
rule 181
rule 182
rule 183
rule 184
rule 185
rule 186
rule 187
rule 188
rule 189
rule 190
rule 191
rule 192
rule 193
rule 194
rule 195
rule 196
rule 197
rule 198
rule 199

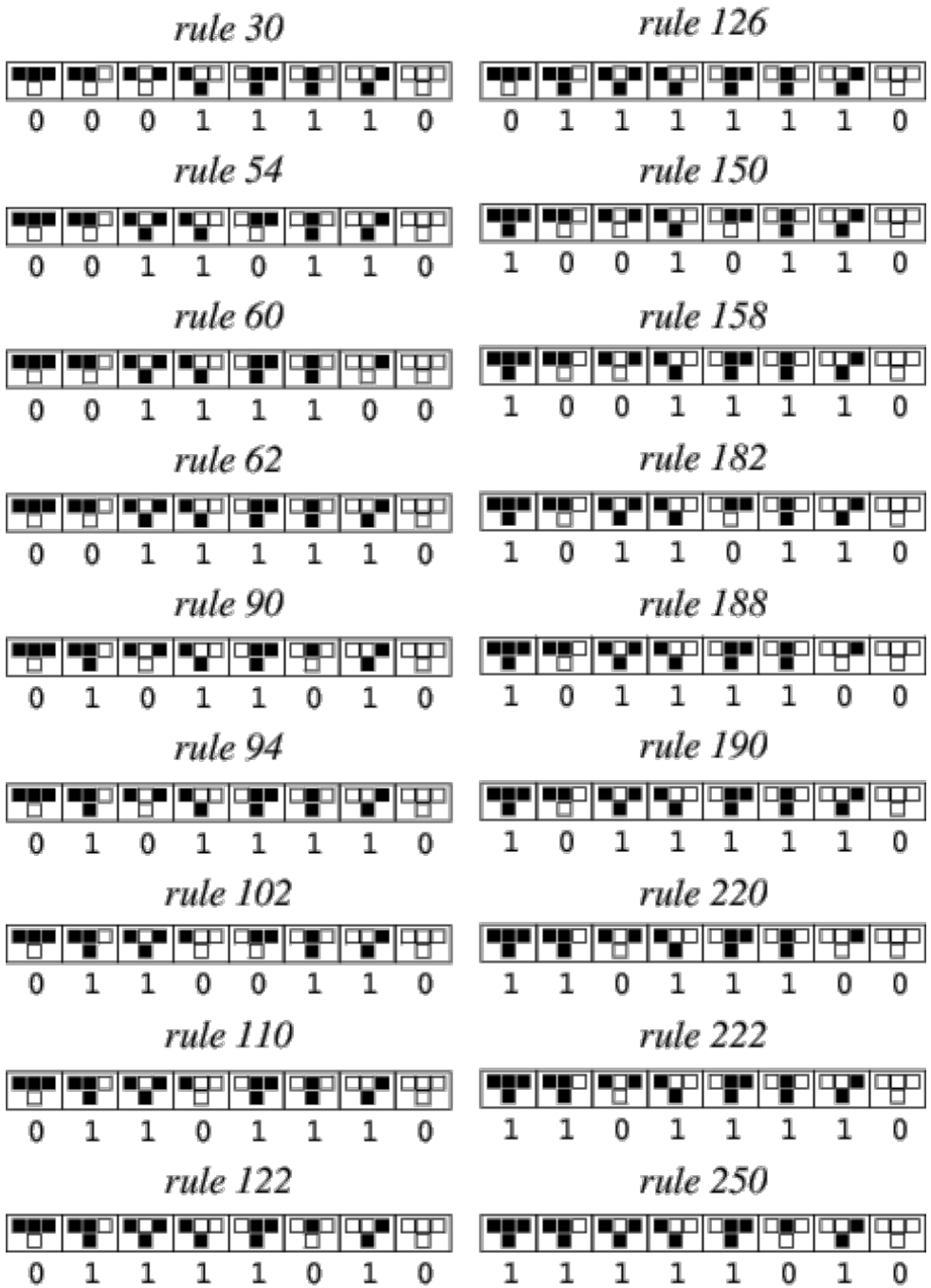

rule 200
rule 201
rule 202
rule 203
rule 204
rule 205
rule 206
rule 207
rule 208
rule 209
rule 210
rule 211
rule 212
rule 213
rule 214
rule 215
rule 216
rule 217
rule 218
rule 219
rule 220
rule 221
rule 222
rule 223
rule 224
rule 225
rule 226
rule 227
rule 228
rule 229
rule 230
rule 231
rule 232
rule 233
rule 234
rule 235
rule 236
rule 237
rule 238
rule 239
rule 240
rule 241
rule 242
rule 243
rule 244
rule 245
rule 246
rule 247
rule 248
rule 249
rule 250
rule 251
rule 252
rule 253
rule 254

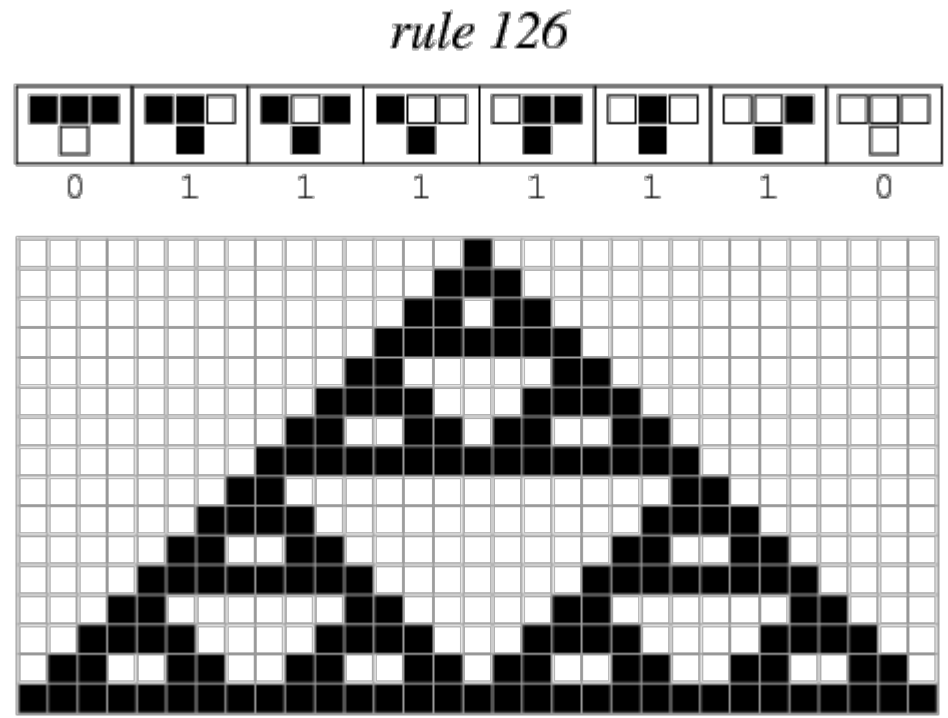
rule 255





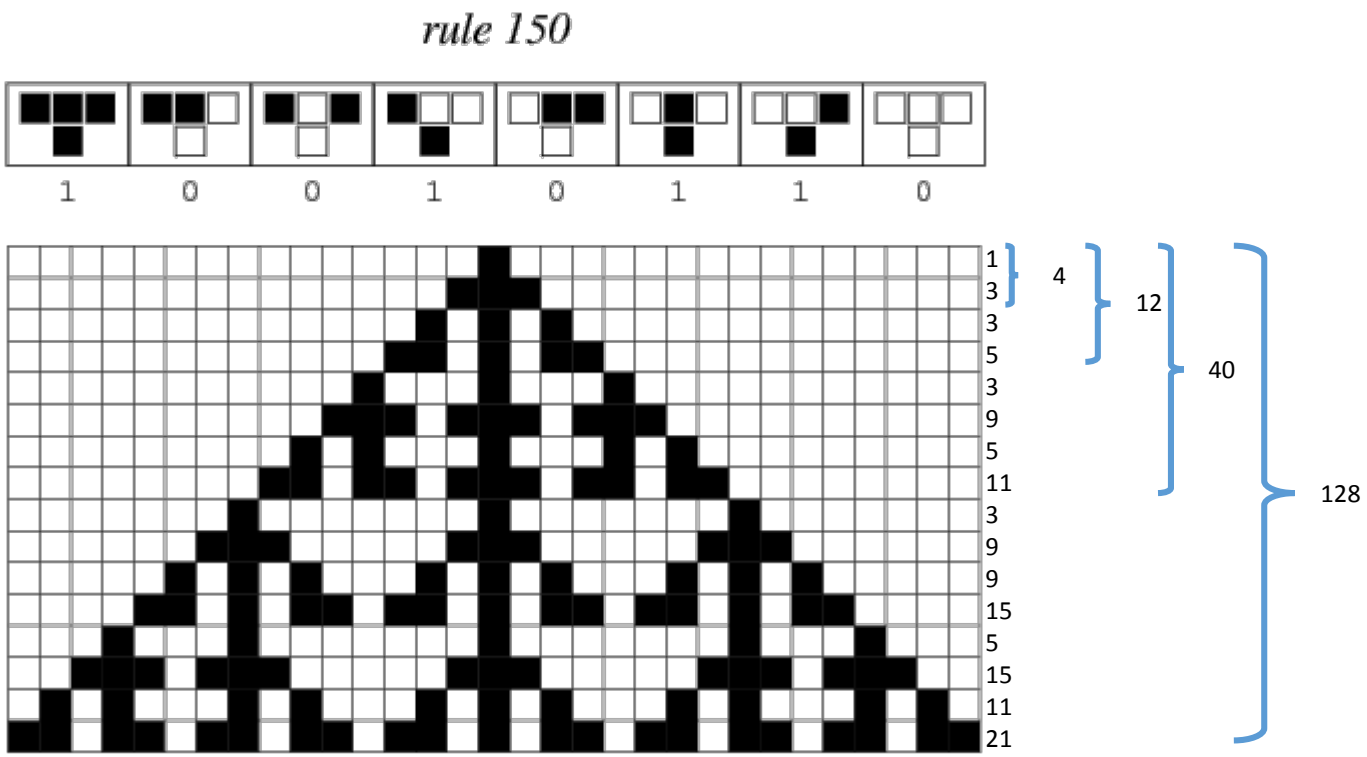
Rule 250 This one is additive, see
http://mathworld.wolfram.com/AdditiveCellularAutomaton.html





Also Rule 90 is “additive”, check Mathworld: http://mathworld.wolfram.com/AdditiveCellularAutomaton.html

source: NKS



D=1 would give 2, 4, 8, 16, …

D=2 would give 2×2, 4×4, 8×8, 16×16, …

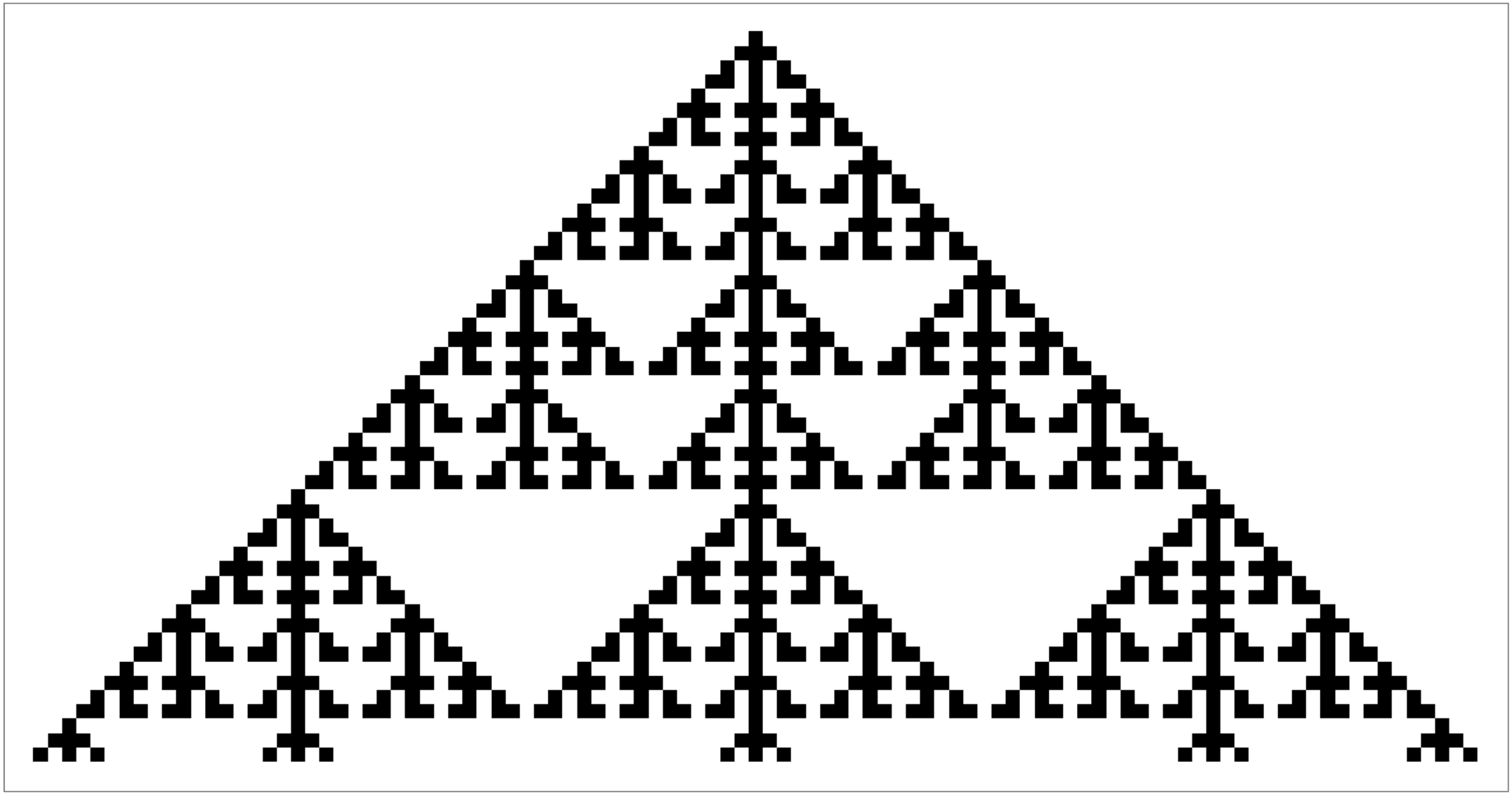
Now we have 2×2, 3×4, 5×8, 8x16, …

Fibonacci numbers: 2,3,5,8,13,21,34, etc.

Limit $F_{n+1}/F_n = (1+\sqrt{5})/2$

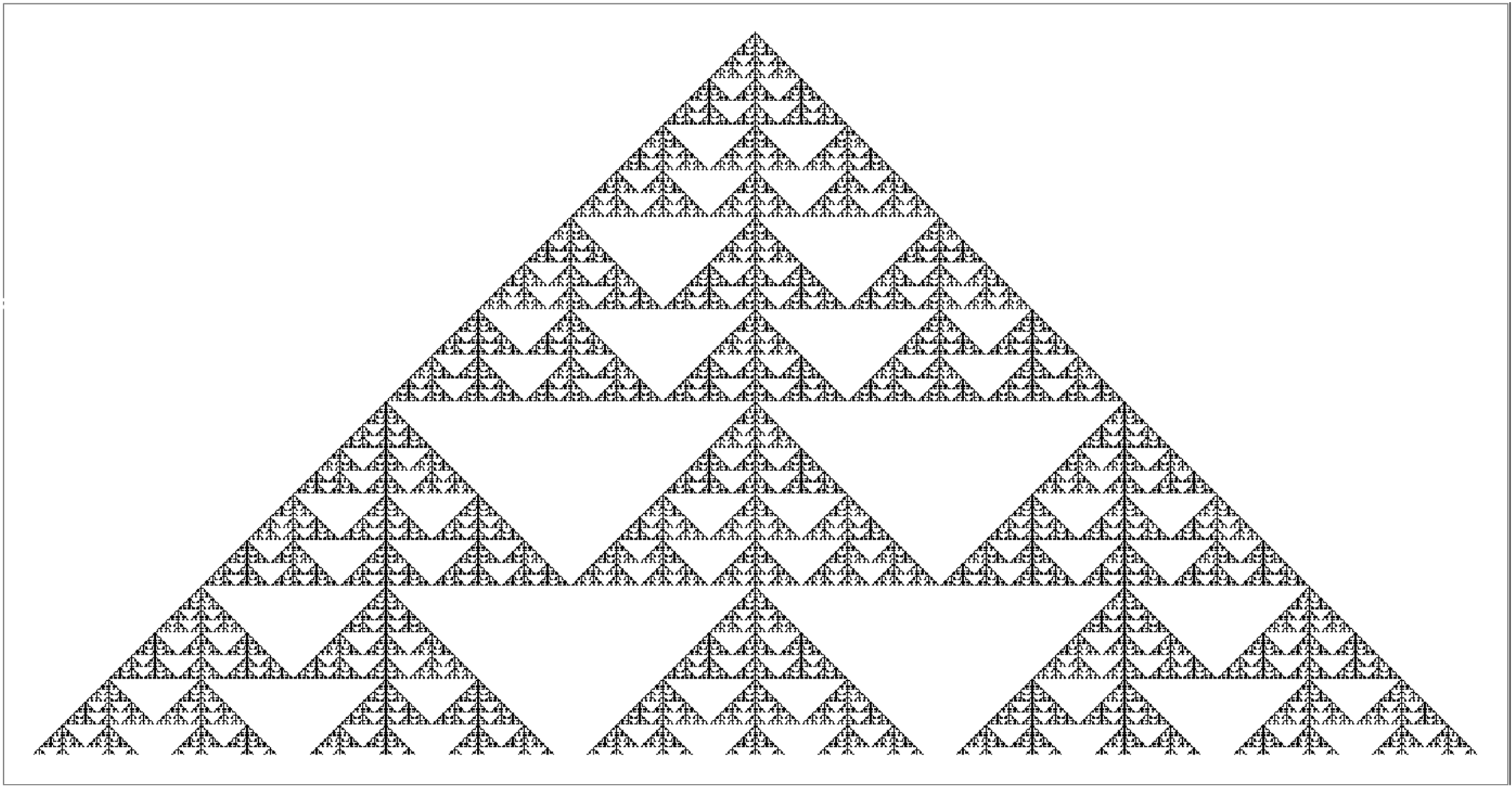
This rule has fractal dimension $\log(1+\sqrt{5}) / \log 2 \approx 1.69$

In[81]:=
```
M = 50;
c = CellularAutomaton[150, {{1}, 0}, M];
ArrayPlot[c]
```

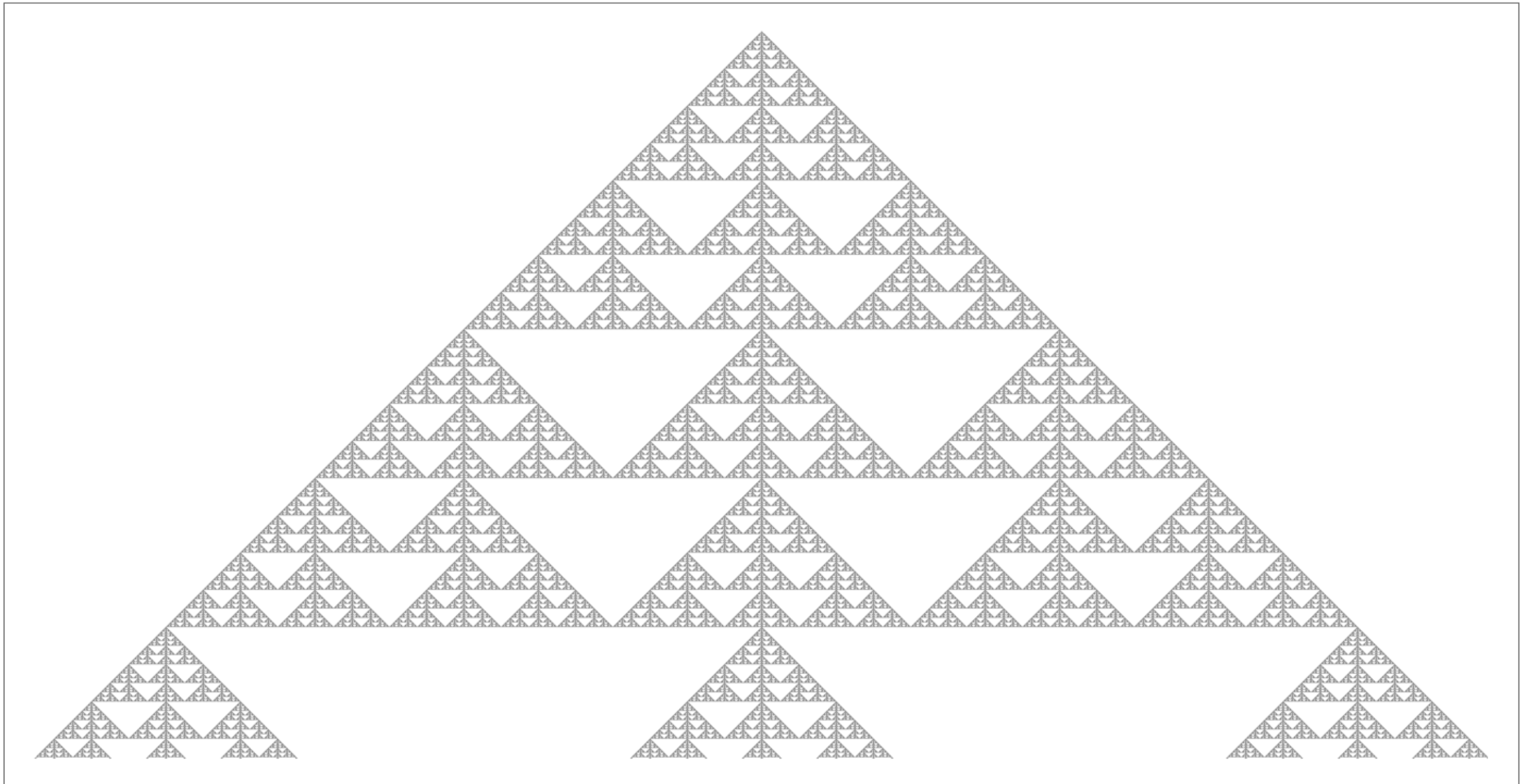


In[78]:=
```
M = 500;
c = CellularAutomaton[150, {{1}, 0}, M];
ArrayPlot[c]
```

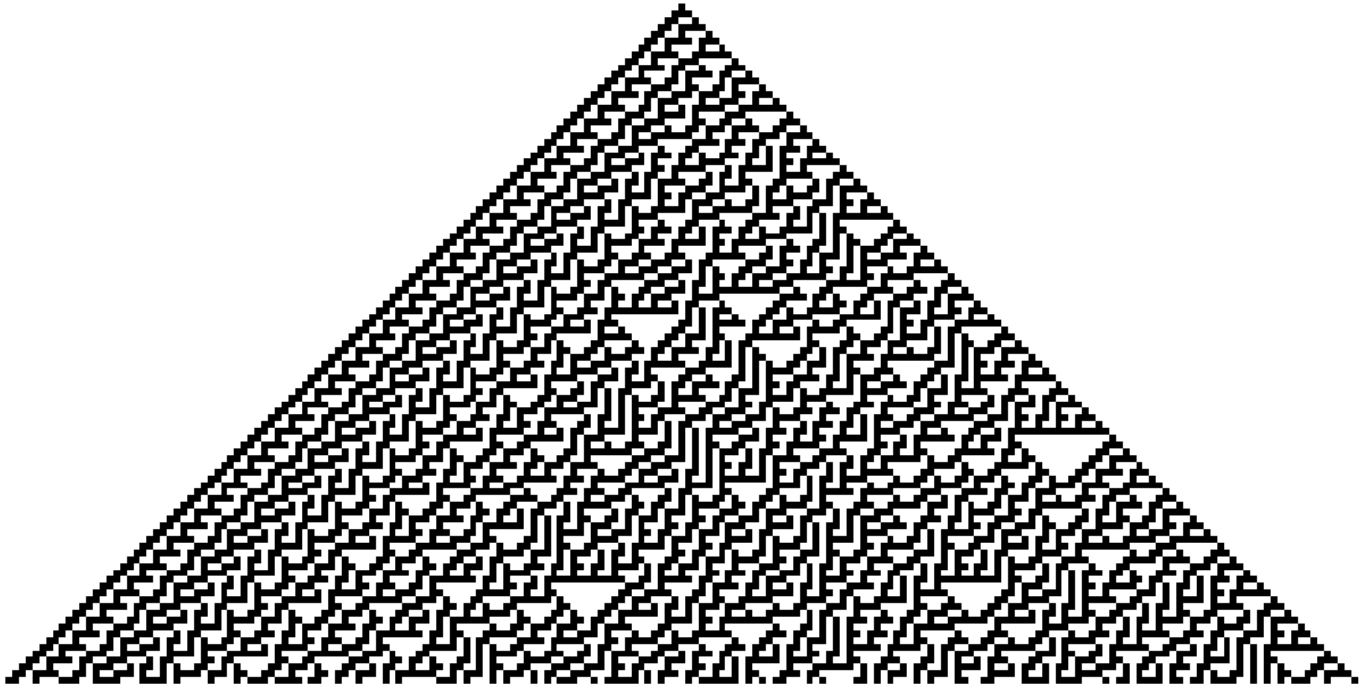
Out[80]=



In[75]:=
```
M = 5000;
c = CellularAutomaton[150, {{1}, 0}, M];
ArrayPlot[c]
```



Rule 30 is chaotic, like a random generator

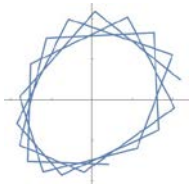


Conus textile by Richard Ling, Wikimedia

An empirical observation is that cellular automata can be classied according to the complexity and information produced by the behavior of the pattern they produce:

- Class 1 : Fixed; all cells converge to a constant black or white set
- Class 2 : Periodic; repeats the same pattern, like a loop
- Class 3 : Chaotic; pseudo-random
- Class 4 : Complex local structures; exhibits behaviors of both class 2 and class 3; with long lived hard to classify structure.

Source: http://www2.math.uu.se/~david/MCS/lecture3.pdf by David_Sumpter

Wolfram’s one-dimensional automata: conclusions

- Some of the automata have complex behavior
- Some rules are perfect random generators (automaton rule 30)
- Even a universal computer can be simulated (automaton rule 110)
- Wolfram believes our whole universe could be just one big cellular automaton