Lecture 4: Introduction to Cellular Automata

Complex Systems 530

What is a cellular automaton?

- **Automata**: “a theoretical machine that changes its internal state based on inputs and its previous state” (usually finite and discrete) - Sayama p.185

- **Cellular automata**: automata on a regular spatial grid, that update state based on their neighbors’ states, using a **state transition function**

- Usually synchronous, discrete in time & space, often deterministic (but not always!)

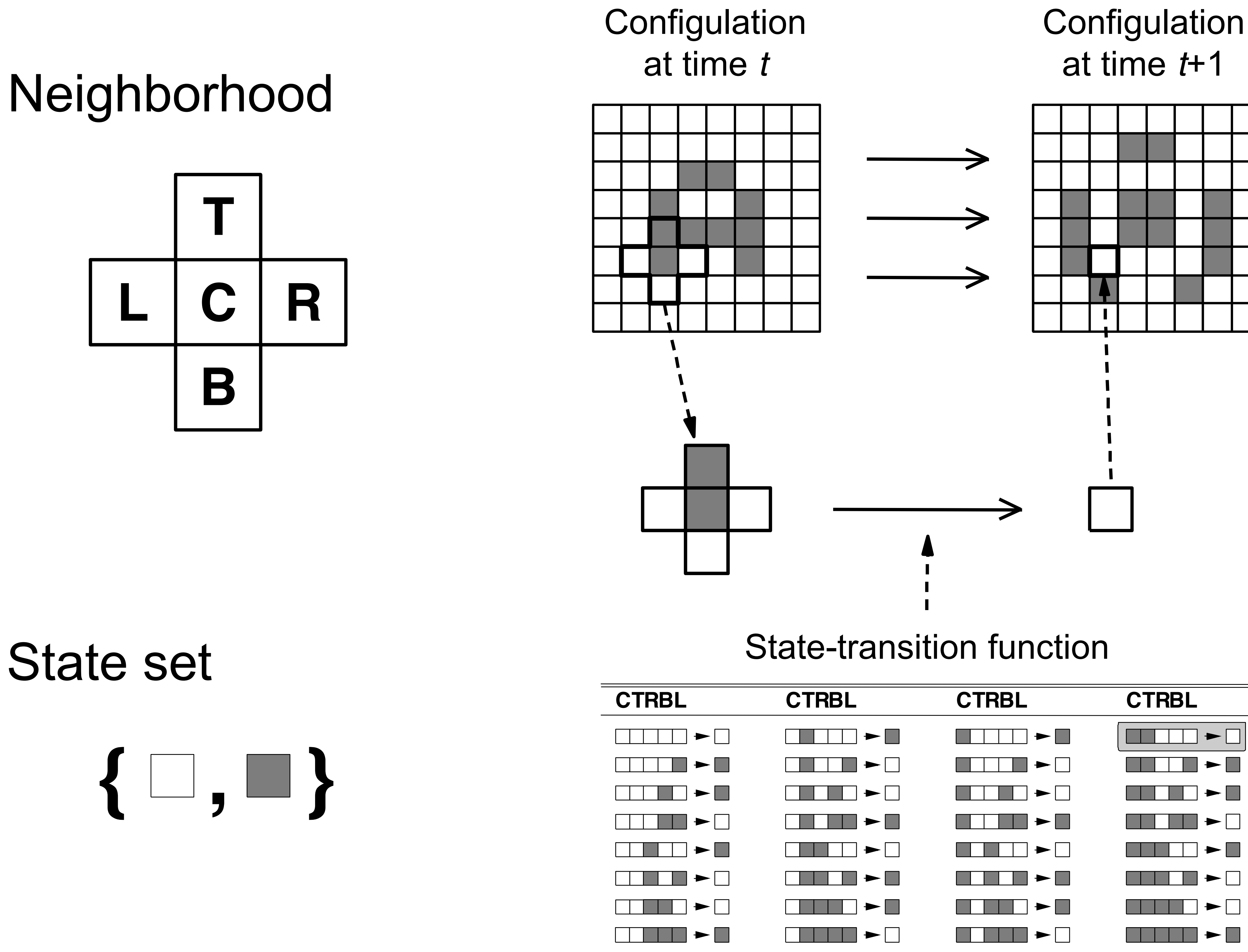


Figure 11.1: Schematic illustration of how cellular automata work.

Cellular automata

- Cellular automata can generate highly nonlinear, even seemingly random behavior

- Much more complexity than one might expect from simple rules—**emergent behavior**

- To explore this, let’s start with an even ‘simpler’ type of cellular automata—1-dimensional CA and some of the classic work of Stephen Wolfram

1-dimensional CA

- We can think of our grid as a string or line of cells
  - **Finite sequence** - 1 row of cells, so everyone has 2 neighbors except the end points
    - Choose how to interpret the ends (lack of neighbors or fixed states at ends)
  - **Ring** - all cells have 2 neighbors
  - **Infinite sequence** - an infinite number of cells arranged in a row

Finite sequence 1D CA

- Start with a 3-cell neighborhood (left, self, right)
- We can fully specify our CA by listing all the possible neighborhood configurations and saying what happens to the center cell, for example:

prev	111	110	101	100	011	010	001	000
next	0	0	1	1	0	0	1	0

- We can name our CA by translating the “next” row from binary to decimal: this is Rule 50!
(256 total possible CAs of this type)

Downey, *Think Complexity* (Chp. 6)

Rule 50

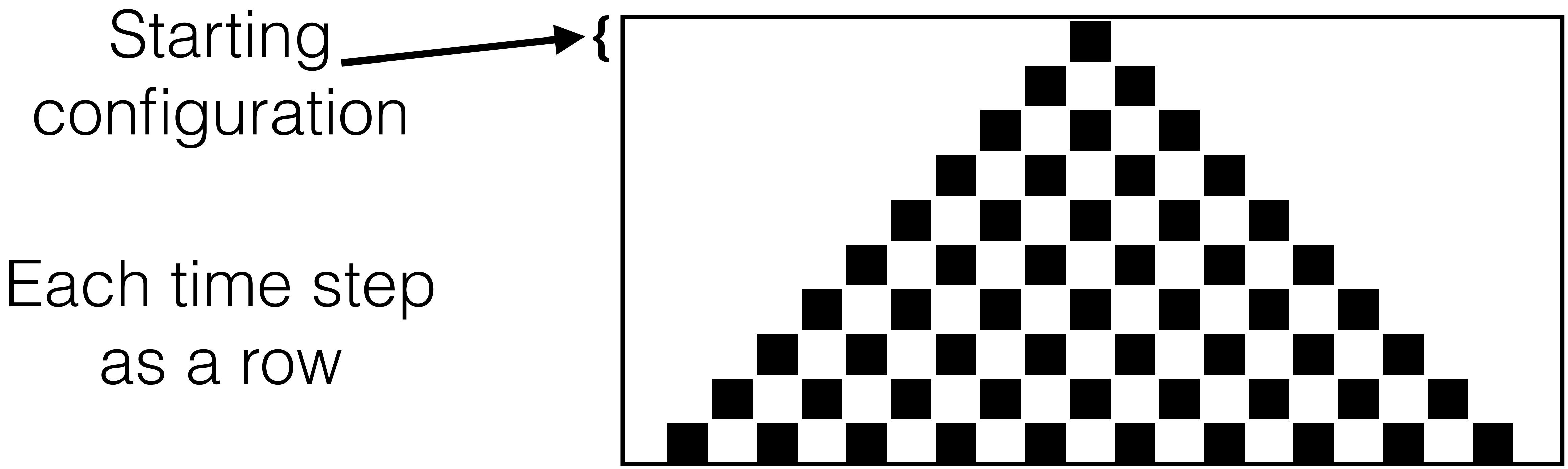
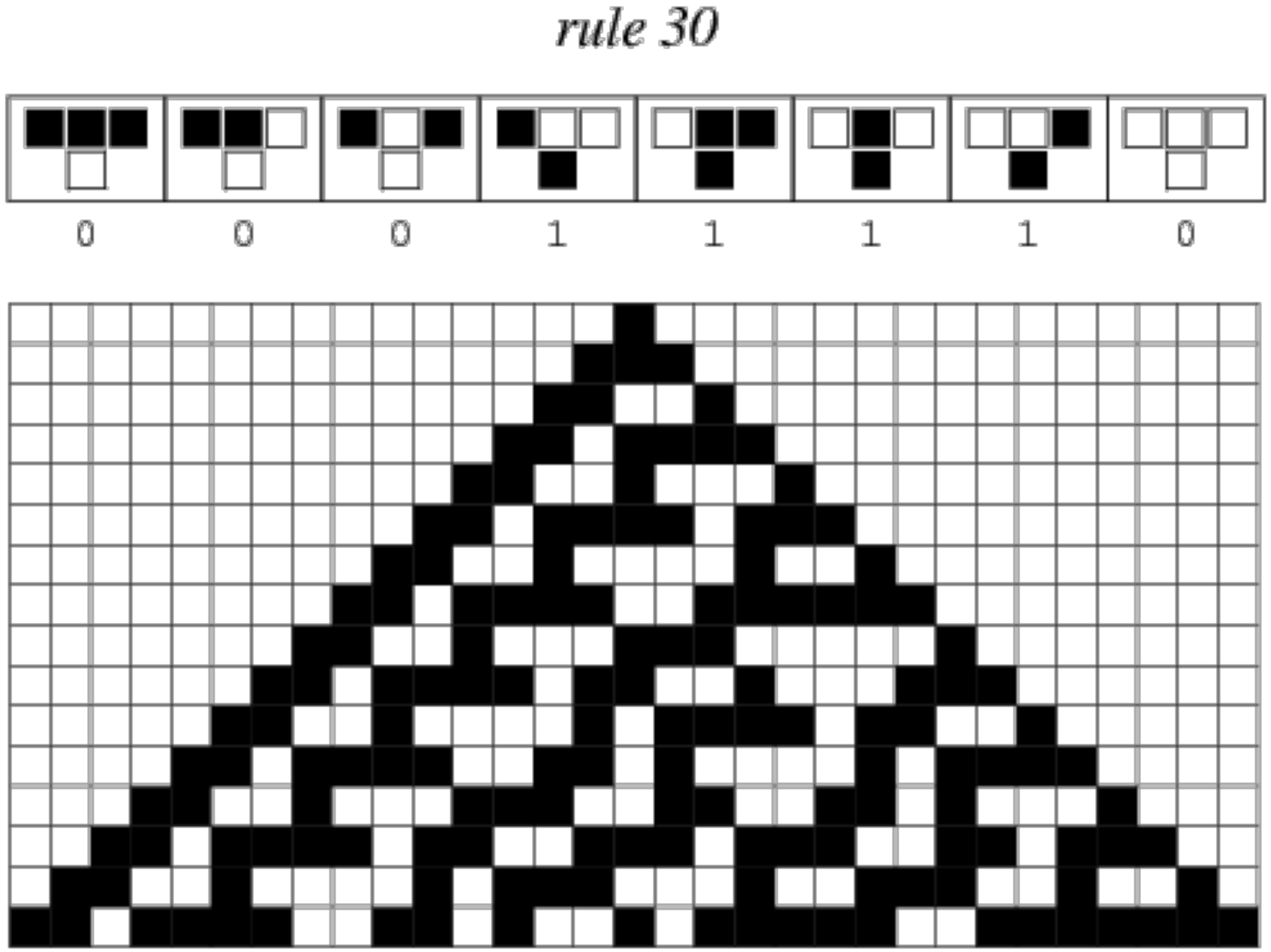


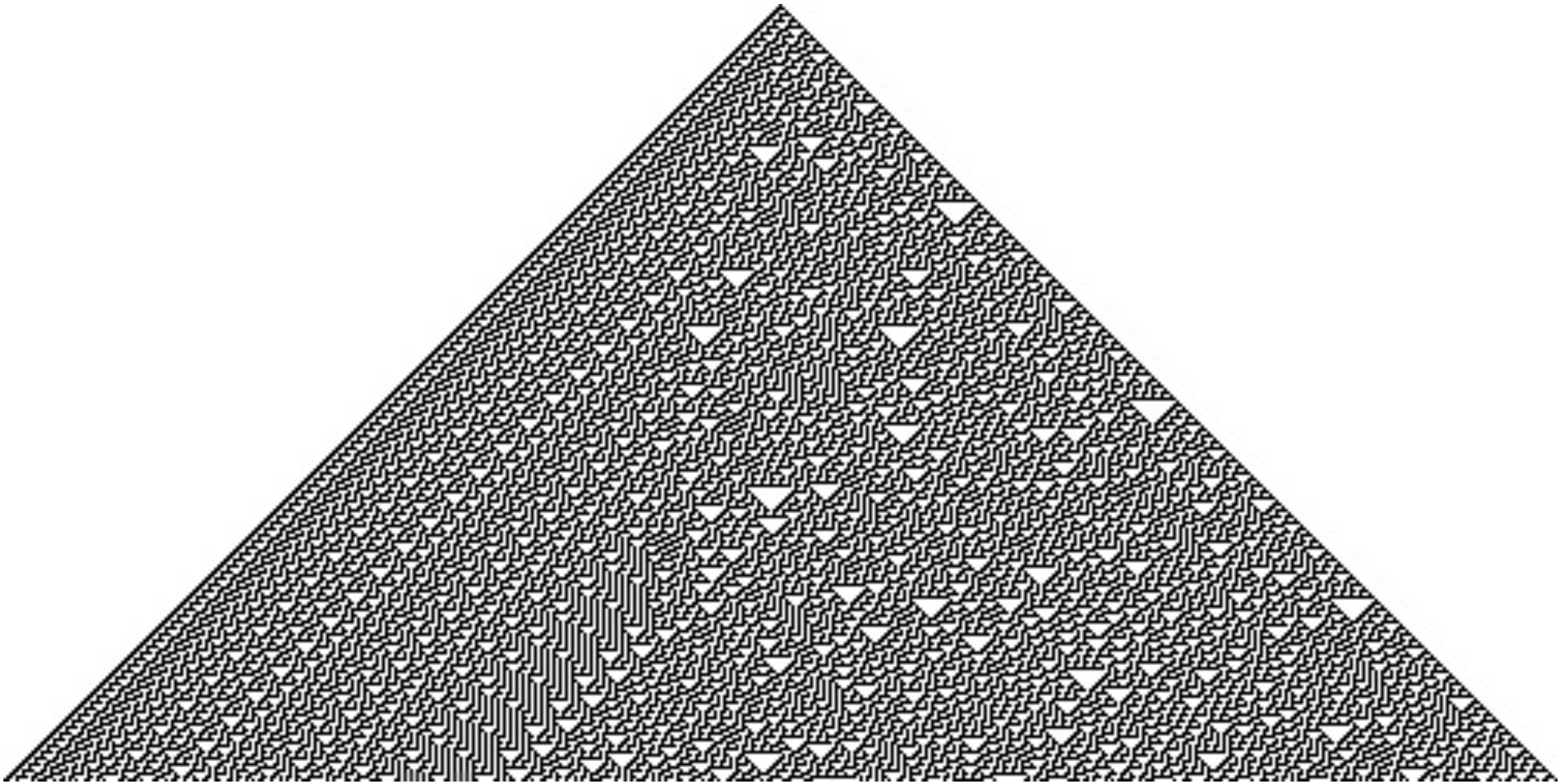
Figure 6.1: Rule 50 after 10 time steps.

Rule 30



What happens if we keep going?

http://mathworld.wolfram.com/Rule30.html



http://mathworld.wolfram.com/Rule30.html

Wolfram’s CA Classification

- CA can produce surprisingly complex behavior
- Wolfram classification - 4 classes of 1D CA
    - **Class I** – almost all initial conditions evolve to a homogeneous state, any initial randomness is lost (e.g. Rule 0)
    - **Class II** – Simple pattern, stable, oscillating, nested structure (e.g. Rule 18)

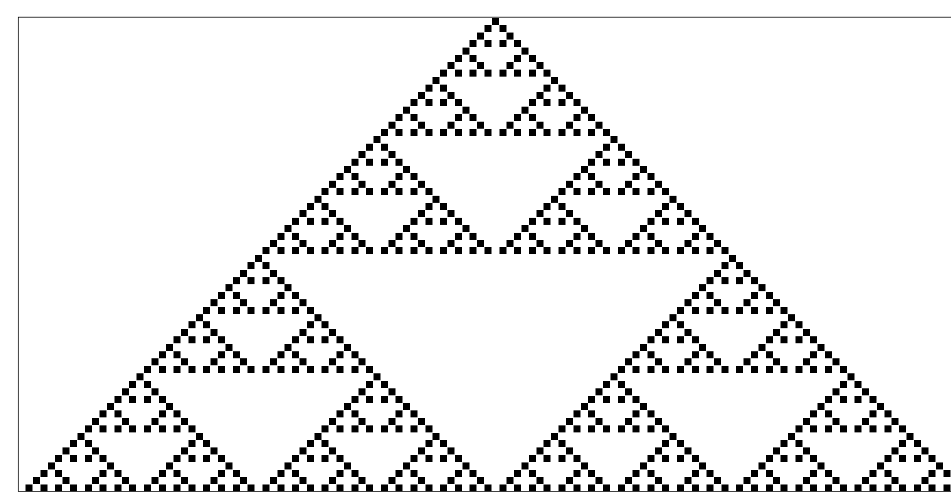


Figure 6.3: Rule 18 after 64 steps.

Downey, *Think Complexity* (Chp. 6)

Wolfram's CA Classification

- **Class III** - CAs that produce seemingly random or chaotic patterns
- Can produce sequences difficult to distinguish statistically from random, though the underlying process is deterministic
- Class III CAs typically do not produce long-lasting structures (persisting over many time steps)

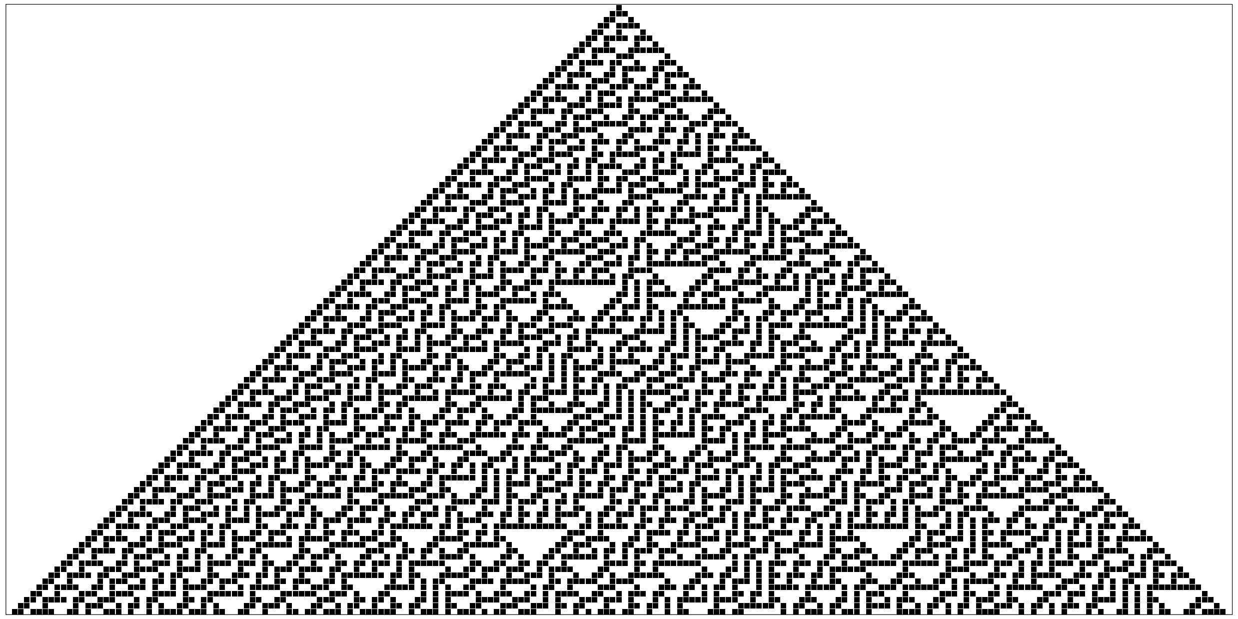


Figure 6.4: Rule 30 after 100 time steps.

Wolfram's CA Classification

- **Class IV** - Evolve in complex ways that involve a mix of "chaotic" and "ordered" (Class II and Class III)
    - Have the potential to evolve local structures that persist over many time steps

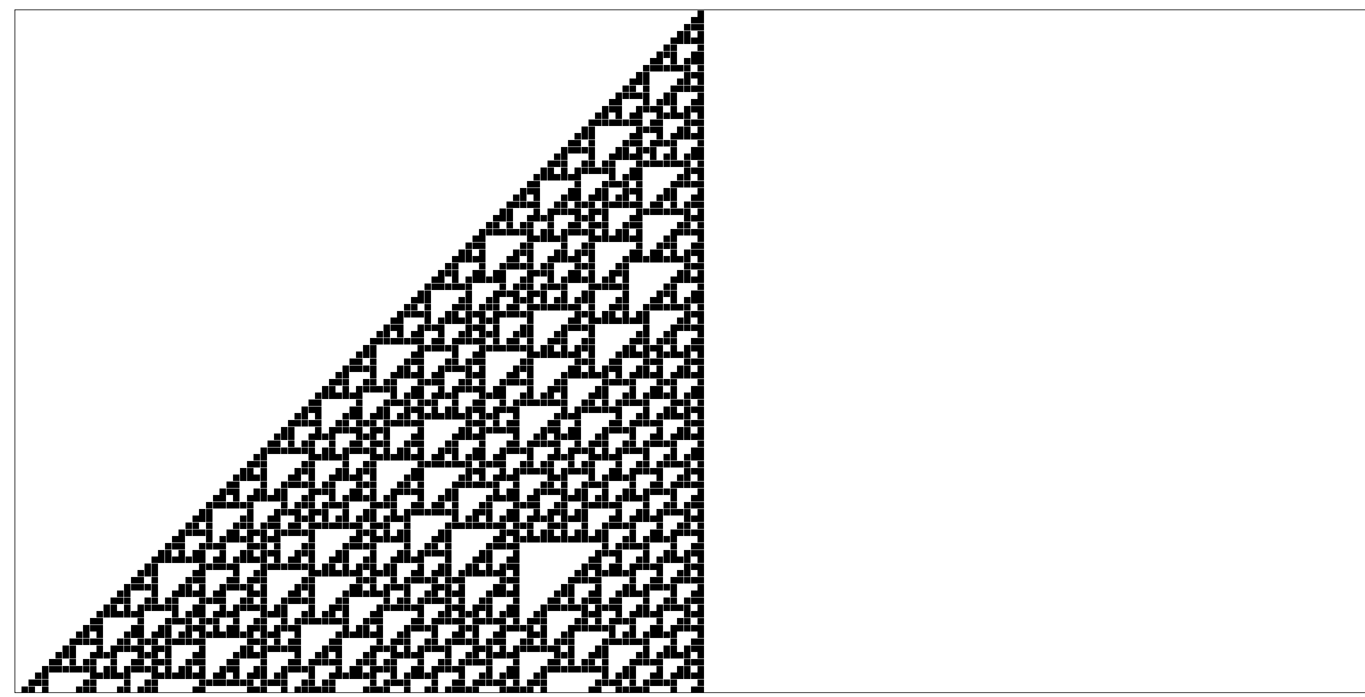
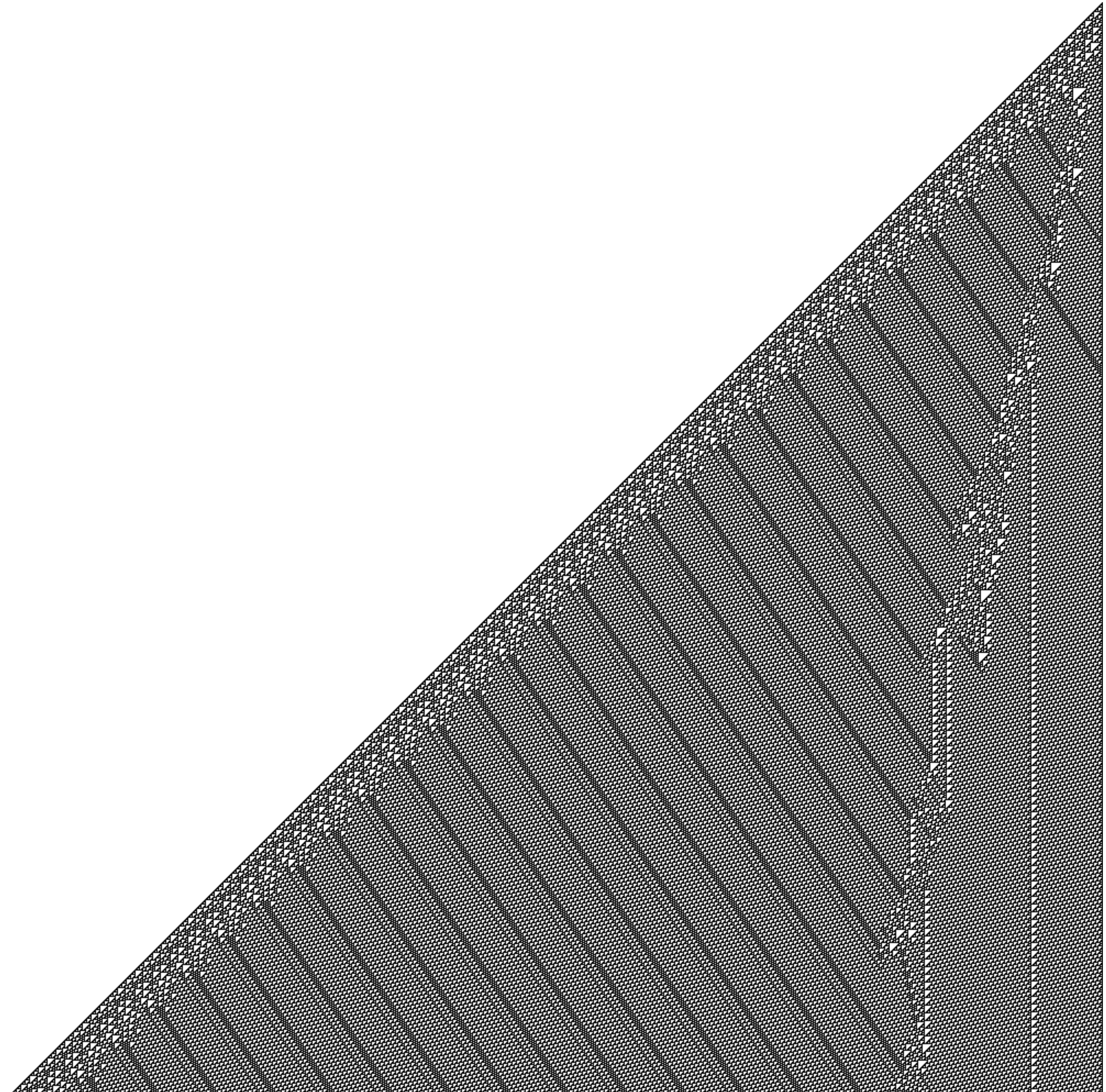


Figure 6.5: Rule 110 after 100 time steps.

Downey, *Think Complexity* (Chp. 6)



https://en.wikipedia.org/wiki/Cellular_automaton

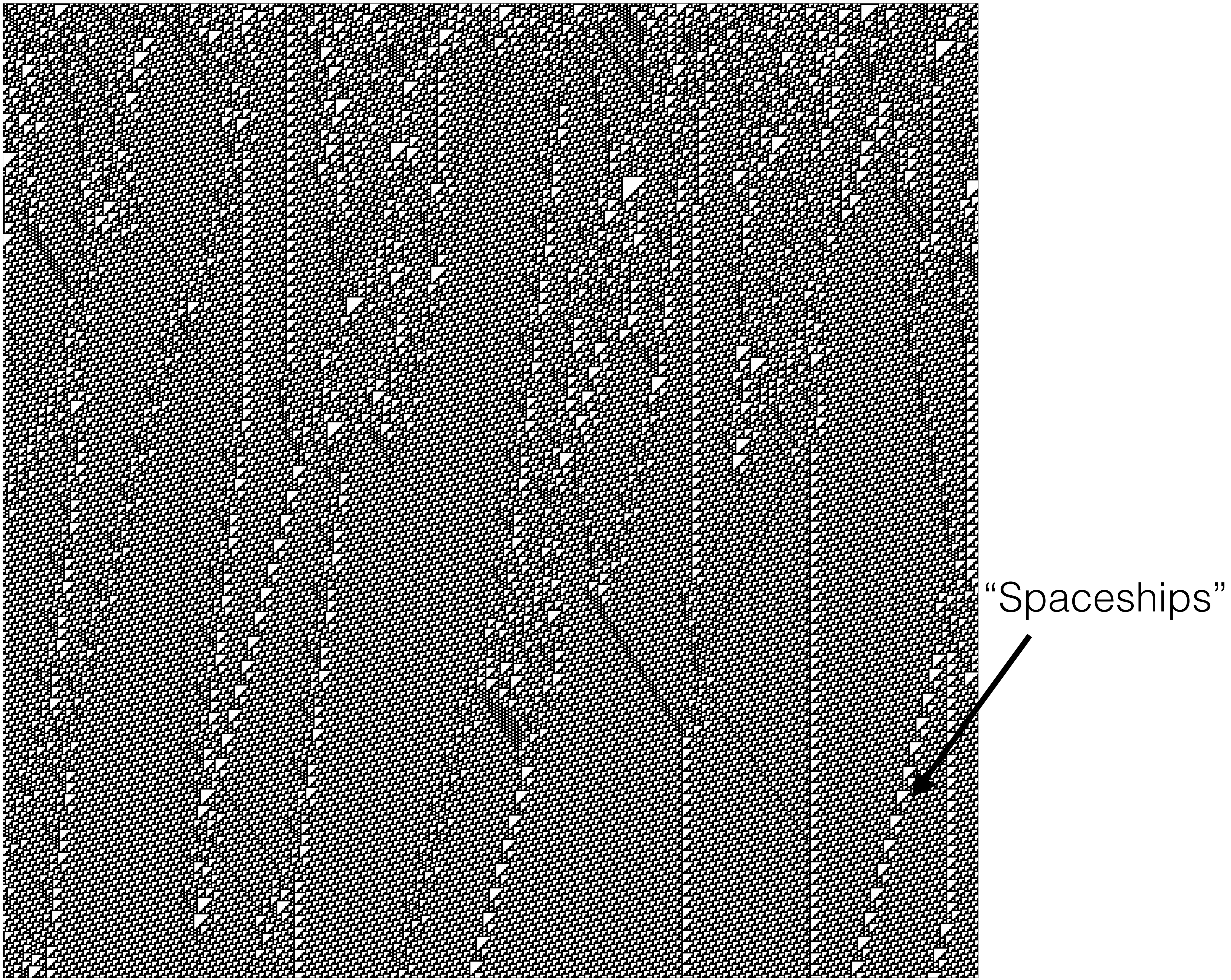


Figure 6.6: Rule 110 with random initial conditions and 600 time steps.

Downey, *Think Complexity* (Chp. 6)

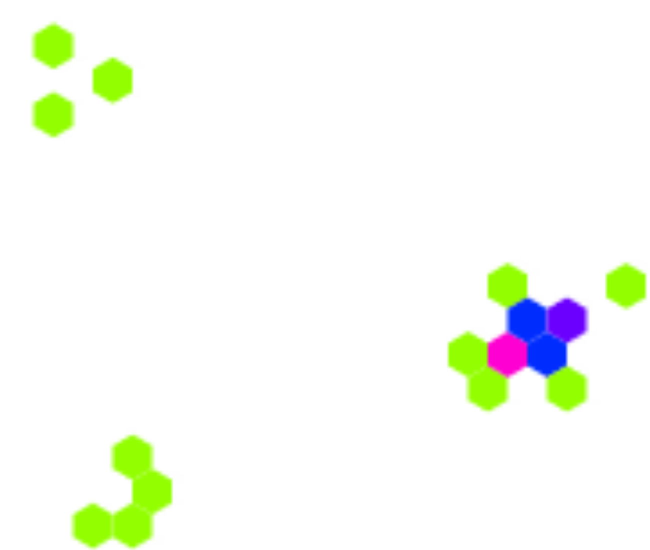
Class IV CA's and computability

- Rule 110 has been proved to be computationally universal, i.e. Turing complete (Cook M.,1998)
- So is Conway's Game of Life (classic 2D CA), and others
- Such CA can be used to compute any computable function (discuss Church-Turing Thesis)
- Wolfram's Conjecture: Every Class IV CA is Turing complete?

Downey, *Think Complexity* (Chp. 6), Wolfram's *A New Kind of Science*

Cellular Automata

- **Dimensionality** - How many dimensions?
- **Boundaries** - none (infinite domain), periodic (wrapped), cut-off (edge cells have fewer neighbors), fixed (edge cells take a fixed state)
- **Grid size**
- **Grid type** - for 2D and higher; square is typical (& will be our focus), but can do others!



https://en.wikipedia.org/wiki/Cellular_automaton

Cellular Automata

- **State Set** - binary, n-ary?
- **Initial conditions** - single cell active, random, etc.
- **Neighborhood** - queen/rook (Moore/Von Neumann), neighborhood radius
- **Rules** - totalistic (depends only on sum over neighborhood, e.g. majority rule), symmetric (e.g. state transition is the same up to rotation)

CA Notation

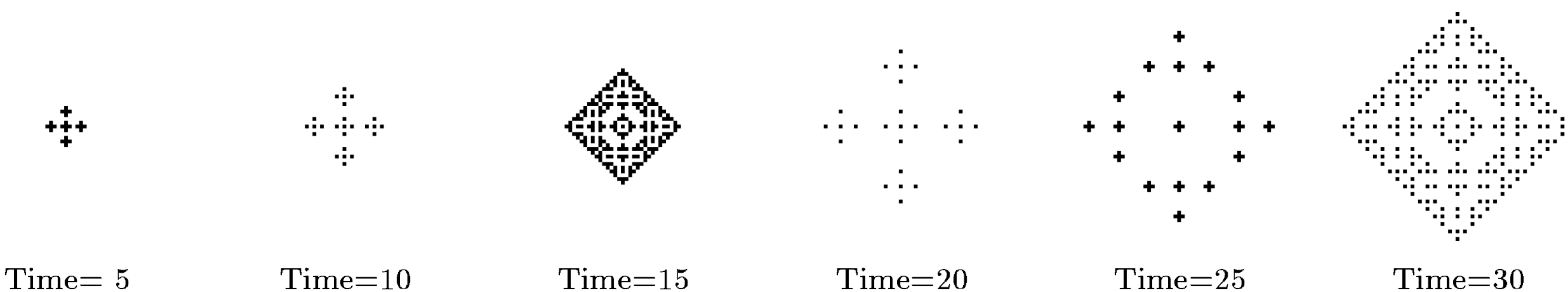
$$s_{t+1}(x) = F(s_t(x + dx_0), s_t(x + dx_1), \ldots, s_t(x + dx_{n-1}))$$

- $s_t(x)$ is the state of cell *x* at time *t*
- $N = \{dx_0, dx_1, \ldots, dx_{n-1}\}$ is the neighborhood
- Neighborhood usually defined as cells within a given radius *r* of *x*

Parity Rule

$$s_{t+1}(x) = \sum_{i=0}^{n-1} s_t(x + dx_i) \mod k$$

- Based on the mod *k* sum of neighborhood values (where *k* is the number of states)
- For binary CA, means they turn on/off based on if sum is even/odd



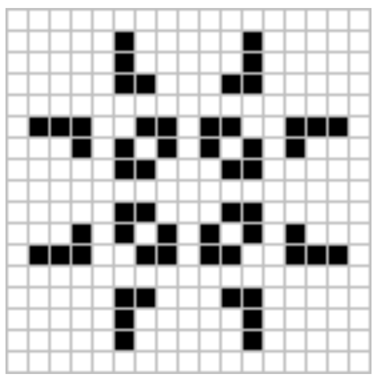
Conway’s Game of Life

- Possibly the most classic/well-known CA
- Large community of researchers/hobbyists, helped kick-start the field of ‘artificial life’
- Produces enormous range of interesting, non-trivial behaviors
- Turing-complete

Conway's Game of Lie

- Queen neighborhood (Moore neighborhood)
- A dead cell becomes alive if surrounded by exactly 3 live cells
- A living cell remains alive if surrounded by 2 or 3 living cells, otherwise it dies (either due to over- or underpopulation)

Conway’s Game of Life



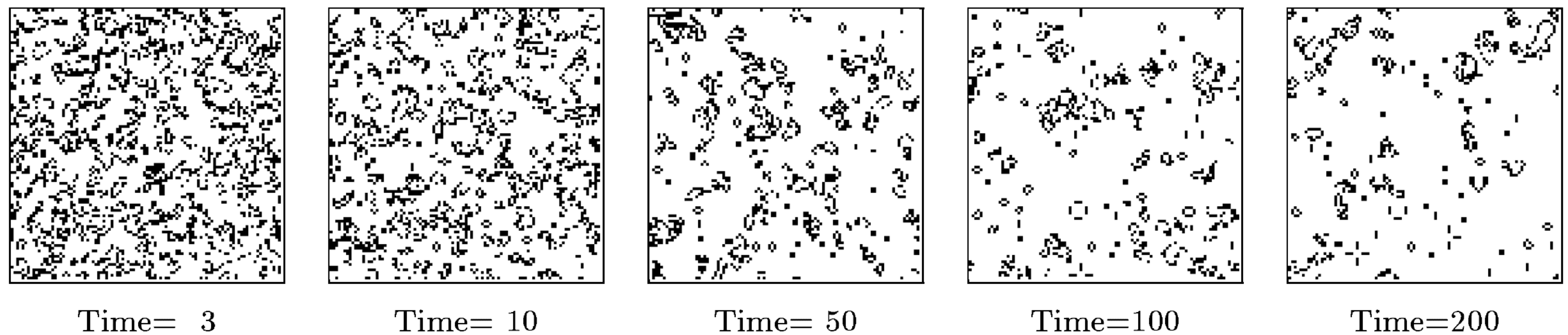
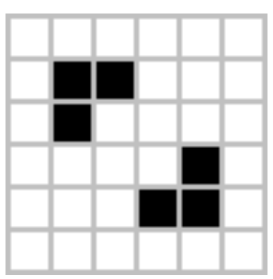
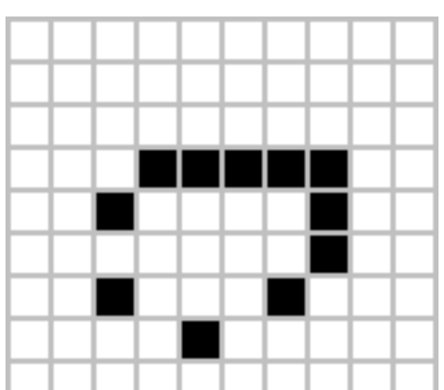
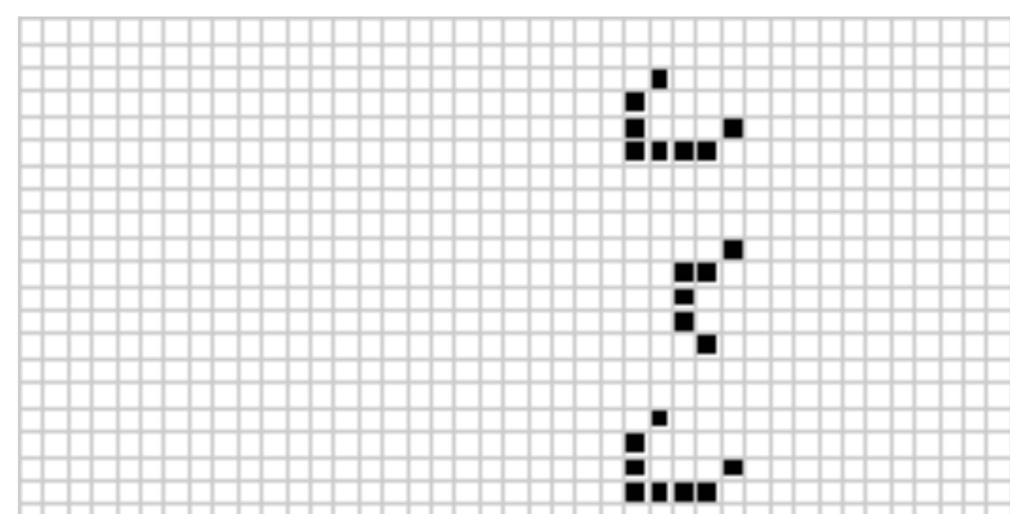


Figure 11.6: Typical behavior of the most well-known binary CA, the Game of Life.







Conway's Game of Life

- Epic collection of Conway's Game of Life patterns: https://youtu.be/C2vgICfQawE?t=70
- Nicky Case Simulator version: https://ncase.me/sim/?s=conway
- Web version to try: https://playgameoflife.com
- ca-gameoflife.py in PyCX
- Game of life wiki: https://conwaylife.com/wiki/Main_Page
- NYT: https://www.nytimes.com/2020/12/28/science/math-conway-game-of-life.html

Turmites

- 2D Turing machine generalizations
- Named “Turmites” after Turing and the fact that the write-head of the ‘machine’ moves similarly to a bug
- The ‘turmite’ or ‘ant’
- E.g. Langton’s Ant

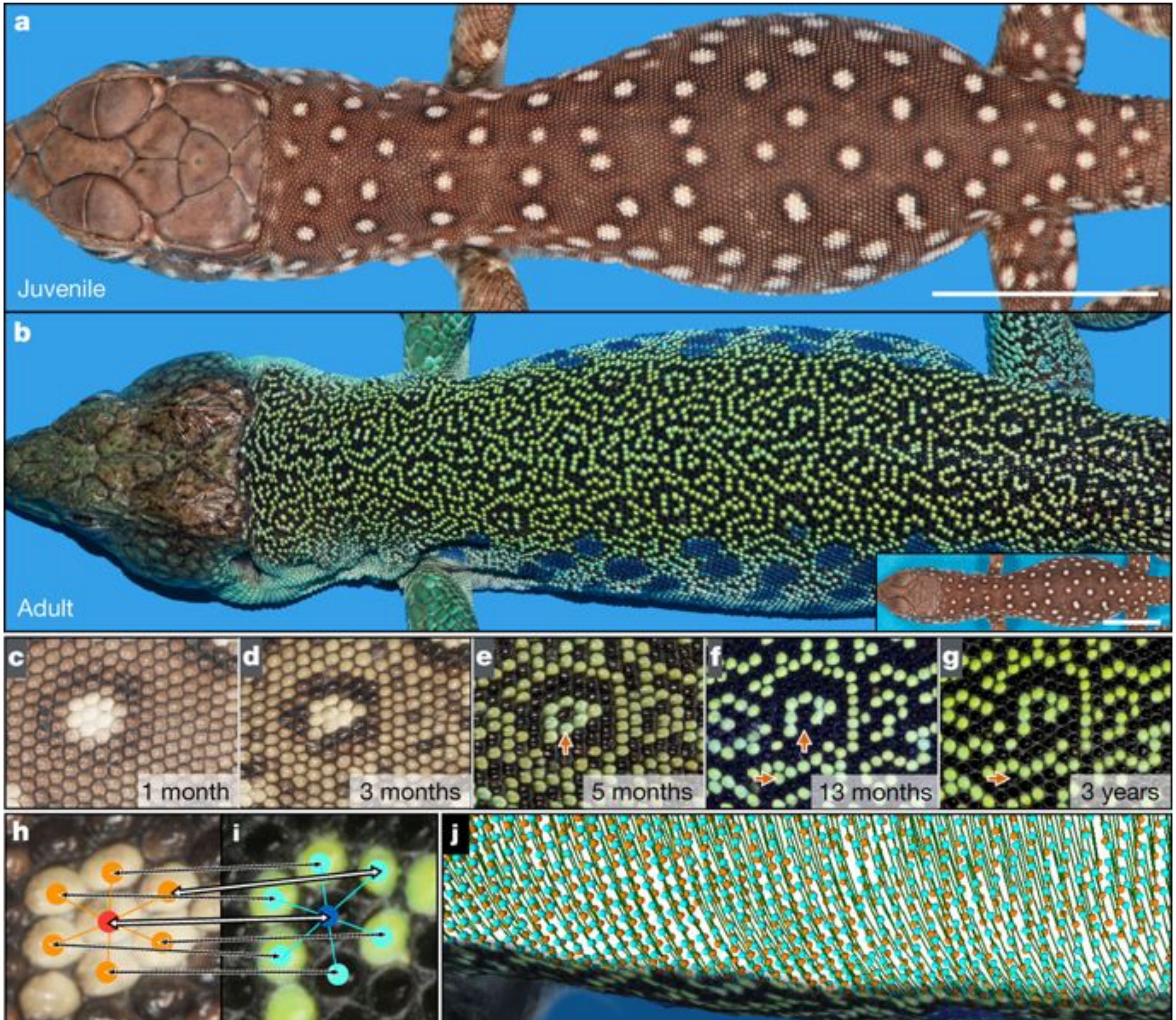
Applications of CA & real-world examples

- Forest fire models/disease epidemics
- Sand heaps/avalanches
- Majority rule and voter models
- Diffusion-limited aggregation (DLA), percolation, lattice models of materials
- And many more—some more realistic than others
- Many ABMs can be viewed as CA, or near-CA (e.g. if we allow probabilistic rather than deterministic rules)

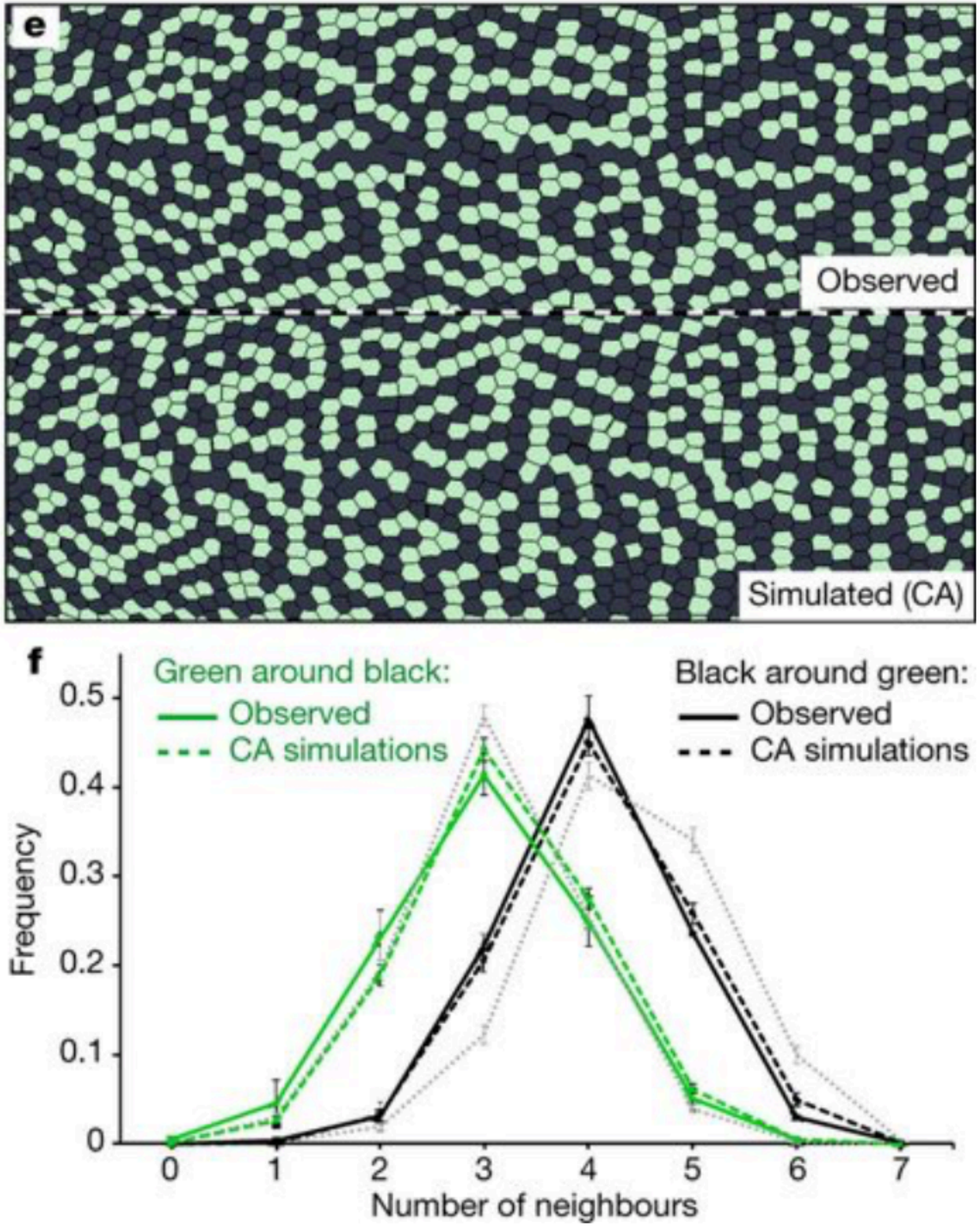
CA on seashells

- *Conus textile* appears to operate with Rule 30 (or close to it)

CA on lizard scales



https://www.nature.com/articles/nature22031



e
Observed
Simulated (CA)
f
Green around black:
Observed
CA simulations
Black around green:
Observed
CA simulations
Frequency
0
0.1
0.2
0.3
0.4
0.5
0
1
2
3
4
5
6
7
Number of neighbours

CA & ABM Dynamics

- Not always easy to interpret! Can have many patterns, as we saw with Game of Life, etc.
- However, sometimes there are major overall patterns that we can see
- More on this next time!

For next time…

- Reading
  - Sayama Chapter 11
  - Think Complexity Chapter 6
- We’ll discuss 2D CA, how to build CA, variations on CA, and theory for how to analyze the complexity and dynamics of CA