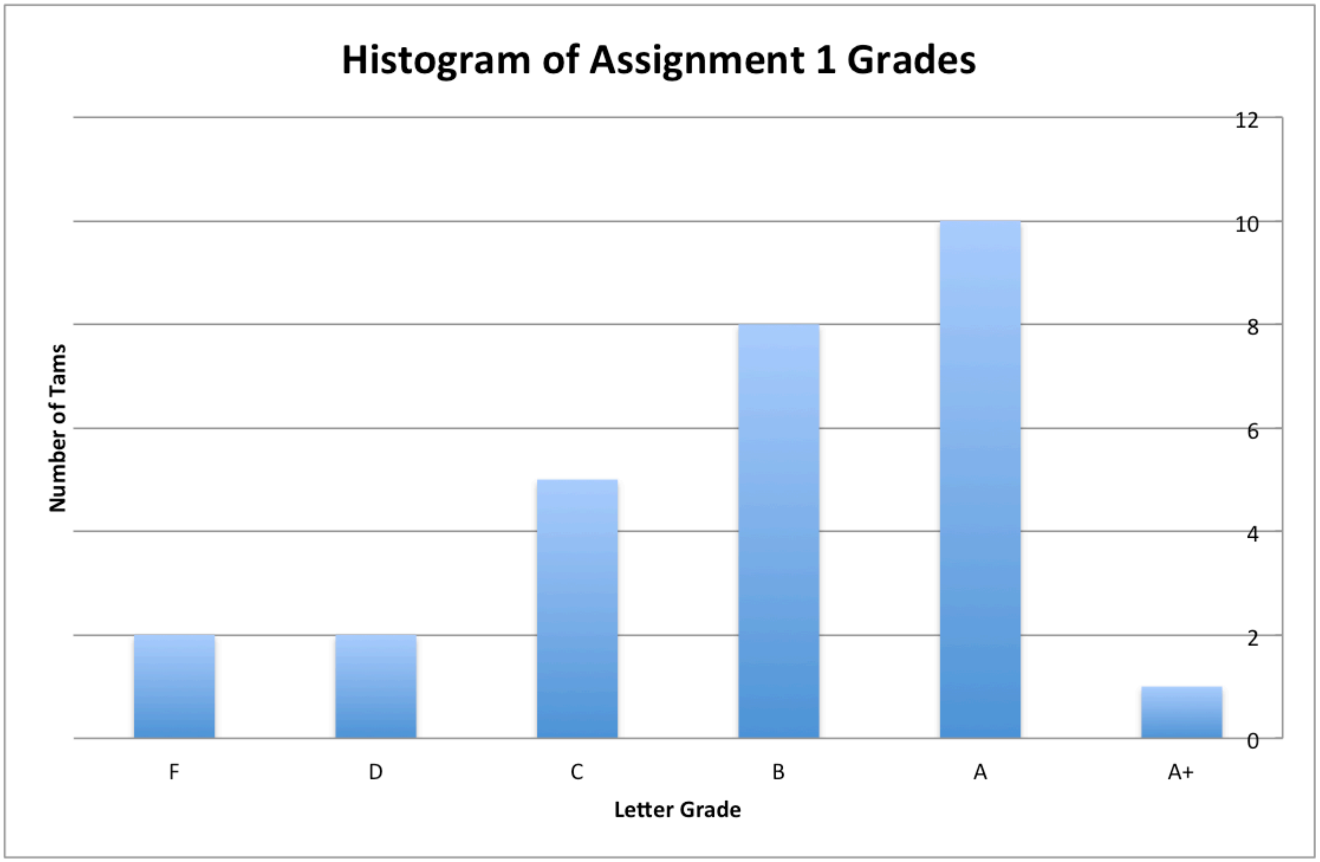
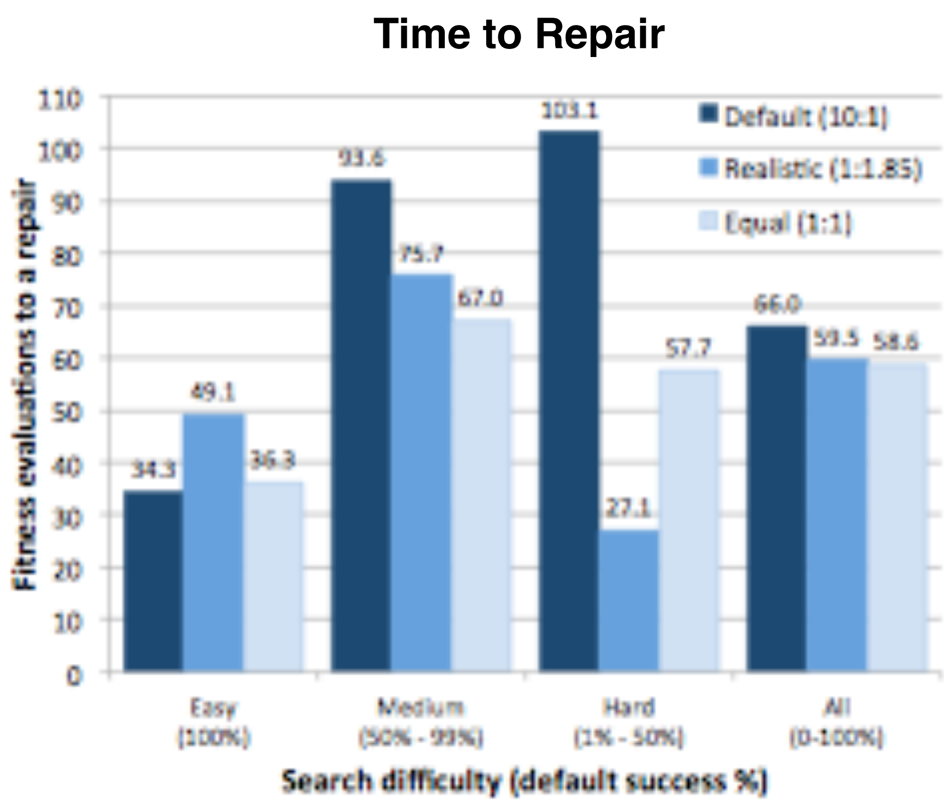


- See Karl after class in CS computer lab if you have a question about your code grade
- See me or Mathew if you have a question about your report grade AFTER filling in rubric & examples.
- If you got <75%, office hrs Th am, Mon pm. Also writing tutors at Office of Grad Studies.
- Meet after class if you need a partner for Project 2
- Send presentation slides to Matthew

GAs

- Claire LeGoues talk
  - Weighted path & constraining the search space
  - GAs challenge assumptions by finding bizarre solutions
  - Crossover
  - Fuzz testing & real-world automated repair for security vulnerabilities
- Natural selection
  - has No Goal
  - operates at multiple levels
  - evolves robustness & evolvability
- Project 2 Part 2
  - evolve cooperation & maybe competition
  - code is in C++

Weighted Path Schemes



Conclude: Both new weighting schemes cause the GP to repair a bug on which the default weighting scheme failed.

Mitchell Chapter 10

- Living systems compute
- Living systems are open systems that exchange energy, materials & <u>information</u>
  - E.g. Erwin Shrodinger (1944) & Lynn Margulis (200) books: *What is Life?* discuss how life creates order from disorder
- Computation is what a complex system does to adapt to its environment
- Life *is* computation?
- Cellular Automata: simplified demonstrations of how life might compute

Rule 110

0	0	0	→	0
0	0	1	→	1
0	1	0	→	1
0	1	1	→	1
1	0	0	→	0
1	0	1	→	1
1	1	0	→	1
1	1	1	→	0

Read the bit string BOTTOM to TOP

111 is leftmost (most significant) bit

000 is the least significant bit

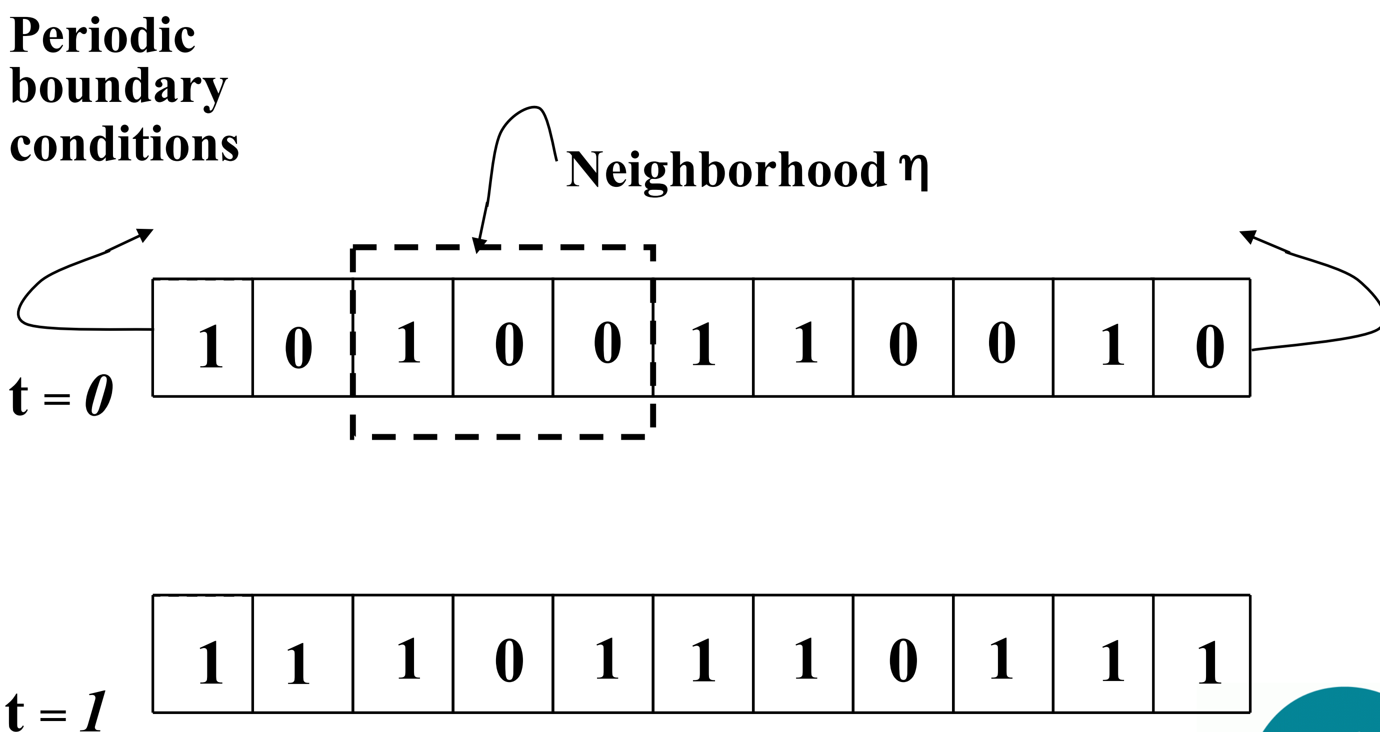
000	001	010	011	100	101	110	111
0	1	1	1	0	1	1	0

0 1 1 0 1 1 1 0 = 0+ 2 + 4 + 8 + 0 + 32 + 64 = Rule 110

Rule Table ϕ :

Neighborhood:	000	001	010	011	100	101	110	111
Output bit:	0	1	1	1	0	1	1	0

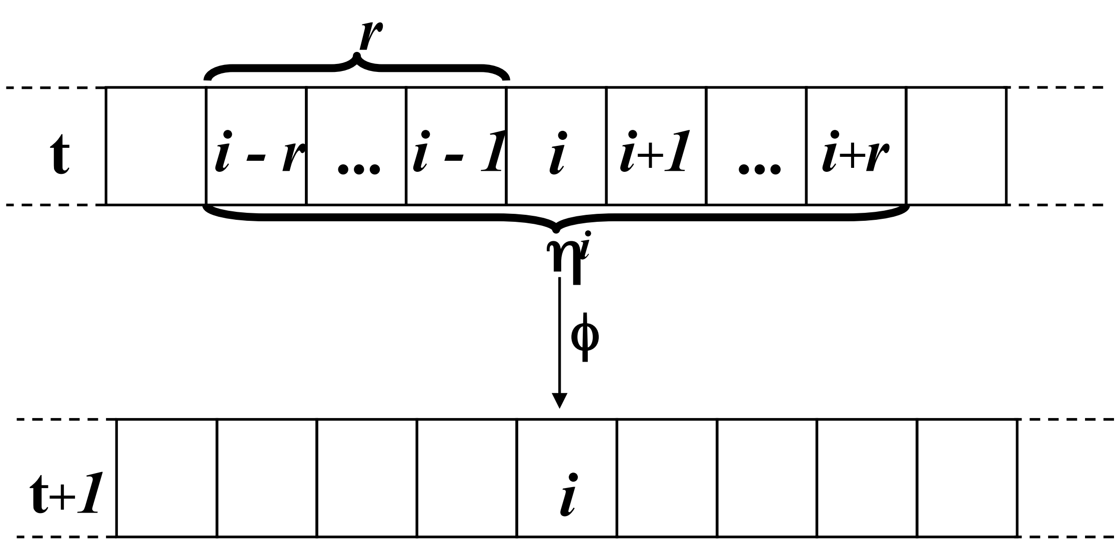
Lattice:



Introduction to Cellular Automata (CA)

- Invented by John von Neumann (circa~1950).
- A cellular automata consists of:
  - A regular arrangement (lattice) of cells.
  - Cells can be in one of a finite number of states at each time step.
  - All cells have the same synchronous update rule.
  - Cells have a local interaction neighborhood.
- Example: The game of life.
- Many cellular automata applications:
  - Hydrodynamics, fluid dynamics
  - Forest fire simulation
  - Models of ecosystems and epidemics
  - Ferro-magnetic modeling (ISING models)

One-Dimensional CA Illustration



The University of New Mexico

One-Dimensional Cellular Automaton (CA)

- Consists of a linear array of identical *cells* (called a *lattice*), each of which can be in a finite number of *k* states.
- The (local) *state* of cell *i* at time *t* is denoted: $s_t^i \in \Sigma = \{0,1,\ldots,k-1\}$
- The (global) state s_t at time *t* is the *configuration* of the entire array,

$$\mathrm{s}_t = (s_t^0, s_t^1, \ldots, s_t^{N-1}) \in \Sigma^N$$

- where *N* is the (possibly infinite) size of the array.

The University of New Mexico

One-Dimensional CA cont.

- At each time step, all cells in the array update their state simultaneously, according to a *local update rule* $\phi : \eta \rightarrow s$.
- This update rule takes as input the *local neighborhood configuration* η of a cell.
- A local neighborhood configuration η consists of s_i and its 2*r* nearest neighbors (*r* cells on either side):

$$\eta_i = (s^{i-r}, \ldots, s, \ldots, s^{i+r})$$

- *r* is called the *radius* of the CA.
- The local update rule ϕ
  - is the same for every cell in the array
  - Represented as a lookup table, with all neighborhood configurations
- The state of each cell at time *t*+1 is determined by applying ϕ to each cell at time *t:*

$$s_{t+1}^{i} = \phi(\eta_t^i)$$

Rule 110

0	0	0	→	0
0	0	1	→	1
0	1	0	→	1
0	1	1	→	1
1	0	0	→	0
1	0	1	→	1
1	1	0	→	1
1	1	1	→	0

The number of rules in the rule table
is the number of neighborhood configurations

$$(|\eta| = k^{2r+1})$$

Each rule can map to one of k states, so
The number of possible rules is

$$k^{2r+1}$$

$$k$$

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

0 1 1 0 1 1 1 0 = 0+ 2 + 4 + 8 + 0 + 32 + 64 = Rule 110
Read the bit string BOTTOM to TOP

Example One-Dimensional CA Rule 110

- The number of states, k=2.
- The alphabet $\Sigma = \{0,1\}$ $\quad |\Sigma| = k$
- The size of the array, N=11.
- The configuration space η

  $$\Sigma^N = \{(0,0,0,0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,0,0,0,1\,1),\ldots\}$$

- The radius r = 1.
- The rule table : $s_{t+1}^i = \phi(\eta_t^i)$

  k^{2r+1} neighborhood configurations in ϕ

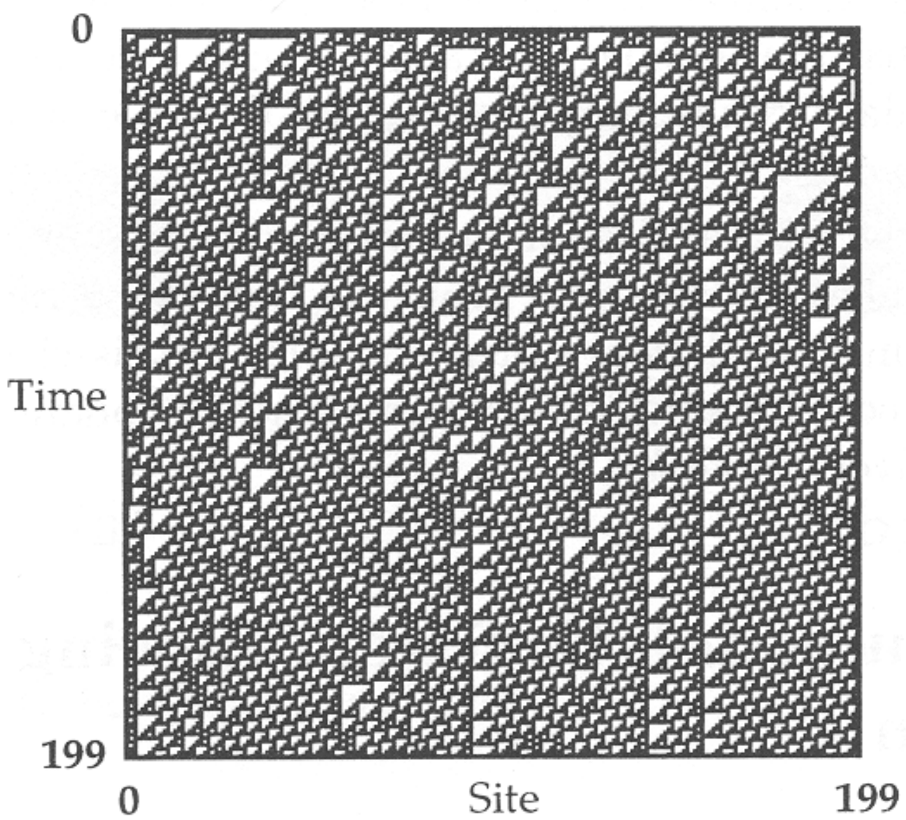
000	001	010	011	100	101	110	111
0	1	1	1	0	1	1	0

- In Wolfram notation this is rule 110 (base 10) because the output states are: 01101110 (base 2).

- Rule 110 supports universal computation.

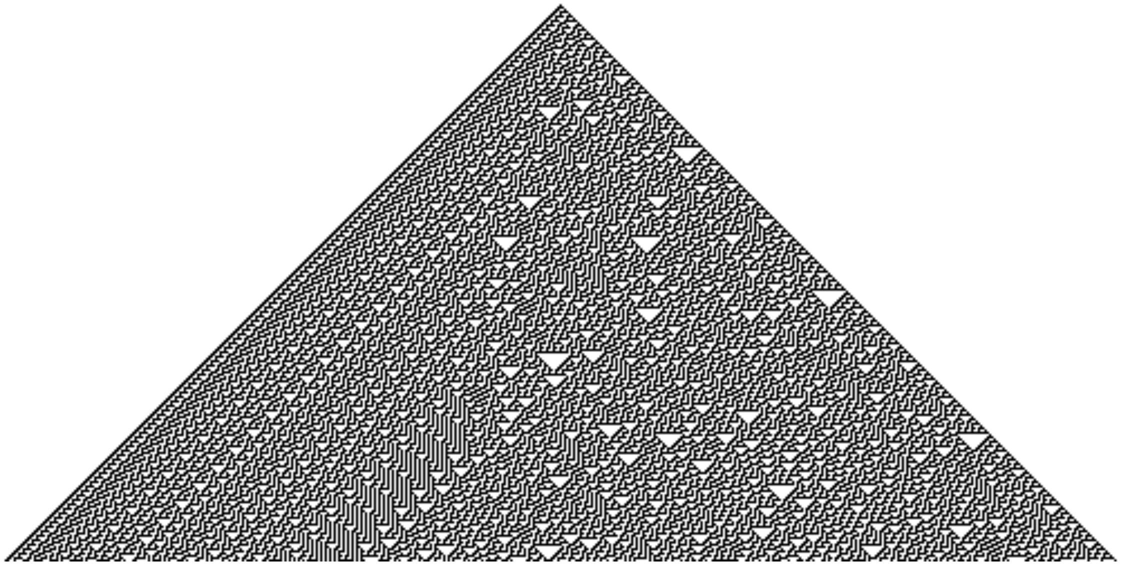
The University of New Mexico

Rule 110 Space-Time Plot



The University of New Mexico

Rule 30



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

The University of New Mexico

Comments on Rule 30

- Generates apparent randomness, despite being finite
- Wolfram uses the central column as a pseudo-random number generator in Mathematica
- Passes many tests for randomness, but many inputs produce regular patterns:
  - All zeroes
  - 00001000111000 repeated infinitely (try separating by 6 1s)

The University of New Mexico

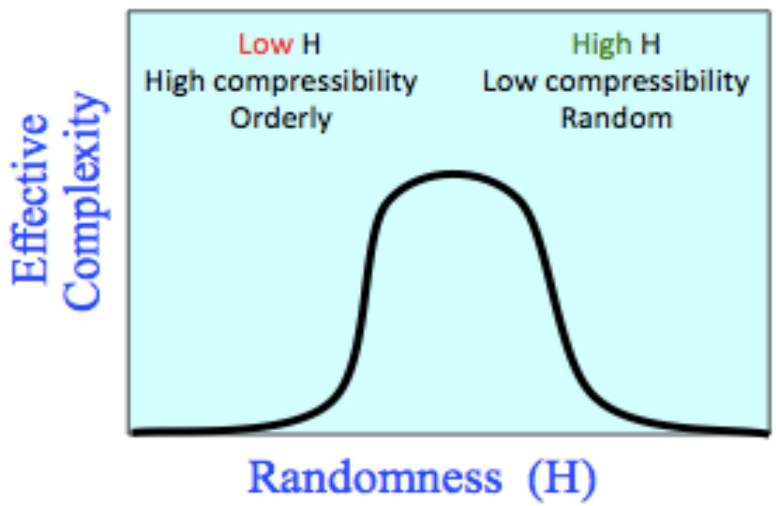
Wolfram's CA Classification

- Class I: Eventually every cell in the array settles into one state, never to change again. Analogous to
  - computer programs that halt after a few steps
  - dynamical systems that have fixed-point attractors
- Class II: Eventually the array settles into a periodic cycle of states
  - computer programs that execute infinite loops
  - dynamical systems that fall into limit cycles.
- Class III: The array forms "aperiodic" random-like patterns.
  - computer programs that are pseudo-random number generators (pass most tests for randomness, highly sensitive to seed, or initial condition).
  - Analogous to chaotic dynamical systems. Determinisitic but almost never repeat themselves, sensitive to initial conditions, embedded unstable limit cycles.

The University of New Mexico

Wolfram's Classification cont.

- Class IV: The array forms *complex* patterns with localized structure that move through space and time:
  - Difficult to describe. Not regular, not periodic, not random.
  - Speculate that it is interesting computation.
- Hypothesis: The most interesting and complex behavior occurs in Class IV CA---"the edge of chaos".
- Example: Rule 110



Wolfram Class I

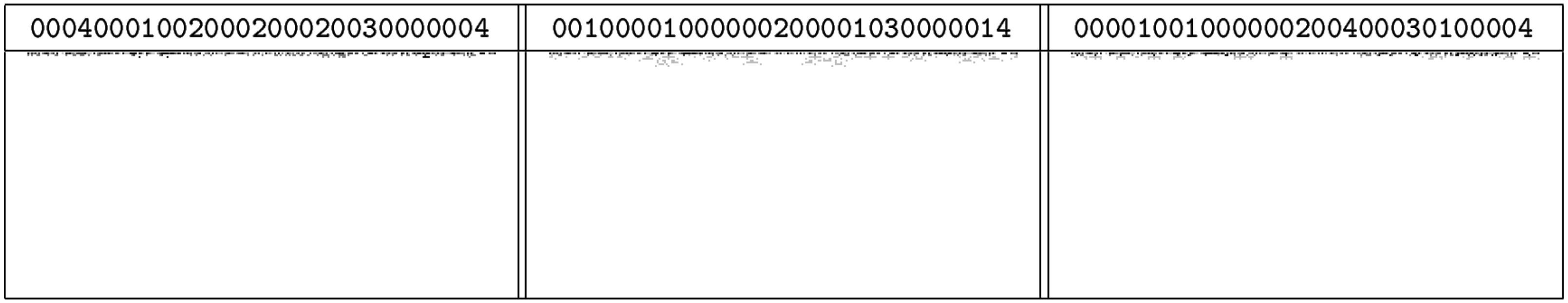


Figure 15.5 Examples of Wolfram's Class I

Wolfram Class II

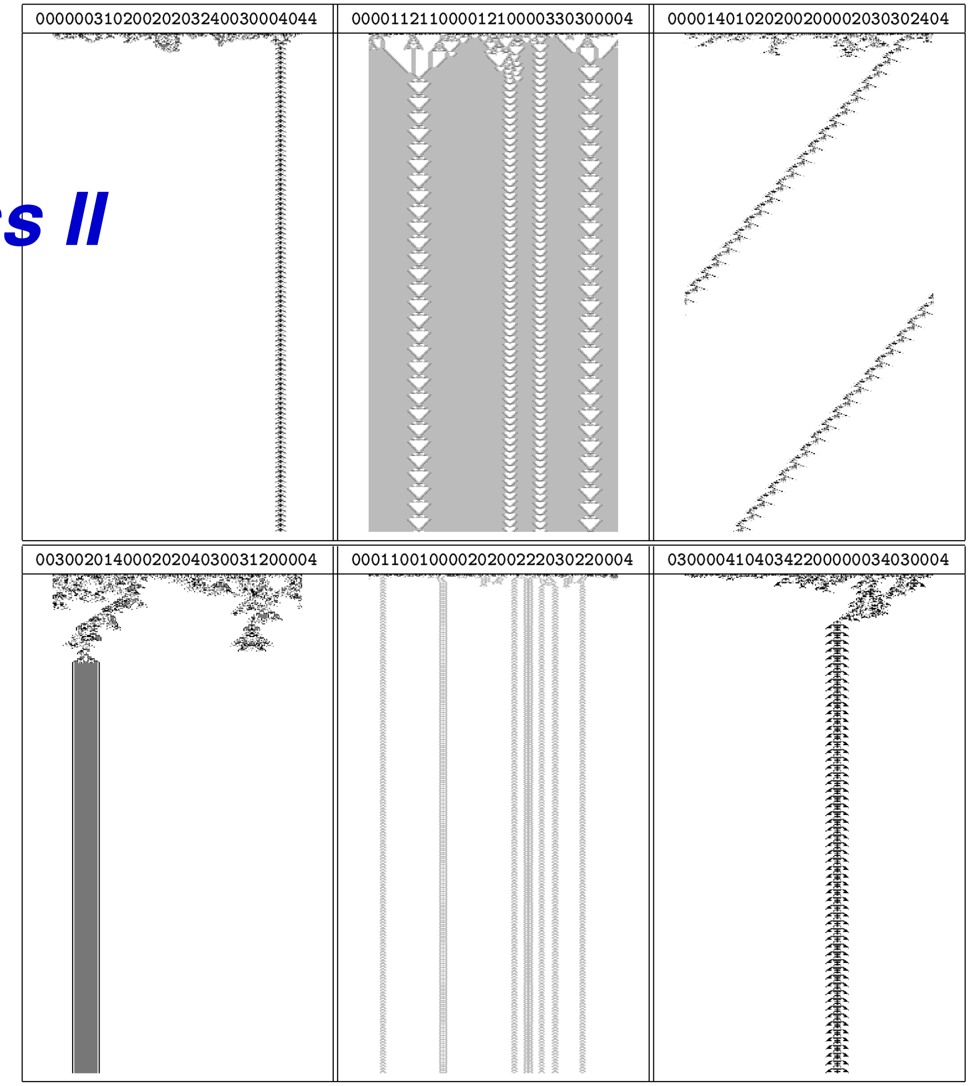


Figure 15.6 Examples of Wolfram's Class II

The University of New Mexico

Wolfram Class III

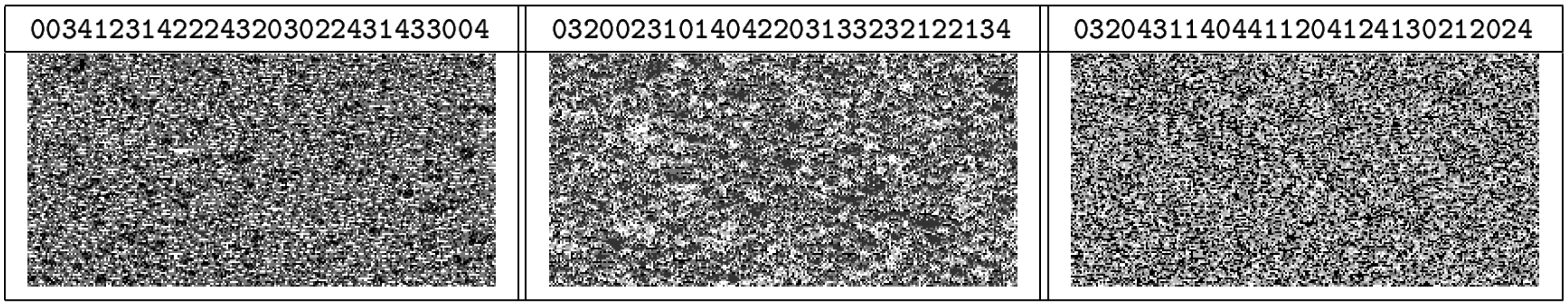


Figure 15.7 Examples of Wolfram's Class III

The University of New Mexico

Wolfram Class IV

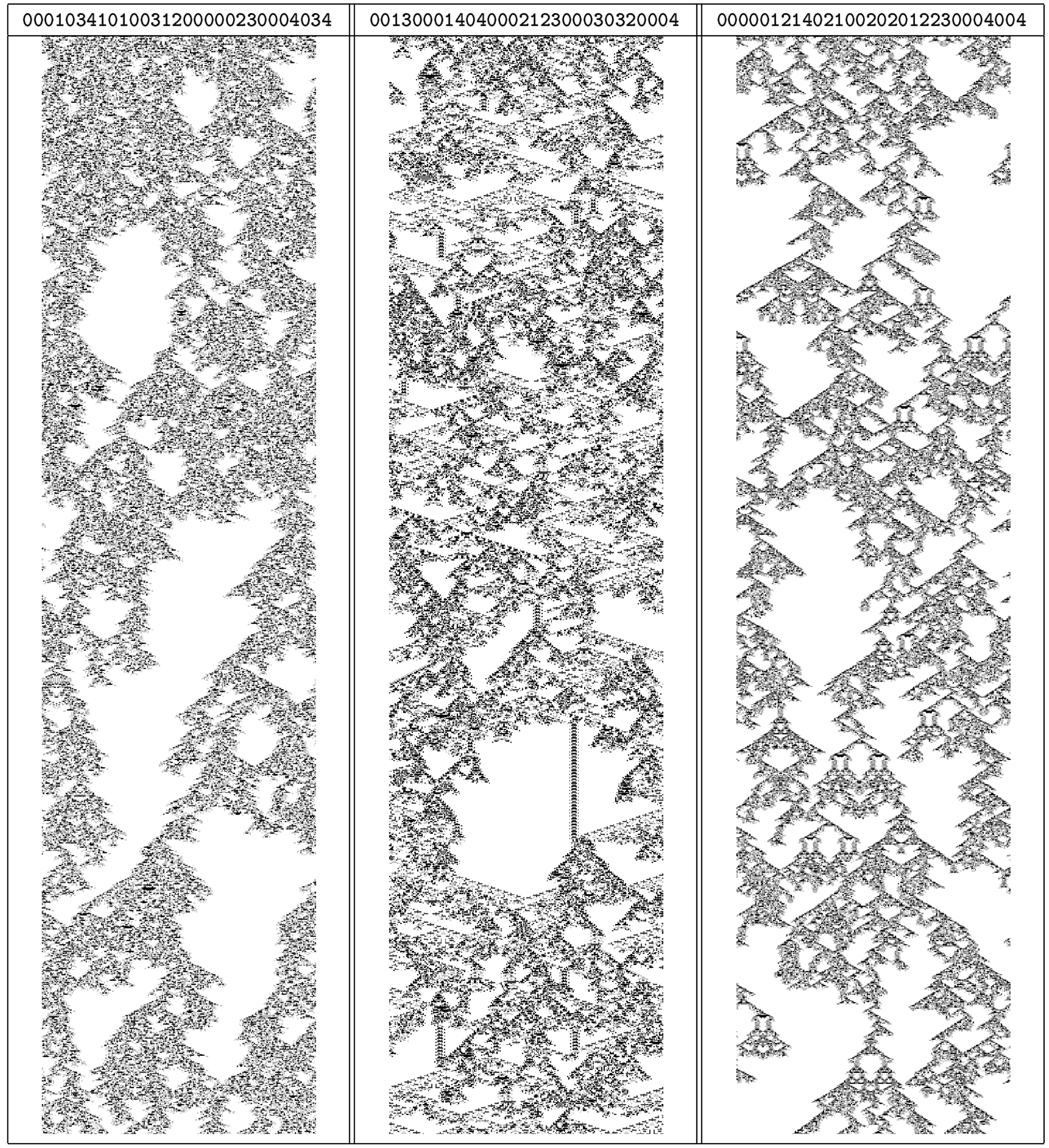


Figure 15.8 Examples of Wolfram's Class IV

Cellular Automata as models of complexity
Wolfram *Nature* 1984

http://twilightstarsong.blogspot.com/2009/03/
shaping-future-stephen-wolfram-ray.html

- 4 classes
  - I: Fixed point
  - II: Limit cycle
  - III: Random or chaotic (unpredictable)
  - IV: Complex
- Beyond $r = 1$ (Figs 3 – 6)
- Entropies & Dimensions
- Information Propagation
- Class IV supports universal computation (?)
  - Long-term behavior undecidable and intractable
  - Simulation is necessary

Entropies & Dimensions

Class I: no randomness in space or time (0 entropy)

Class II: randomness in space, predictable in time after a transient

Class III: unpredictable in time and space

Class IV: undecidable

The University of New Mexico

Information Propagation

Class I: no information propagation

Class II: propagation in a finite region

Class III: propagation over an increasing region, at fixed speed, over time

Class IV: irregular propagation over (potentially) infinite region

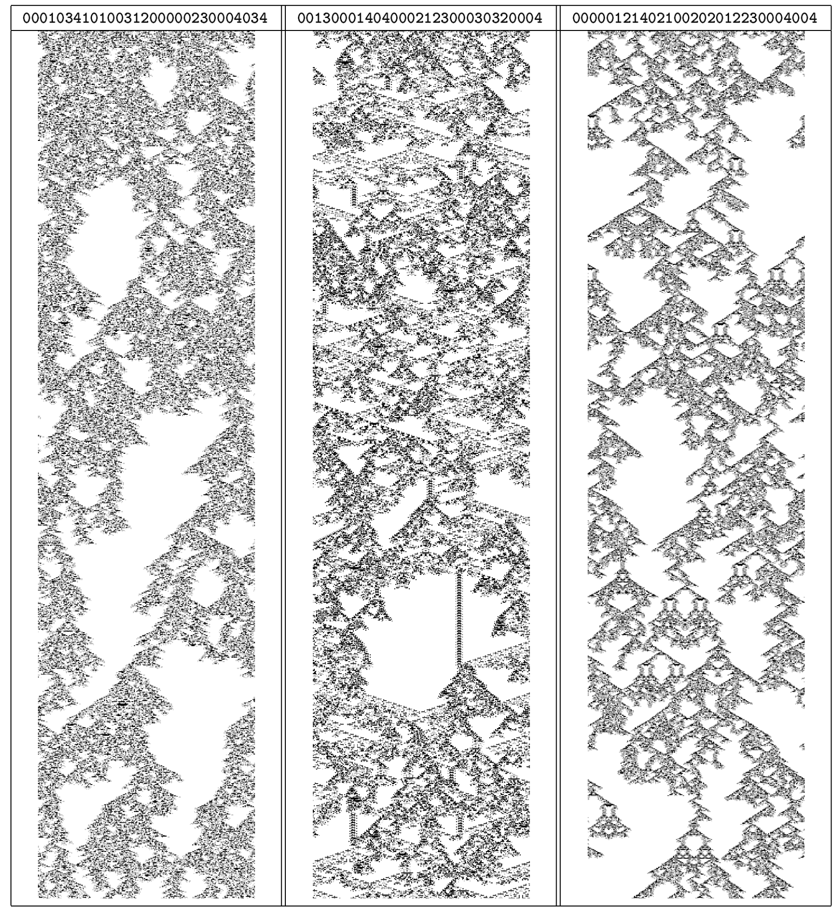


Figure 15.8 Examples of Wolfram's Class IV