Exploring Complexity

In Science and Technology

Nov. 8, 2010

Jeff Fletcher

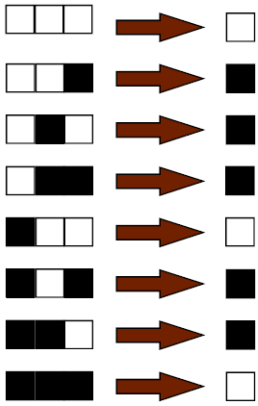
Logistics

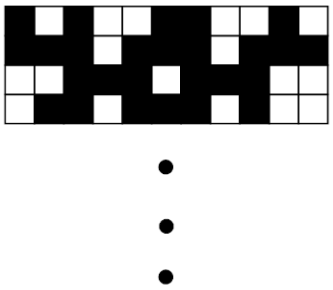
- Due
  - HW6 and Lab5 due Monday Nov. 15
- Ideas for final papers
  - Proposals (one paragraph) due today
- Questions?

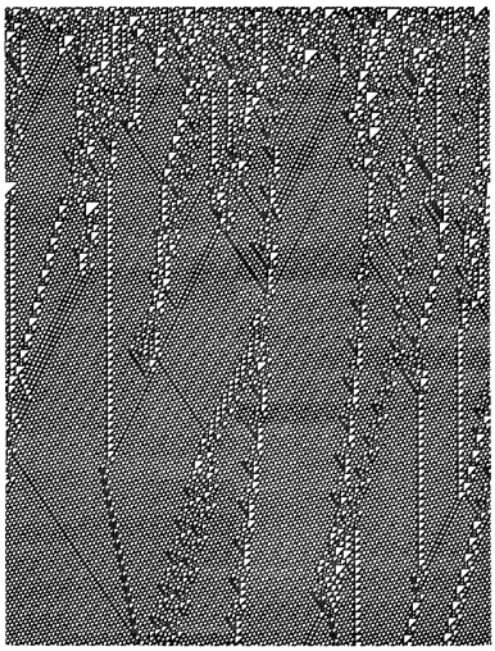
Elementary cellular automata

- one-dimensional, two states (black and white = on and off)

Rule:







Wolfram's hypothesis

- All class 4 CAs can support universal computation
- Outline of Proof
  - Define "cyclic tag systems" (http://en.wikipedia.org/wiki/Tag_system)
    - Created by Matthew Cook under the employ of Stephen Wolfram
  - Prove they are universal (they can emulate Turing machines).
  - Show ECA 110 can emulate a cyclic tag system.

Outline of Wolfram's A New Kind of Science

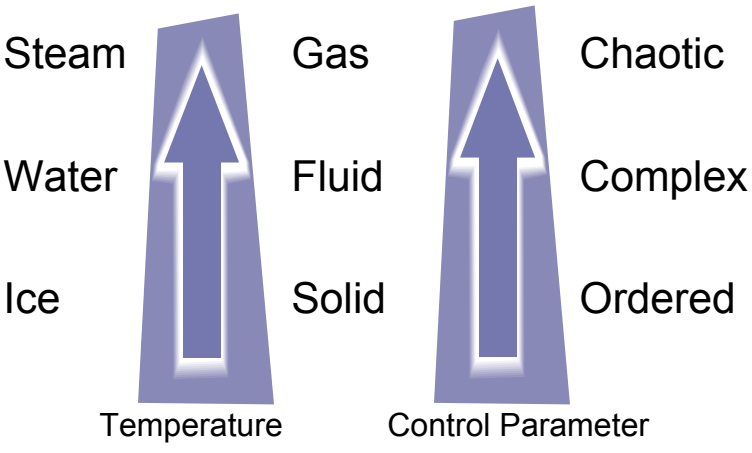
(from MM review, Science, 2002)

- Simple programs can produce complex, and random-looking behavior
  - Complex and random-looking behavior in nature comes from simple programs.
- Natural systems can be modeled using cellular-automata-like architectures
- Cellular automata are a framework for understanding nature
- Principle of computational equivalence

Principle of Computational Equivalence

- The ability to support universal computation is very common in nature.
- Universal computation is an upper limit on the sophistication of computations in nature.
- Computing processes in nature are almost always equivalent in sophistication.

Phase transitions



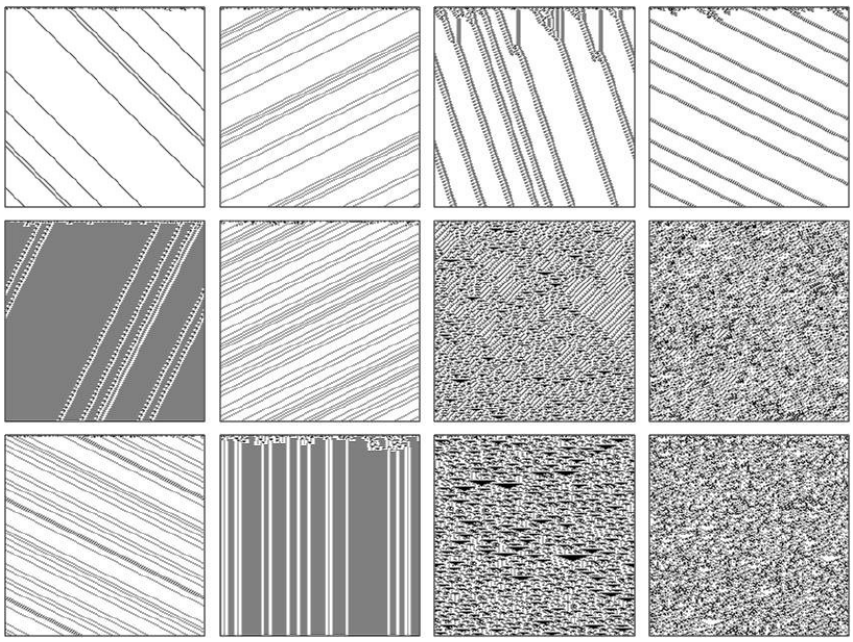
Hypothesis (Langton, Packard, Kauffman, others)

- Need maximally “fluid” state to maximize potential for:
  - information processing
  - complexity of dynamics
  - ability to adapt

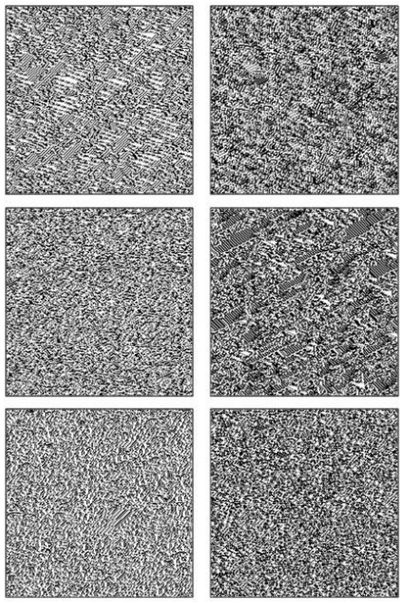
The ‘edge of chaos’ in cellular automata

(C. Langton, Physica D, 42:12-37, 1990)

- Langton devised an “order parameter” for
- cellular automata called λ.
- • For binary-state CAs, λ is defined as follows:
- $\lambda = \frac{\text{number of 1s in rule table' s output bits}}{\text{number of entries in rule table}}$



λ = 0.1 λ = 0.2 λ = 0.3 λ = 0.4



λ = 0.4 λ = 0.5

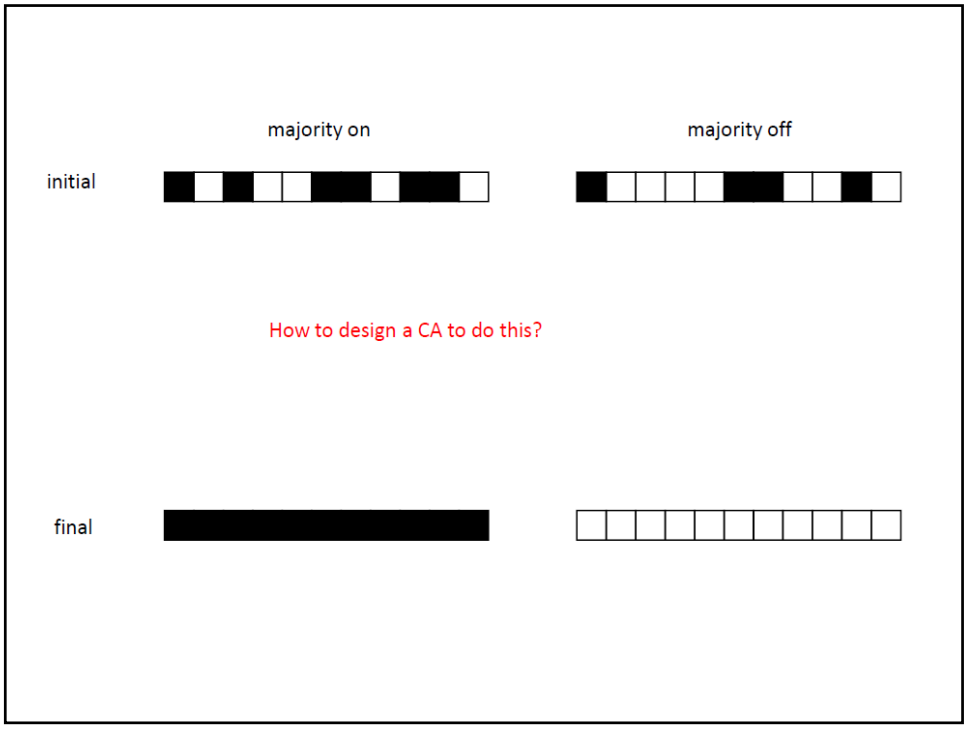
http://math.hws.edu/xJava/CA/EdgeOfChaosCA.html

How to define “potential for computation”?

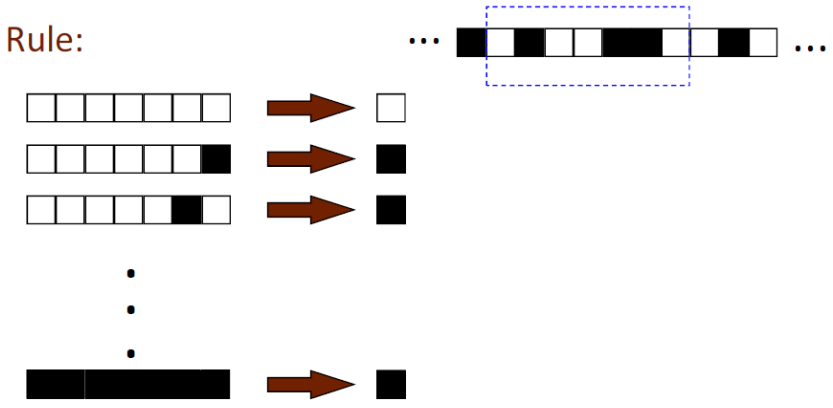
- Adaptation to the Edge of Chaos (N. Packard, 1988)
- Main ideas:
  - Define a task for cellular automata requiring non-trivial computation.
- Use a genetic algorithm to evolve a population of cellular automaton rules, with fitness defined as performance on the task.
- Analyze the distribution of lambda values in the final generation.

A computational task for cellular automata

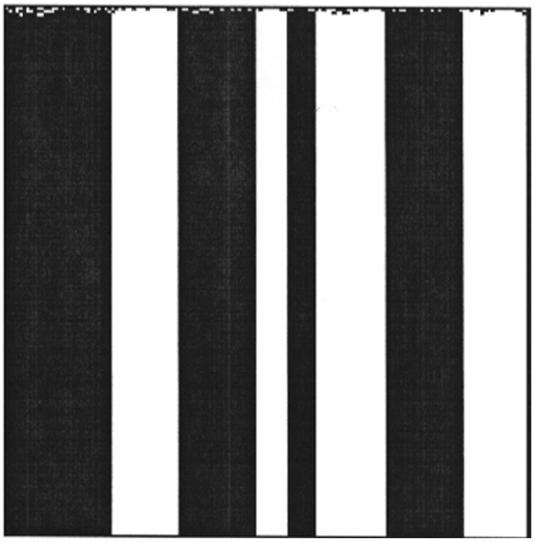
- Design a cellular automata to decide whether or not the initial pattern has a majority of “on”cells.
  - If a majority of cells are initially on, then after some number of iterations, all cells should turn on
  - Otherwise, after some number of iterations, all cells should turn off.



Packard used cellular automata with 6 neighbors for each cell:



A candidate solution that does not work: local majority voting

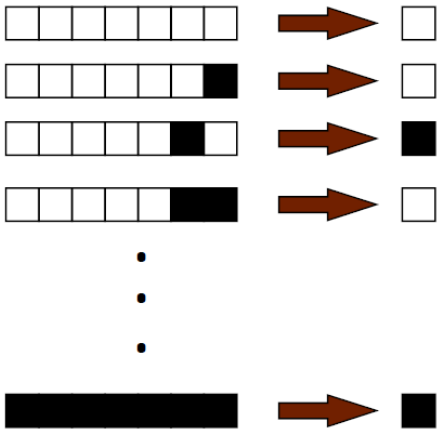


Evolving cellular automata with genetic algorithms

- Create a random population of candidate cellular automata rules
- The “fitness” of each cellular automaton is how well it performs the task. (Analogous to surviving in an environment.)
- The fittest cellular automata get to reproduce themselves, with mutations and crossovers.
- This process continues for many generations.

The “chromosome” of a cellular automaton is an encoding of its rule table:

Rule table:



“Chromosome”:

0
0
1
0
•
•
•
1

Create a random population of candidate cellular automata rules:

rule 1: 00100011000100101111000101001101110000...
rule 2: 00011001101010111111110000111010010 10...
rule 3: 11111000100101010000000111000100101 01...

.
.
.

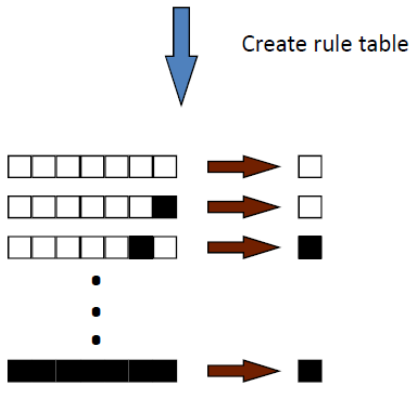
rule 100: 00101110100000011111000001010010111 11...

Calculating the Fitness of a Rule

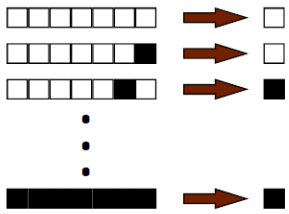
- For each rule, create the corresponding cellular automaton. Run that cellular automaton on many initial configurations.
- Fitness of rule = fraction of correct classifications

For each cellular automaton rule in the population:

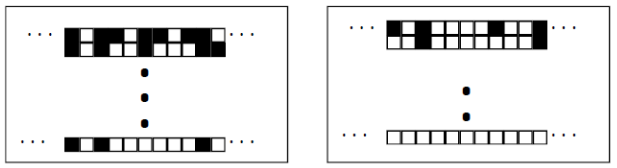
rule 1: 00100011000100101111000101001101110000...1



rule 1 rule table:



Run corresponding cellular automaton on many random initial lattice configurations



incorrect correct

etc.

Fitness of rule = fraction of correct classifications

GA Population:

rule 1: 00100011000100101111000101001101110000... Fitness = 0.5
rule 2: 00011001101010111111110000011101001010... Fitness = 0.2
rule 3: 11111000100101010000000111000100101010... Fitness = 0.4

.
.
.

rule 100:00101110100000011111000001010010111111... Fitness = 0.0

Select fittest rules to reproduce themselves

rule 1: 00100011000100101111000101001101110000... Fitness = 0.5
rule 3: 11111000100101010000000111000100101010... Fitness = 0.4

.
.
.

Create new generation via crossover and mutation:

Parents:

rule 1: 0010001 1000100101111000101001101110000...

rule 3: 1111100 0100101010000000111000100101010...

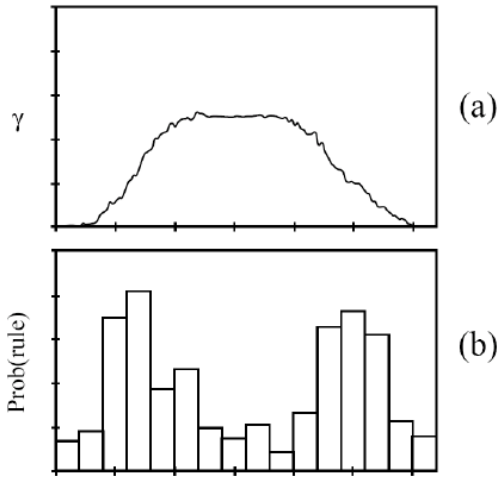
Children:

0010000010010101000000011100010010101...

1111100 1000100101111000101000101110000...

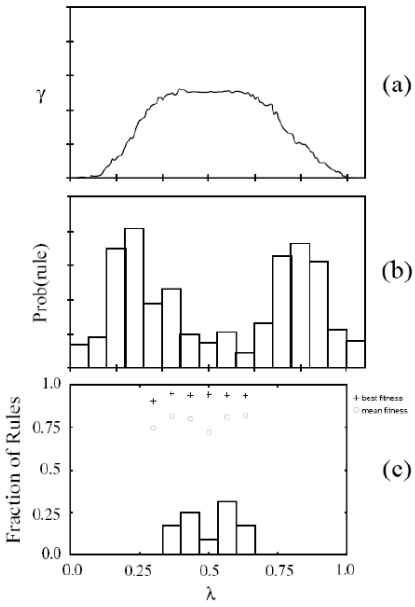
Continue this process until new generation is complete. Then start over with the new generation.

Keep iterating for many generations.



Packard's results

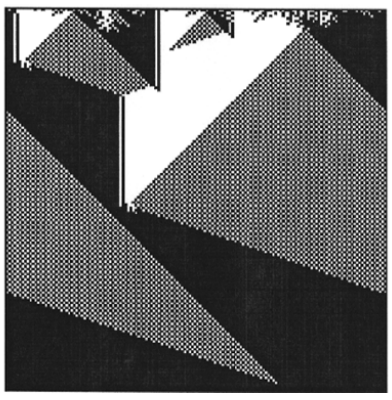
Mitchell et al. results



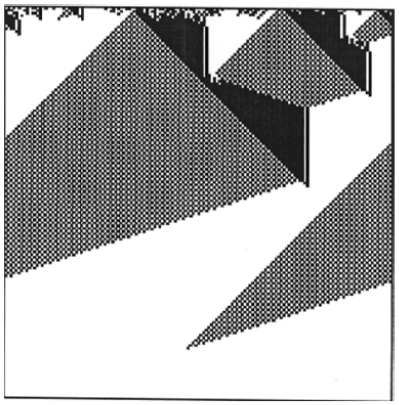
- Notion of "Computation at the Edge of Chaos" is still an open question

A cellular automaton evolved by the genetic algorithm

majority on



majority off

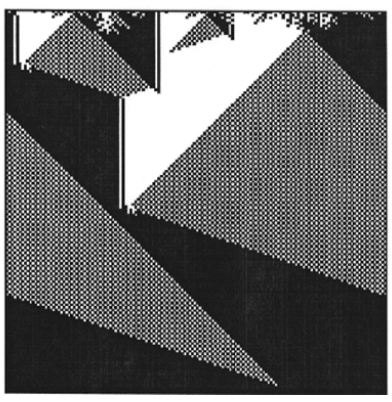


Stephen Wolfram's last problem

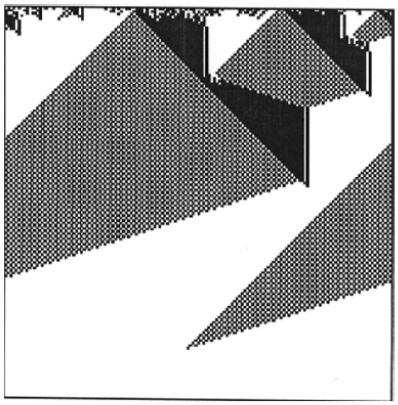
- from ``Twenty problems in the theory of cellular automata'' (Wolfram, 1985):
- 20. What higher-level descriptions of information processing in cellular automata can be given?
- "It seems likely that a radically new approach is needed"

A cellular automaton evolved by the genetic algorithm

majority on

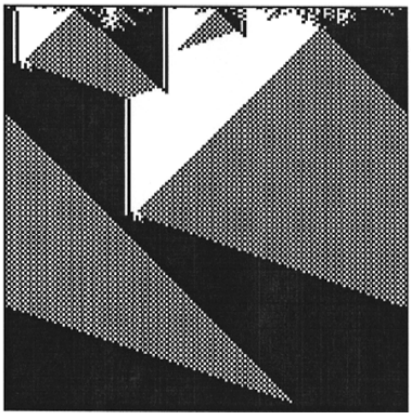


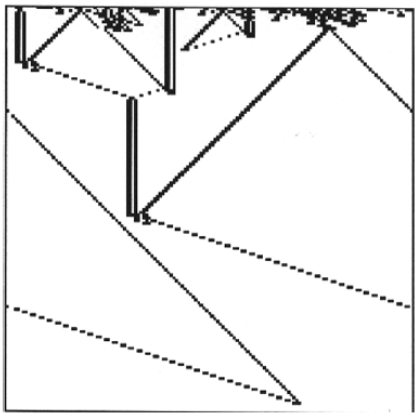
majority off



How do we describe information processing in complex systems?

- Simple patterns filtered out: “Particles”





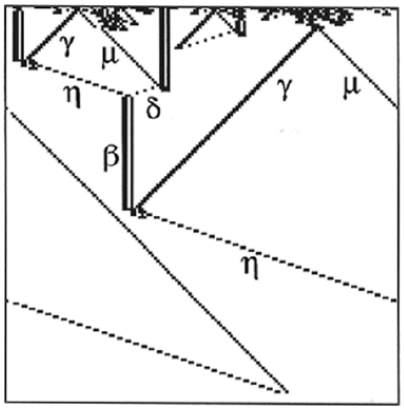
Laws of particle physics

Regular Domains		
$\Lambda^0 = 0^*$	$\Lambda^1 = 1^*$	$\Lambda^2 = (01)^*$

Particles (Velocities)	
$\alpha \sim \Lambda^0\Lambda^1$ (0)	$\beta \sim \Lambda^1 01 \Lambda^0$ (0)
$\gamma \sim \Lambda^0\Lambda^2$ (-1)	$\delta \sim \Lambda^2\Lambda^0$ (-3)
$\eta \sim \Lambda^1\Lambda^2$ (3)	$\mu \sim \Lambda^2\Lambda^1$ (1)

Interactions	
decay	$\alpha \to \gamma + \mu$
react	$\beta + \gamma \to \eta,\ \mu + \beta \to \delta,\ \eta + \delta \to \beta$
annihilate	$\eta + \mu \to \emptyset_1,\ \gamma + \delta \to \emptyset_0$

Particles



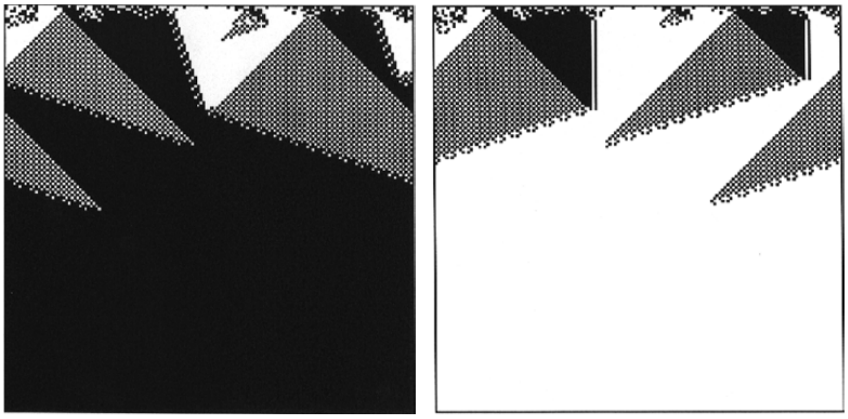
Level of particles can explain:

- Why one CA is fitter than another
- What mistakes are made
- How the GA produced the observed series of innovations
- Particles give an “information processing” description of the collective behavior
  - “Algorithmic” level

How the genetic algorithm evolved cellular automata

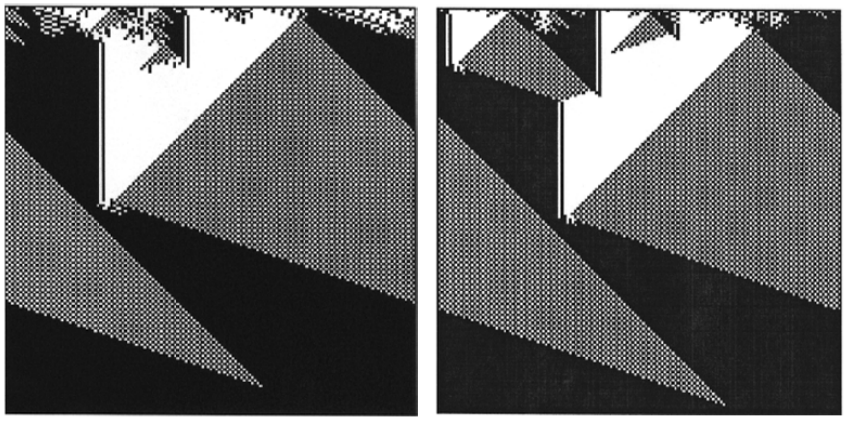
generation 8 generation 13

How the genetic algorithm evolved cellular automata



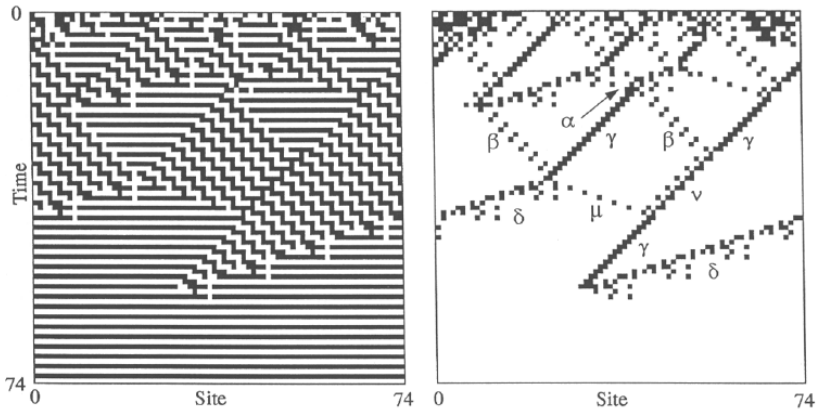
generation 17 generation 18

How the genetic algorithm evolved cellular automata



generation 33 generation 64

Another Task: Synchronization

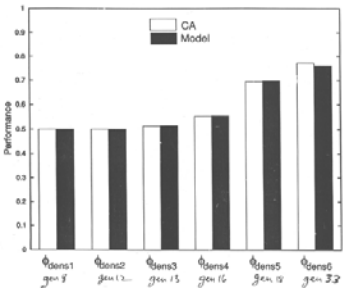


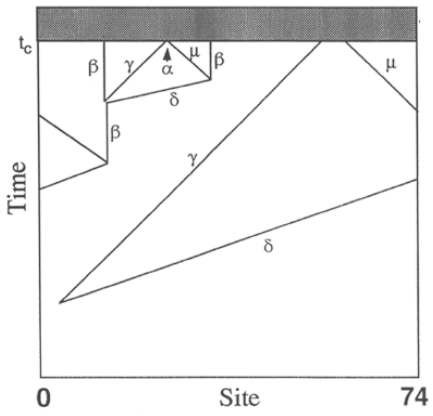
Match between CA and Particle Model

Regular Domains		
$\Lambda^0 = 0^*$	$\Lambda^1 = 1^*$	$\Lambda^2 = (01)^*$

Particles (Velocities)	
$\alpha \sim \Lambda^0\Lambda^1$ (0)	$\beta \sim \Lambda^101\Lambda^0$ (0)
$\gamma \sim \Lambda^0\Lambda^2$ (-1)	$\delta \sim \Lambda^2\Lambda^0$ (-3)
$\eta \sim \Lambda^1\Lambda^2$ (3)	$\mu \sim \Lambda^2\Lambda^1$ (1)

Interactions	
decay	$\alpha \to \gamma + \mu$
react	$\beta + \gamma \to \eta$, $\mu + \beta \to \delta$, $\eta + \delta \to \beta$
annihilate	$\eta + \mu \to \emptyset_1$, $\gamma + \delta \to \emptyset_0$





Lab 5

- http://www.cs.uic.edu/~wilkinson/Applets/automata.html

- Deborah Gordon (ant behavior)
  - Saw last time
  - http://www.ted.com/talks/deborah_gordon_digs_ants.html
- Bonnie Bassler (bacteria quorum sensing)
  - See today
  - http://www.ted.com/index.php/talks/bonnie_bassler_on_how_bacteria_communicate.html