Cellular Automata

What are Cellular Automata?

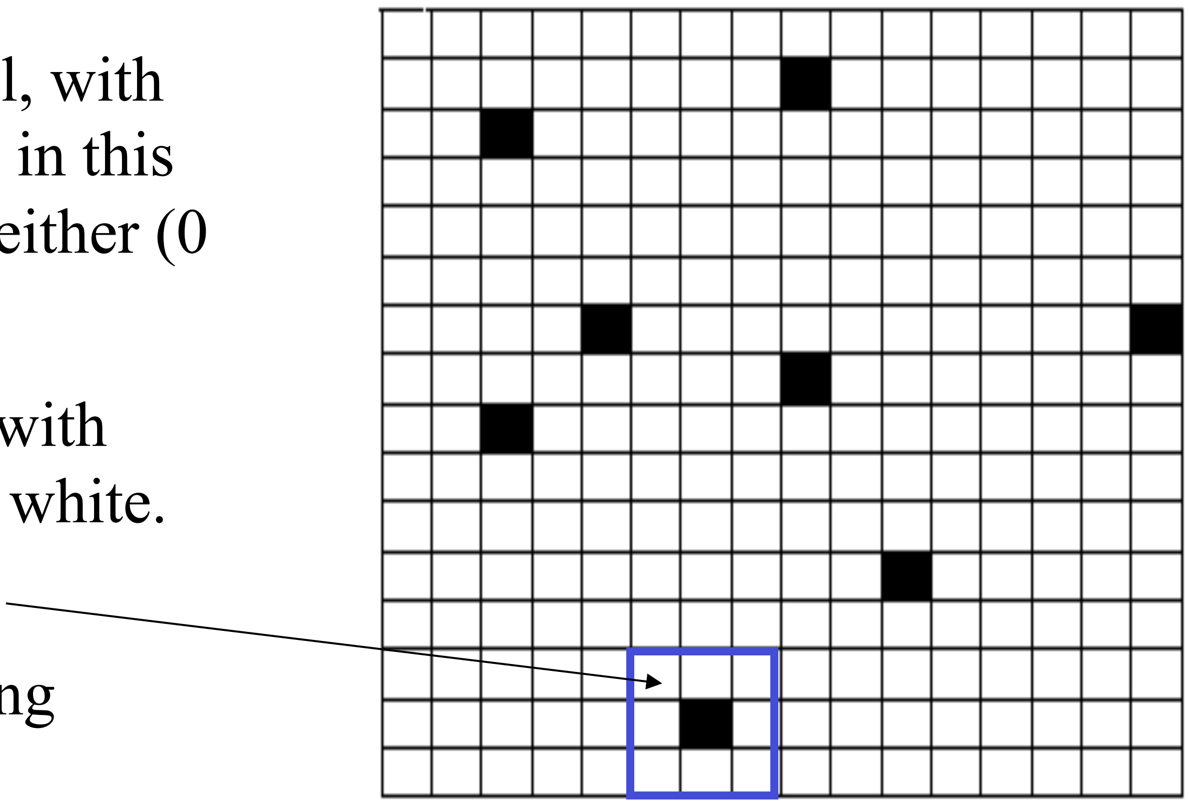
- It is a model that can be used to show how the elements of a system interact with each other.
- Each element of the system is assigned a cell.
- The cells can be:
  - 2-dimensional squares,
  - 3-dimensional blocks
  - or another shape such as a hexagon.

Example of a two-dimensional automaton

Here is a 2-d model, with 256 cells, each cell in this example can be in either (0 or 1) state,

State 1 is encoded with color black, 0 with white.

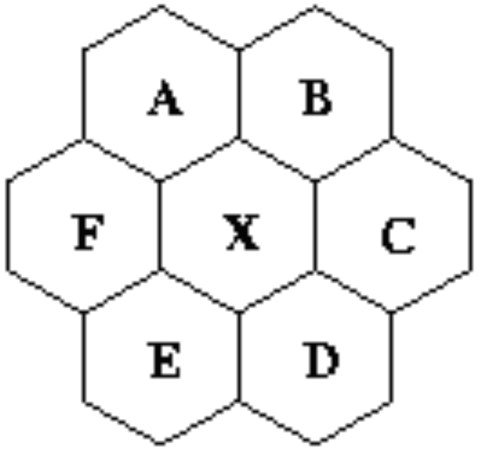
Each cell has eight neighbors (excluding itself).

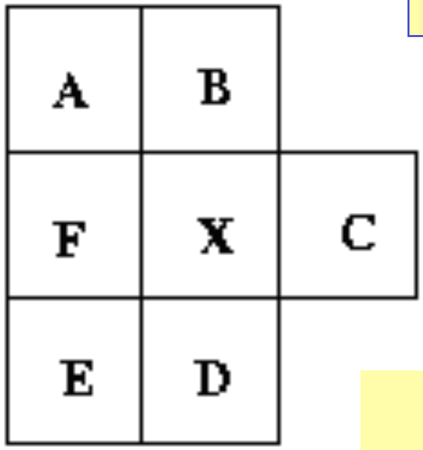


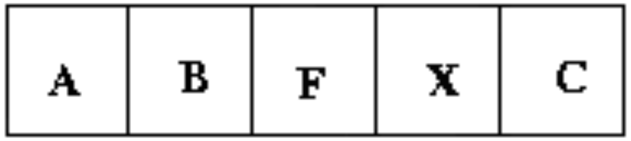
Cellular Automata: the specification

- A neighborhood function that specifies which of the cell’s adjacent cells affect its state.
- A transition function that specifies mapping from state of neighbor cells to state of given cell

Examples of neighborhoods







Different models.

(A,B,C,D,E,F,X) are cells

Each cell has a defined neighborhood.

For example, in a one dimension cellular automaton, a neighborhood of radius one for a given cell would include the cell to the immediate right and the cell to the immediate left.

The cell itself may or may not be included in the neighborhood.

What is the main characteristics of Cellular Automata?

- Synchronous computation
- Infinitely-large grid:
  - but finite occupancy,
  - grows when needed
- Various dimensions (1D, 2D, 3D, …)
- Typically fine-grain
- If cells are distributed, they still need to communicate across boundaries; they communicate once per cycle.

What are the Applications of Cellular Automata?

- Universal computers (embedded Turing machines)
- Self-reproduction
- Diffusion equations
- Artificial Life
- Digital Physics

Some Examples of Application of CA: Simulation Models

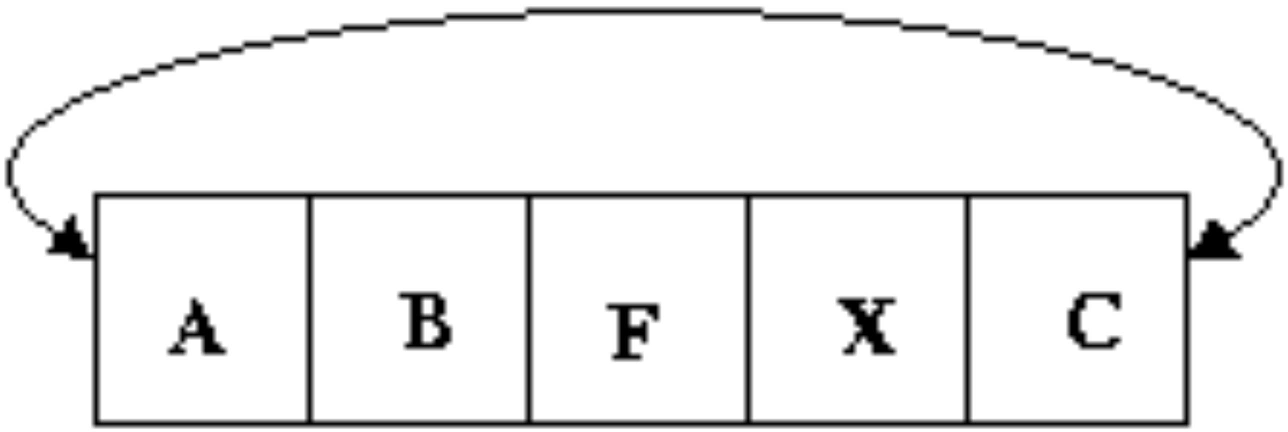
- “Game of Life”
- Gas particles: Billiard-ball model
- Ising model: Ferro-magnetic spins
- Heat equation simulation
- Percolation models
- Wire models
- Lattice Gas models

One-Dimensional Cellular Automaton with Wrap-around

•The cells on the end may (or may not) be treated as "touching" each other as if the line of cells were circular.

If we consider them as they touch each other, then the cell (A) is a neighbor of cell (C)

The way we consider the Ends touch each other



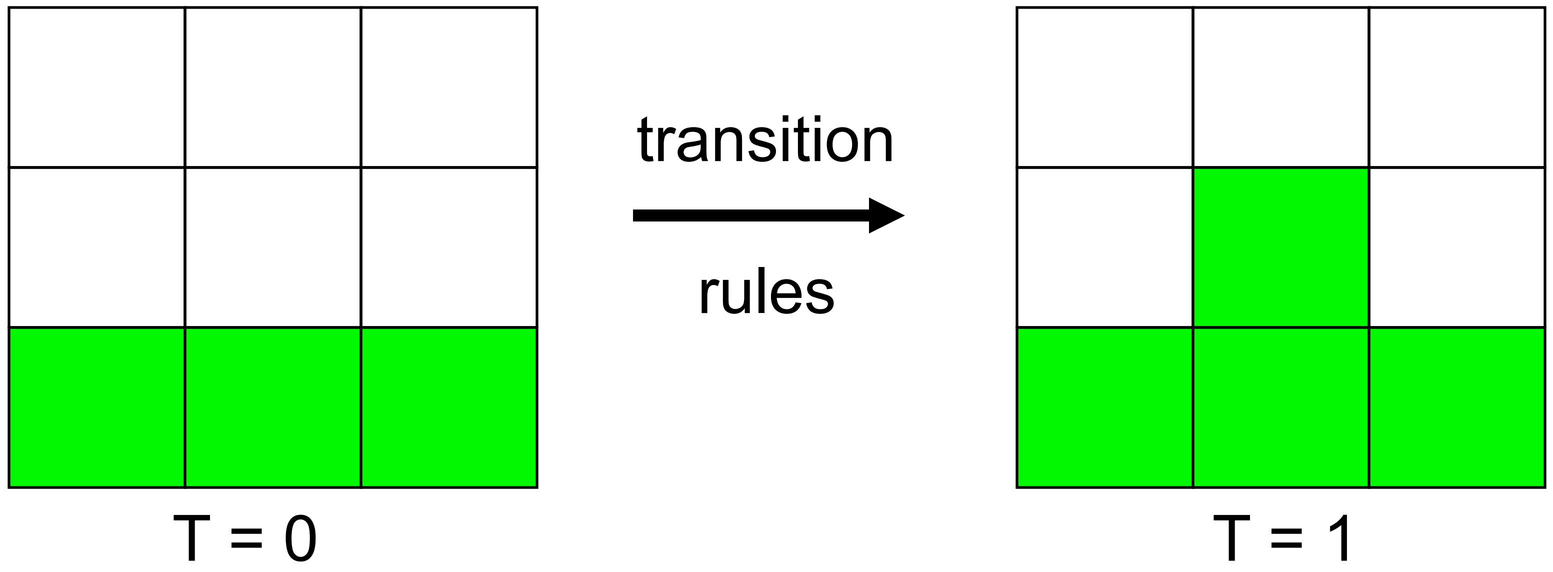
Life - The Game

Life - Conway’s Game of Life

John H. Conway

Life - The Game

- A cell dies or lives according to some *transition rule*



- The world is round (flips over edges)
- How many rules for Life? 20, 40, 100, 1000?

Life - The Game

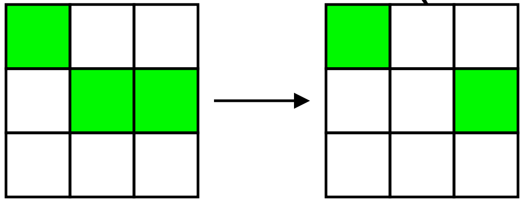
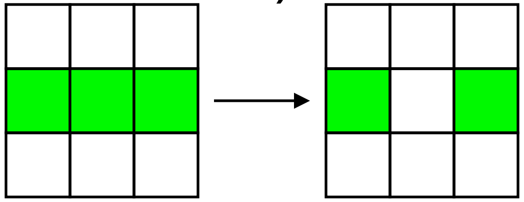
Three simple rules

- dies if number of alive neighbour cells =< 2 (*loneliness*)
- dies if number of alive neighbour cells >= 5 (*overcrowding*)
- lives is number of alive neighbour cells = 3 (*procreation*)

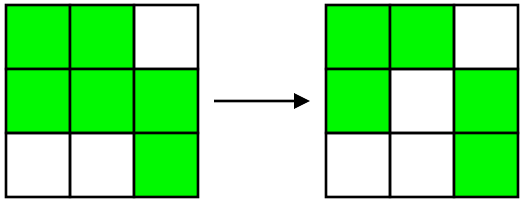
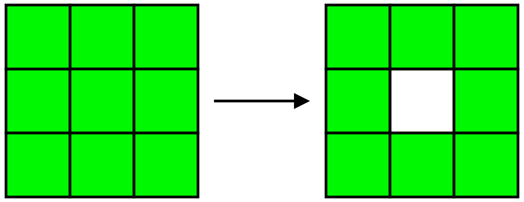
Life - The Game

Examples of the rules

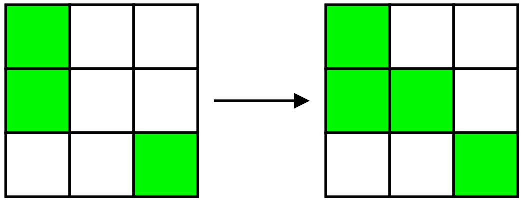
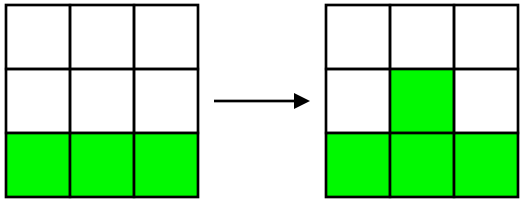
- **loneliness** (dies if #alive =< 2)

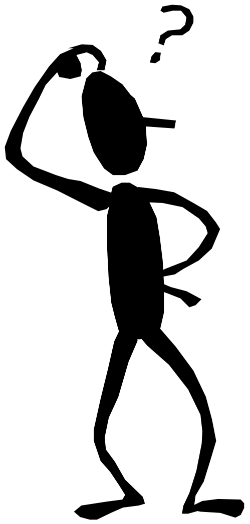
- **overcrowding** (dies if #alive >= 5)

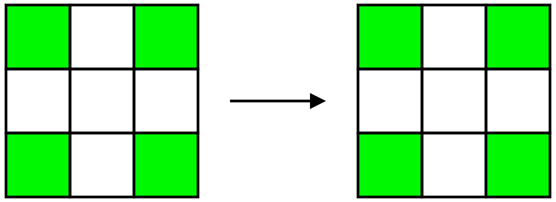



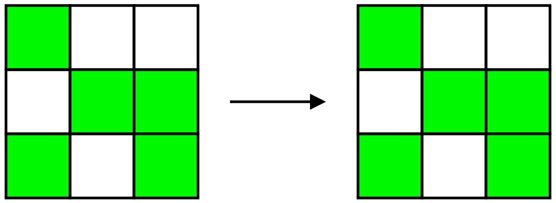
- **procreation** (lives if #alive = 3)

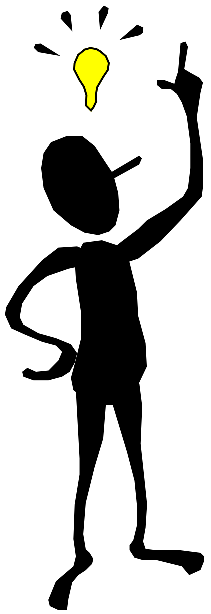



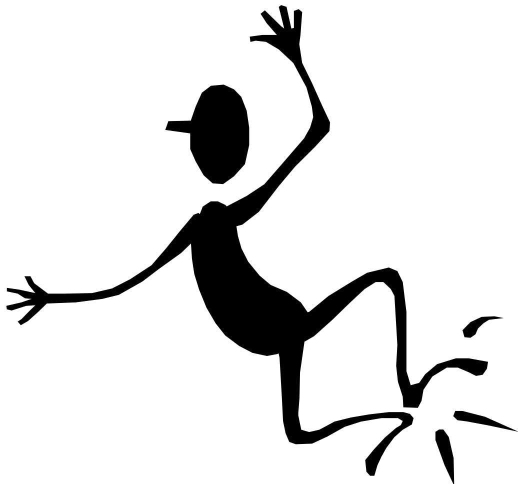
How to design the functions for rules?



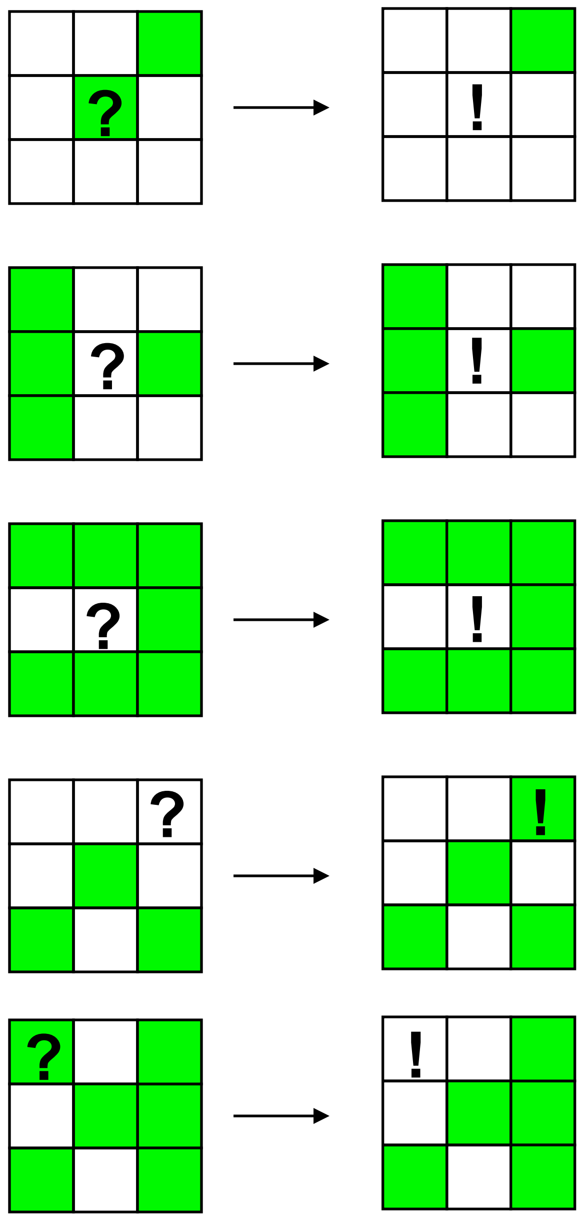


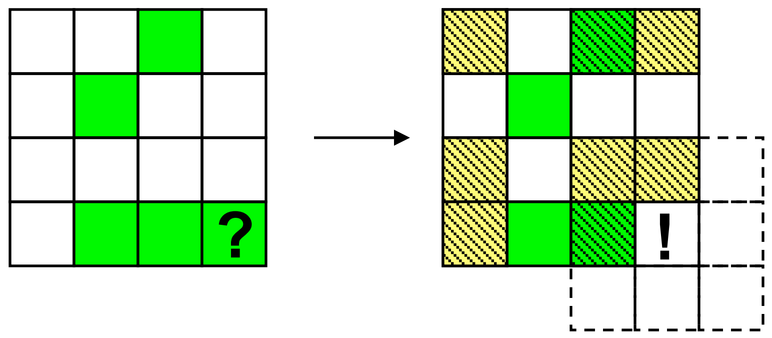


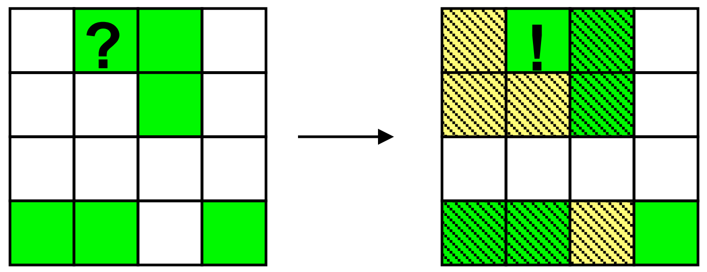




How to design the functions for rules?

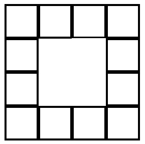


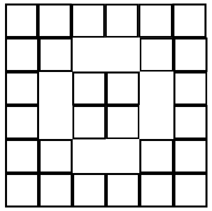


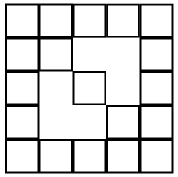


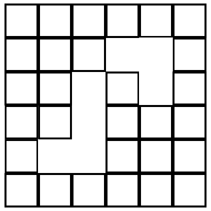
Life - Patterns

Stable

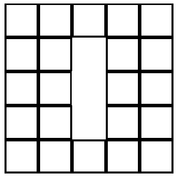

block

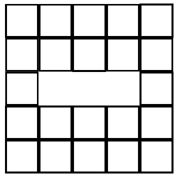

pond


ship

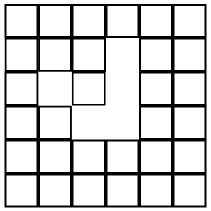

eater

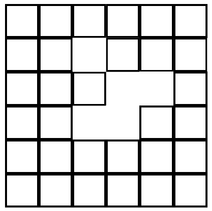
Periodic

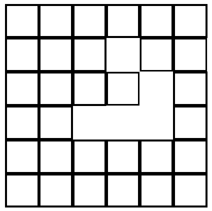

time = 1

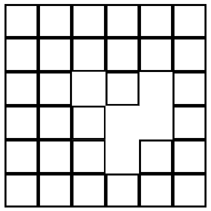

time = 2

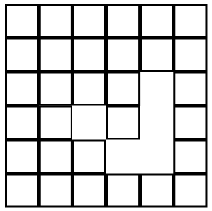
Moving


Time = 1


time = 2

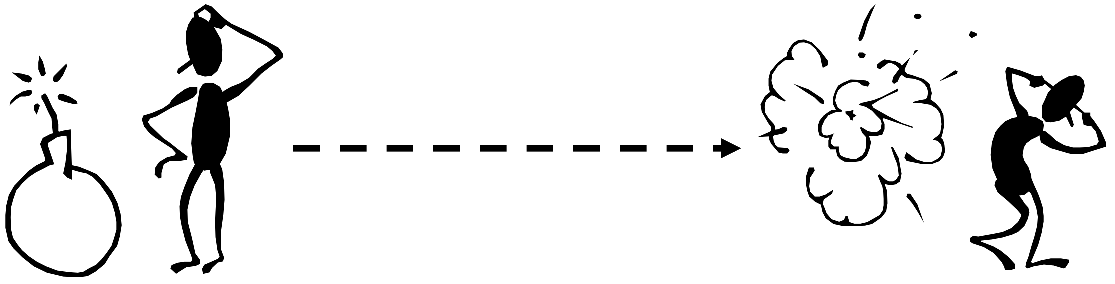

time = 3


time = 4


time = 5

Cellular Automata - Introduction

Traditional science



• Newton laws

• states

• problem: detailed description of states impossible etc etc

Heisenberg principle

• states that it is impossible to precisely know the *speed* and the *location* of a particle

• basis of quantum theory

Beyond Life - Cellular Automata

“A CA is an array of identically programmed automata, or cells, which interact with one another in a neighbourhood and have definite state”

Cellular Automata - Array

"A CA is an array of identically programmed automata, or cells, which interact with one another in a neighbourhood and have definite state"

1	45	34	12	90	4	27	7

H	Q	M	S	W	E	T	G

1	0	1	1	0	1	0	0

🟩		🟩	🟩		🟩		

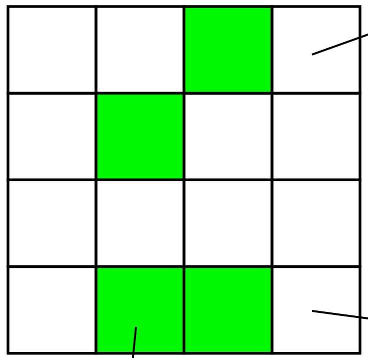
- 1 dimensional
- 2 dimensional

G	O	M	R
A	W	J	D
X	R	E	P
N	I	Z	T

		🟩	
	🟩		
	🟩	🟩	

Cellular Automata - Cells

"A CA is an array of identically programmed automata, or cells, which interact with one another in a neighbourhood and have definite state"



- if #alive =< 2, then die
- if #alive = 3, then live
- if #alive >= 5, then die

- if #alive =< 2, then die
- if #alive = 3, then live
- if #alive >= 5, then die

- if #alive =< 2, then die
- if #alive = 3, then live
- if #alive >= 5, then die

Cellular Automata - Interaction

“A CA is an array of identically programmed automata, or cells, which interact with one another in a neighbourhood and have definite state”

<u>Da rulez</u>

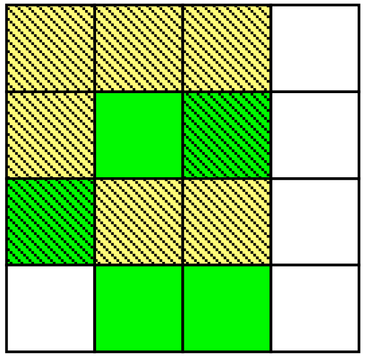
if #alive =< 2, then die

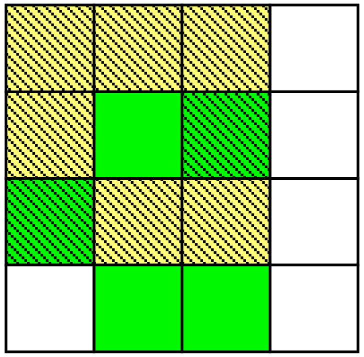
if #alive = 3, then live

if #alive >= 5, then die

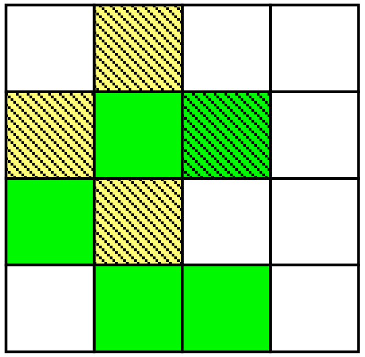
Cellular Automata - Neighborhood

"A CA is an array of identically programmed automata, or cells, which interact with one another in a neighborhood and have definite state"





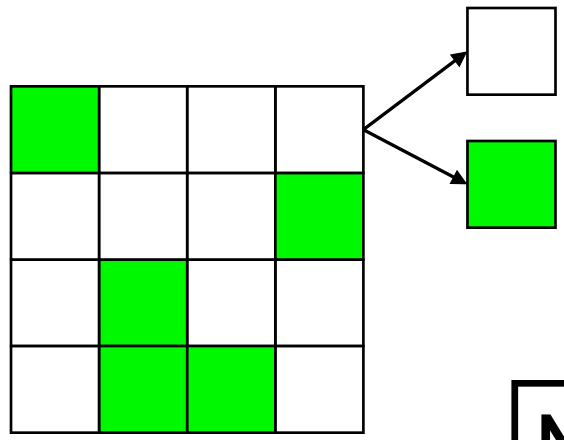
Moore neighborhood



Neumann neighborhood

Cellular Automata - States

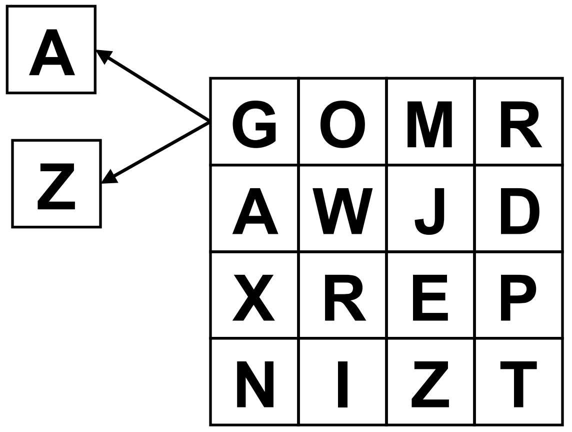
"A CA is an array of identically programmed automata, or cells, which interact with one another in a neighborhood and have definite state"



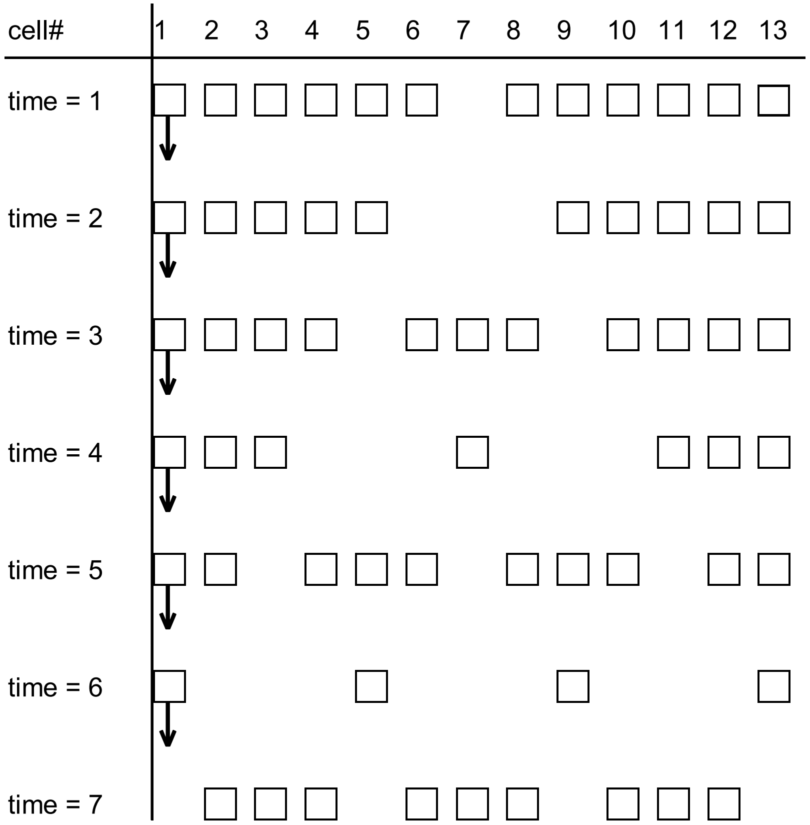
2 possible states: ON OFF

Never infinite!

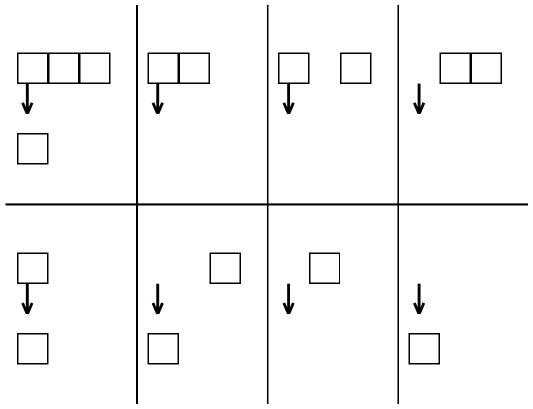
26 possible states: A … Z



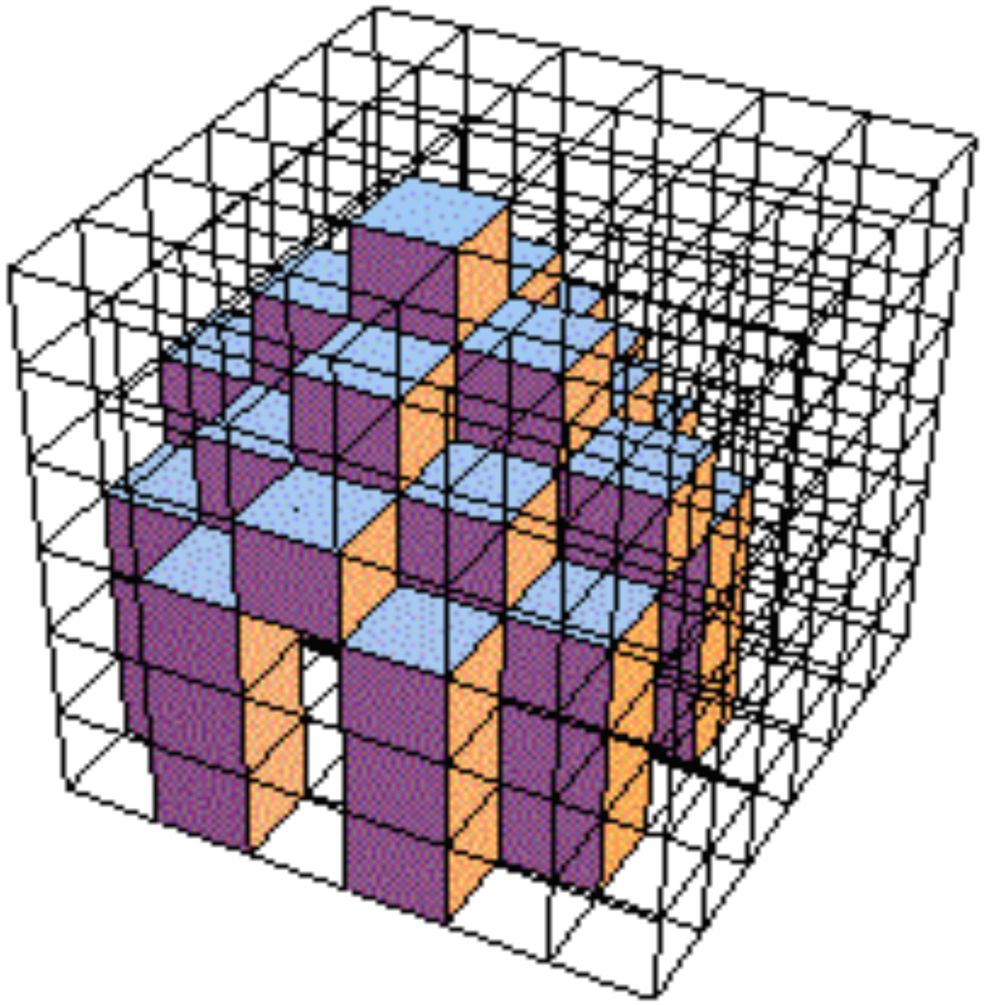
Cellular Automata - Simple 1D example



The rules



Example of a three-dimensional cellular automaton



One difficulty with three-dimensional cellular automata is the graphical representation (on two-dimensional paper or screen)

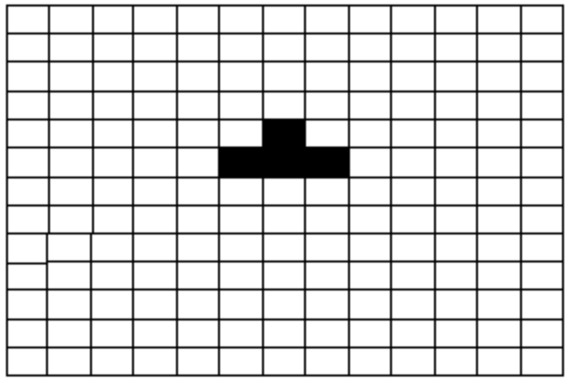
Example of description of a cellular automaton

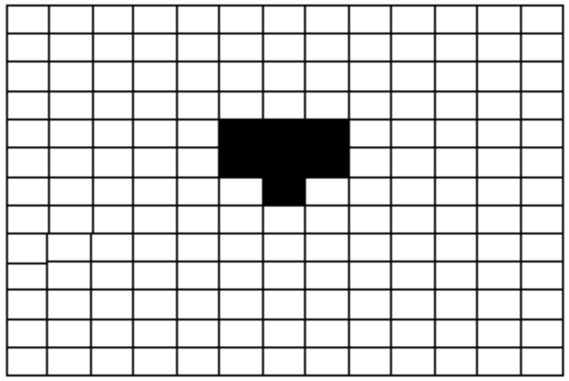
In the initial configuration of the cellular automata, each cell is assigned a "starting" value from the range of possible values.

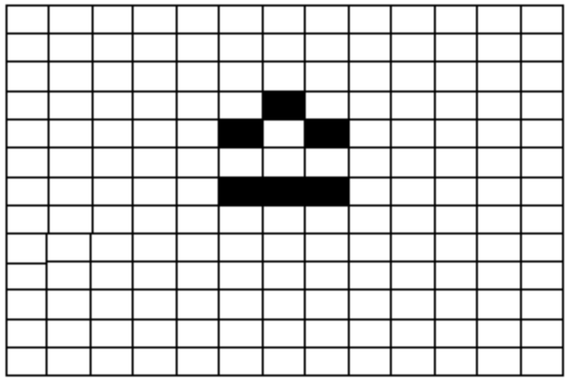
For example, if the range of possible values is 0 or 1, then each cell would be assigned a 0 or a 1 in the initial configuration.

Each value represents a color to the computer.

Each cell is associated a transition rule.


Initial


Step1

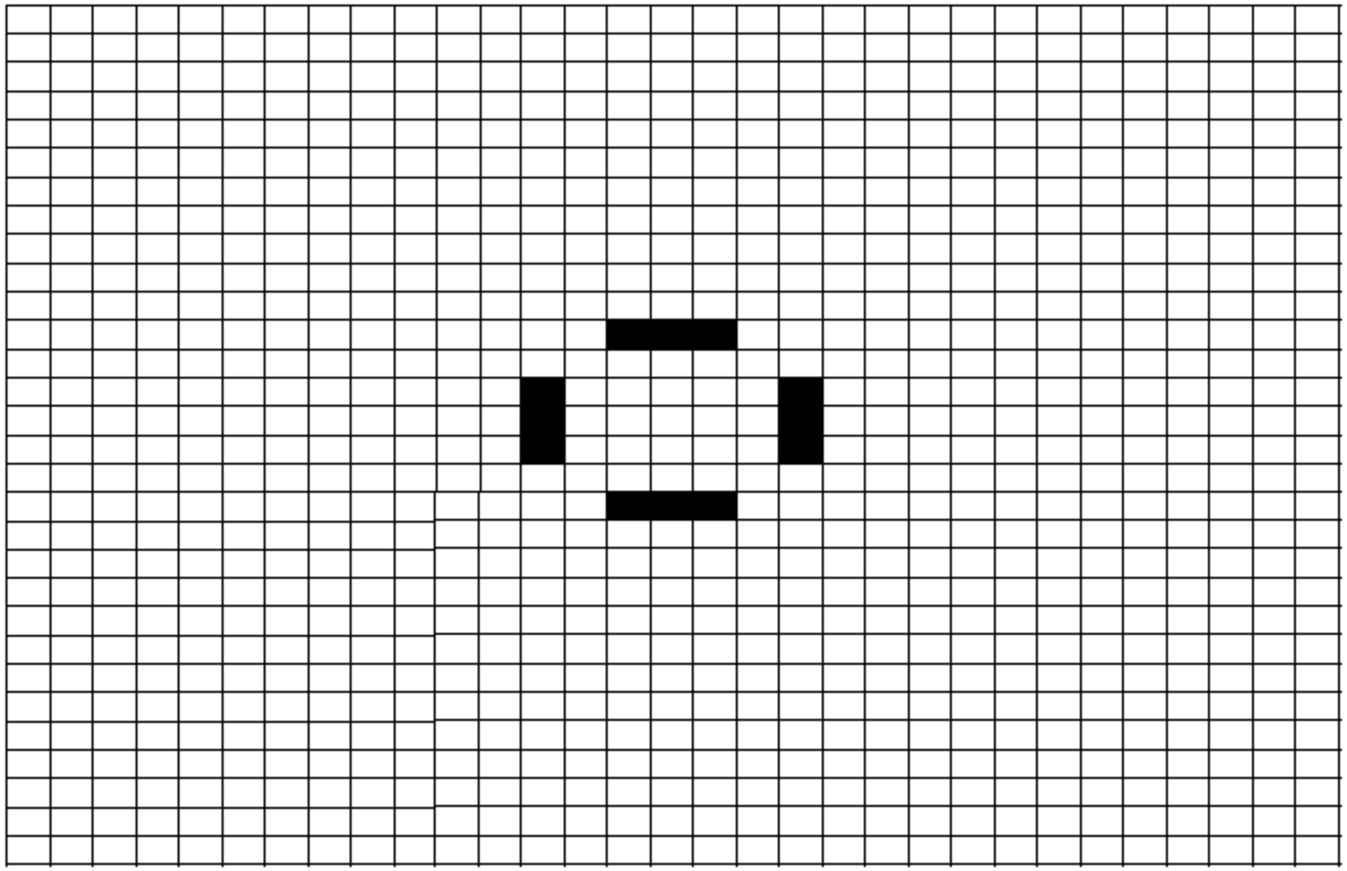

Step2

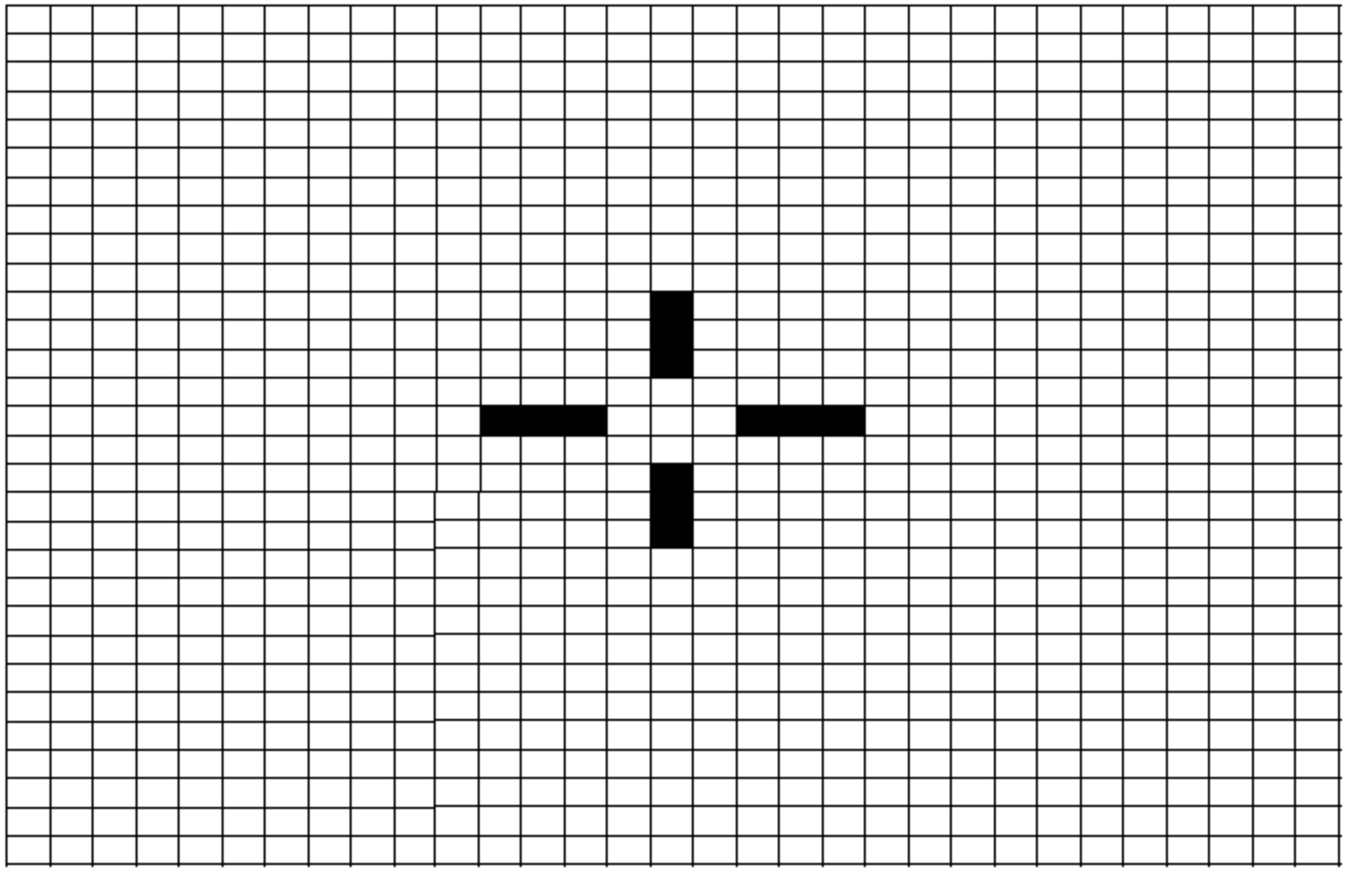
Here is an example

In this example, the initial configuration for all the cells is state 0, except for 4 cells in state 1.

- The transition rule for this example, is :
  - a cell stays in state 1 (black), if it has two or three black neighbors.
  - a cell changes to black, if it has exactly three black neighbors.

The next slide shows animation for this example





In this example:

- Each cell has 8 neighbors,
- Each cell can be in one possible value at any given time,
  - the transition takes place in discrete times
    - it helps to imagine a clock feeding all the cells.
- Each State is encoded with a unique color,
  - The transition rule takes as input the present states (i.e., the present values) of all of the cells in a given cell's neighborhood and generates the next state (i.e., the next value) of the given cell.
  - When applied to all of the cells individually in a cellular automata, the next state of the whole cellular automata is generated from the present state.
  - Then the next state of the cellular automata is copied to the (new) present state and the process is repeated for as many clock cycles as desired.

Let us now find some real-life examples!!!

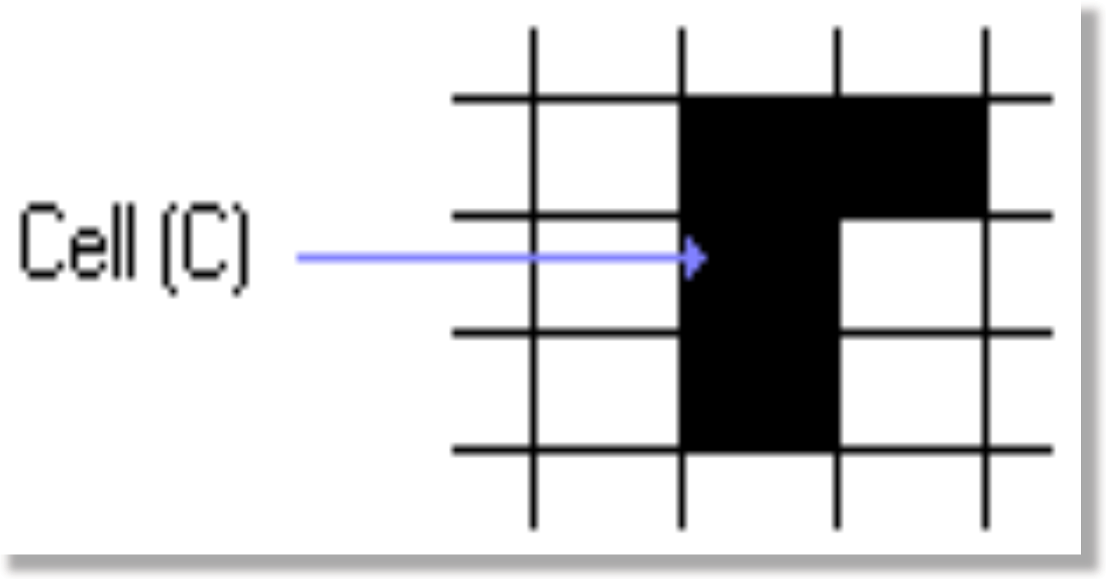
Example of a Cellular Automaton: VOTE

- Vote is an example of the simplest possible kind of eight-neighbor CA.
- Vote is so simple because:
  - (1) Vote is a "one-bit rule" and,
  - (2) Vote is "totalistic.”
- What do these expressions mean?

Example of a Cellular Automaton: VOTE

- **NineSums:**
  The NineSum for a cell (C) is the sum of 1' s in all the surrounding cells (neighbors including cell (C)).
- **EightSum:**
  EightSum for a cell (C) is the sum of 1' s in all the surrounding cells (neighbors excluding cell (C)).

Example of a Cellular Automaton: VOTE



- Let’s consider the above example to explain what NineSume & EightSum are.
- In this example, each cell can be in either 0 or 1 state.
- Cell C has 8 neighbors, 3 of them are in state 1,
  - Then the EightSum for cell C is 3, NineSume is 4.

Example of a Cellular Automaton: VOTE

- Vote is a one-bit rule.
- The cells of Vote have only two possible states: on or off, zero or one.
- Choosing between two options requires one bit of information, and this is why we call Vote a one-bit rule.

Example of a Cellular Automaton: VOTE

- Vote is totalistic.
- A totalistic rule updates a cell C by forming the EightSum of the eight neighbors, adding in the value of C itself to get the full NineSum, and then determining the cell's new value strictly on the basis of where the NineSum lies in the range of ten possibilities 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
- Under a totalistic rule, a cell's next state depends only on the total number of bits in its nine-cell neighborhood.
- Let us present more detail of this example.

Example of a Cellular Automaton: VOTE

How many different eight-neighbor 1-bit totalistic rules are there?

A rule like this is completely specified by a ten-entry lookup table which gives the new cell value for each of the ten possible neighborhood NineSums.

Each of the entries has to be 0 or 1, so filling in such a lookup table involves making ten consecutive binary decisions, which can be done in 2^{10} different ways.

Example of a Cellular Automaton: VOTE

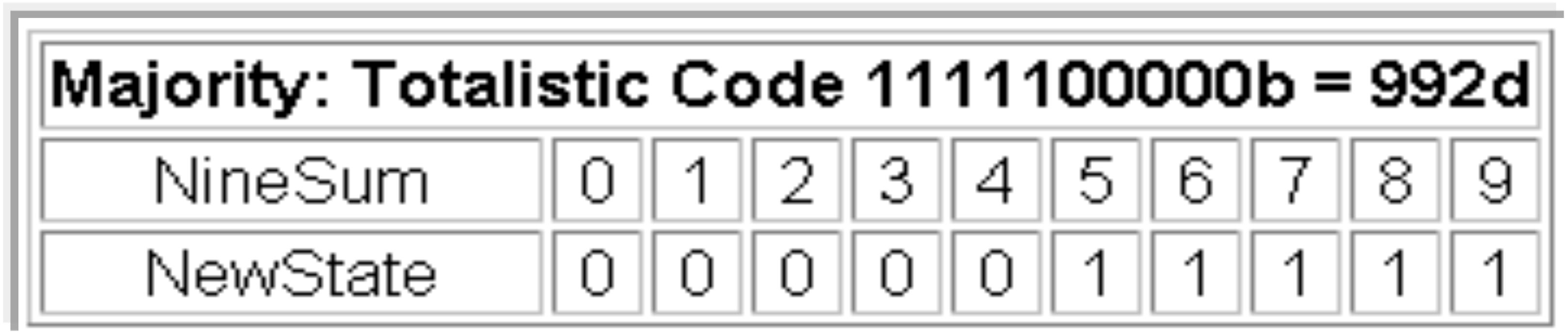
Each time a cell is updated, the NineSum of the cell and its eight neighbors is formed.

The idea behind Vote's rule is that if most cells in your neighborhood are 1, then you go to 1, and if most cells in your neighborhood are 0, then you go to 0.

What do we mean by "most cells in your neighborhood?"

Since there are nine cells in your neighborhood, the most obvious interpretation is to assume that "most" means "five or more".

Here is the lookup table for this simple majority rule.



Majority: Totalistic Code 1111100000b = 992d

NineSum	0	1	2	3	4	5	6	7	8	9
NewState	0	0	0	0	0	1	1	1	1	1

Many "models of life" can be created like this that illustrate congestion, scarcity of resources, competing species, etc.

At this time we have an idea about what the Cellular Automata is.

Let us now try to get closer to the basic digital logic aspects

and find a different definition for Cellular Automata.

Another definition of a Cellular Automaton

- Let us first try to define the cellular automata as follows:
  - it is a Finite State Machine, with one transition function for all the cells,
  - this transition function changes the current state of a cell depending on the previous state for that cell and its neighbors.
- All we need to do is to:
  - *design the transition function,*
  - *set the initial state.*
- Cells here use Flip-Flops.
  - In general, registers.

What is the advantage of the CA over standard FSM?

- One advantage is that each cell uses as many data as number of neighbor cells to calculate its next state.
- Next state represents an information that will be available for all neighbors including itself,
- The goal is to design the transition function for these cells
- There is no State Assignment problem because all cells are the same
- Transition function is the same for all cells.
- But the complexity is to design the Transition function that fits our application!!!

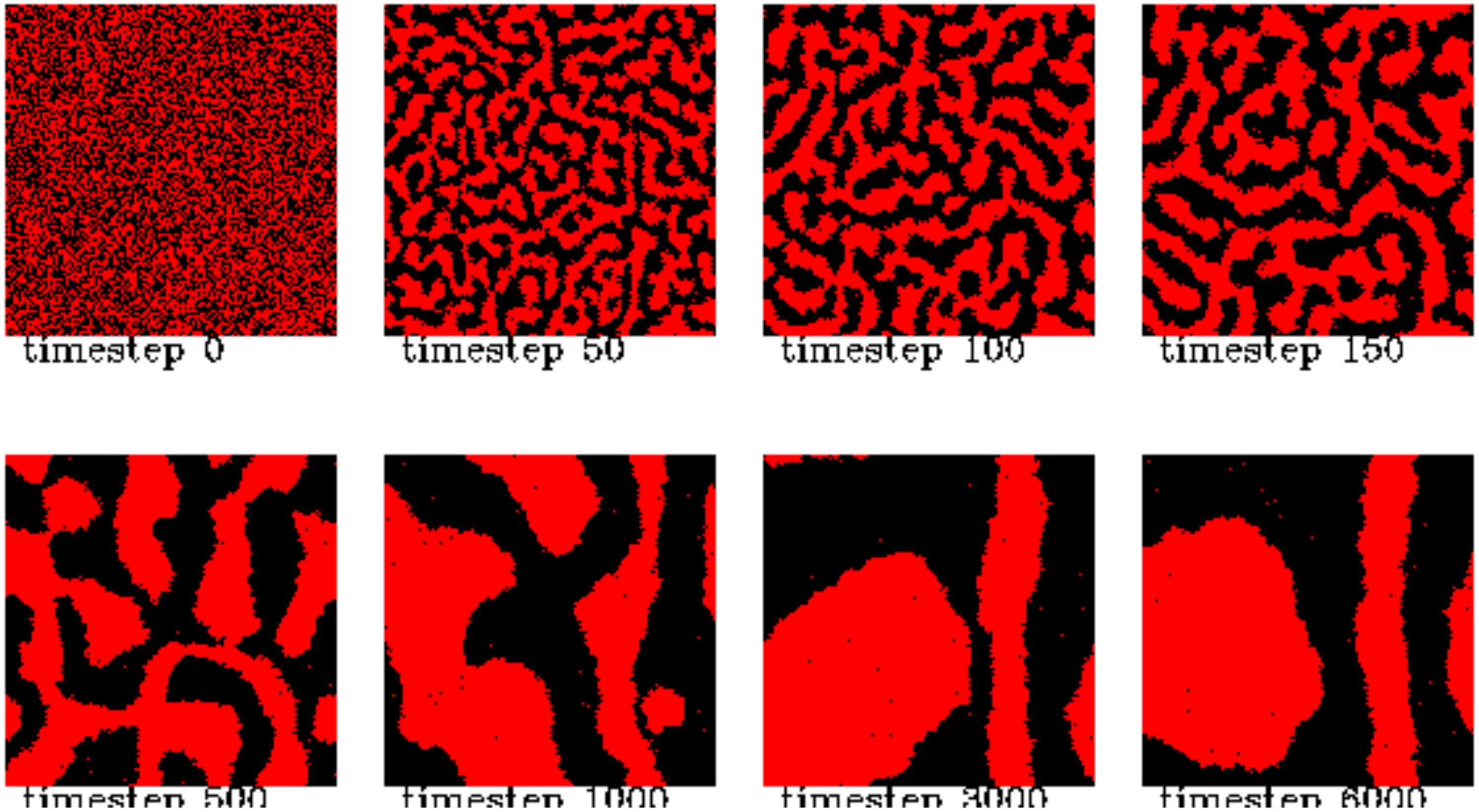
Formally, a CA with limited size is an FSM, but this model is not practical because of too many states of FSM, Cartesian Product of individual CA states.

Cellular Automata are often used to visualize complex dynamic phenomena

- Images that represent evolutions of states of different cellular automata are used:
  - each cell can be in state (0 or 1or 2 …..or n),
  - each state is encoded with a unique color,
  - each cellular automata has its own transition rule.

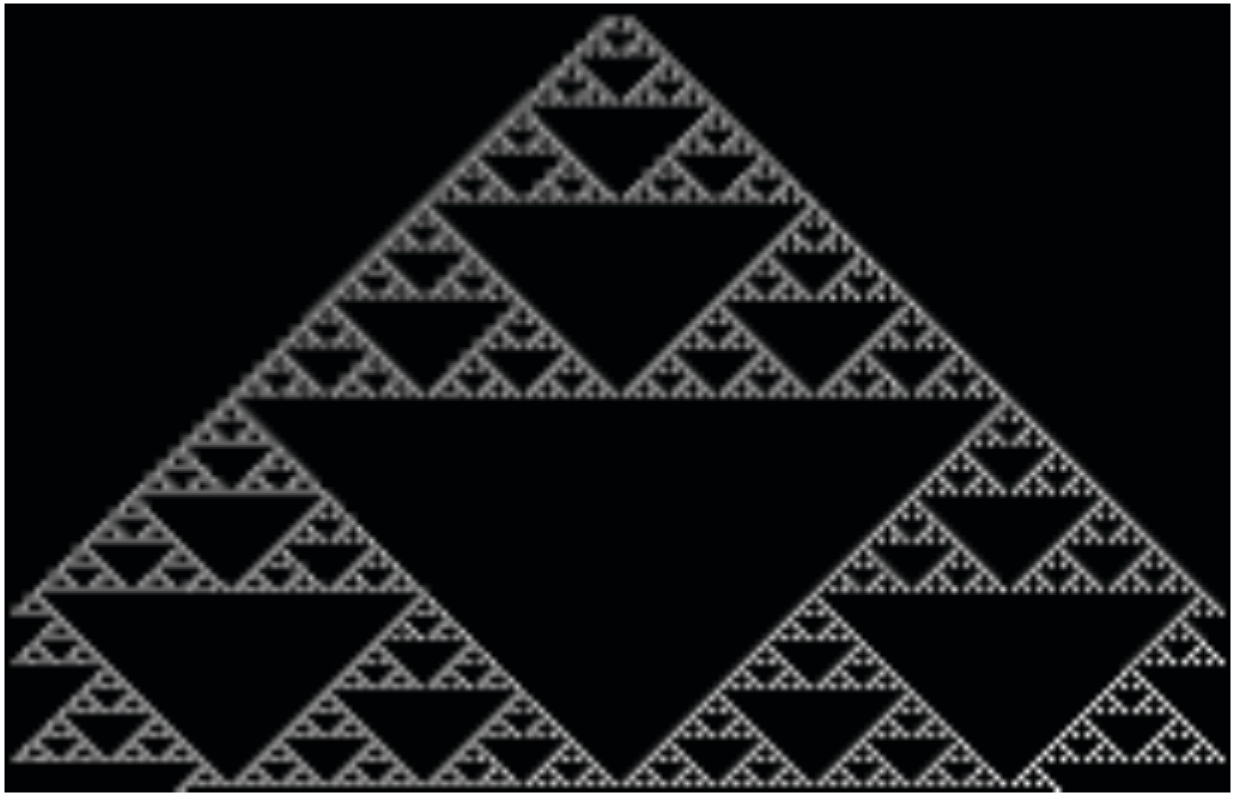
Oil&Water Simulation

(Bruce Boghosian, Boston Univ.)

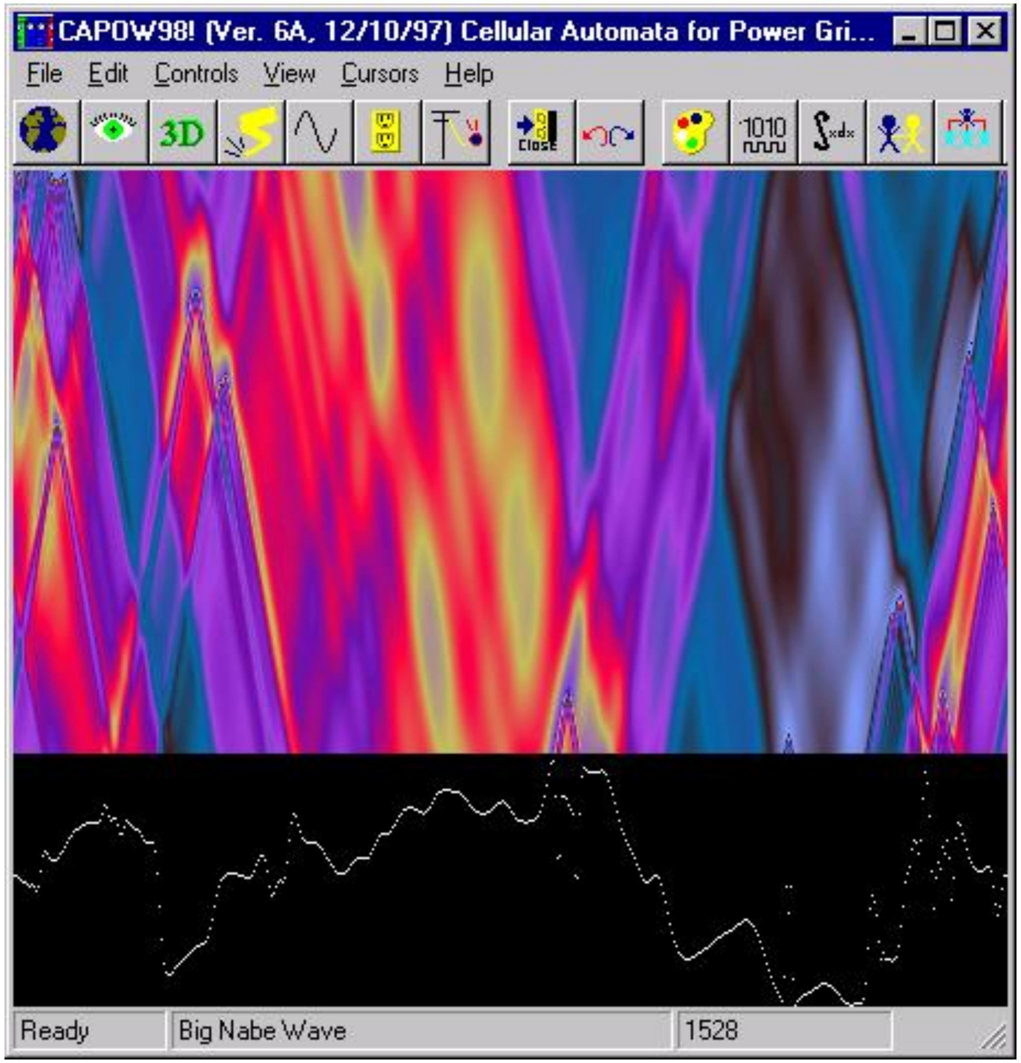


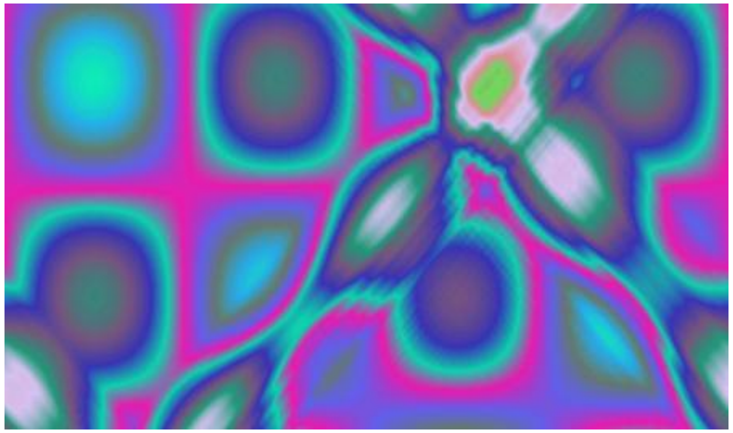
http://physics.bu.edu/~bruceb/MolSim/

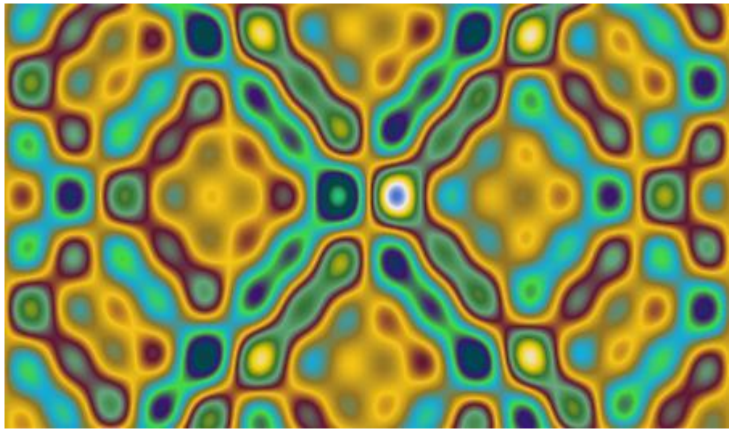
Cellular Automata - Pascal Triangle



Cellular Automata - Beauty

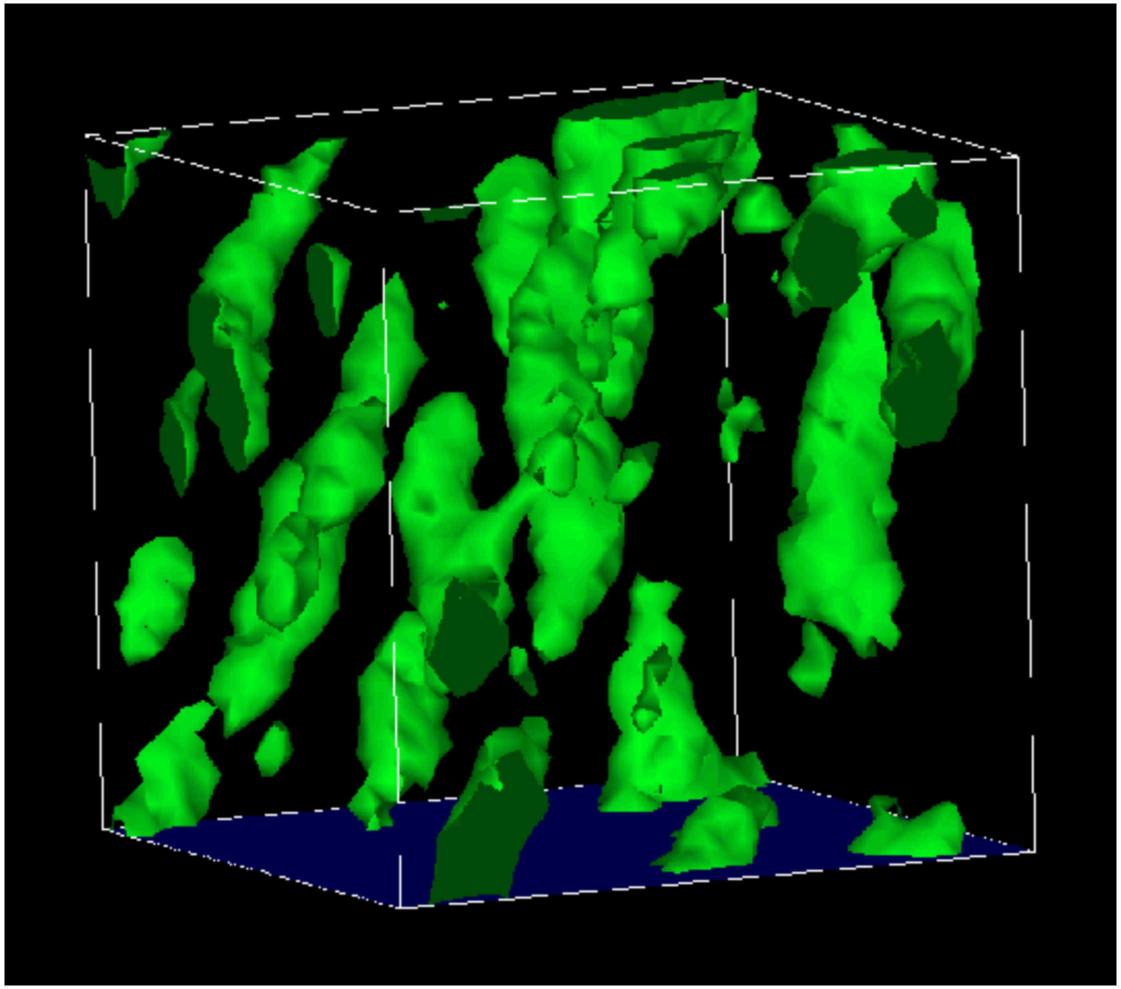

CAPOW98! (Ver. 6A, 12/10/97) Cellular Automata for Power Gri...
File Edit Controls View Cursors Help
3D
1010
Close
Ready
Big Nabe Wave
1528

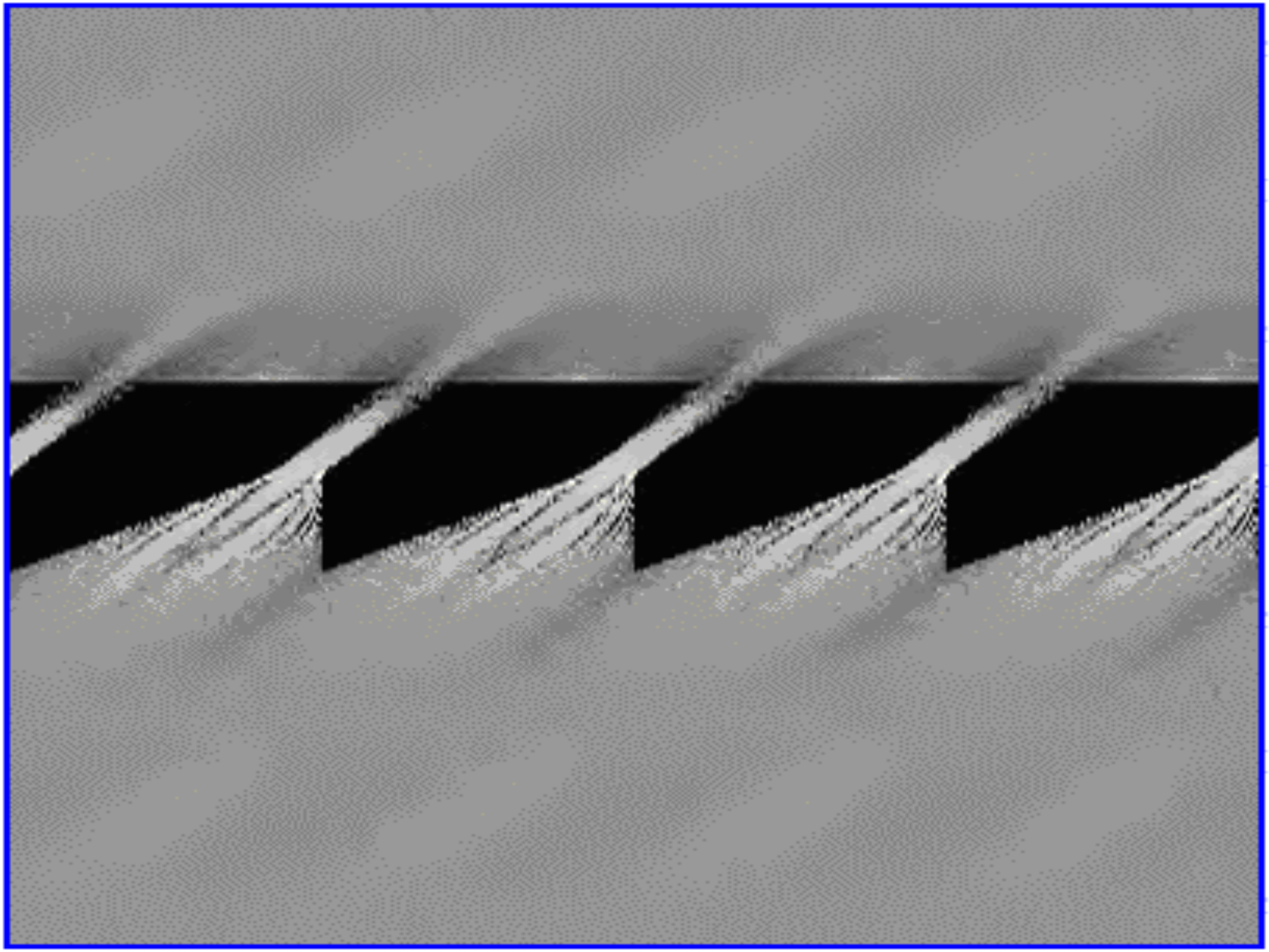


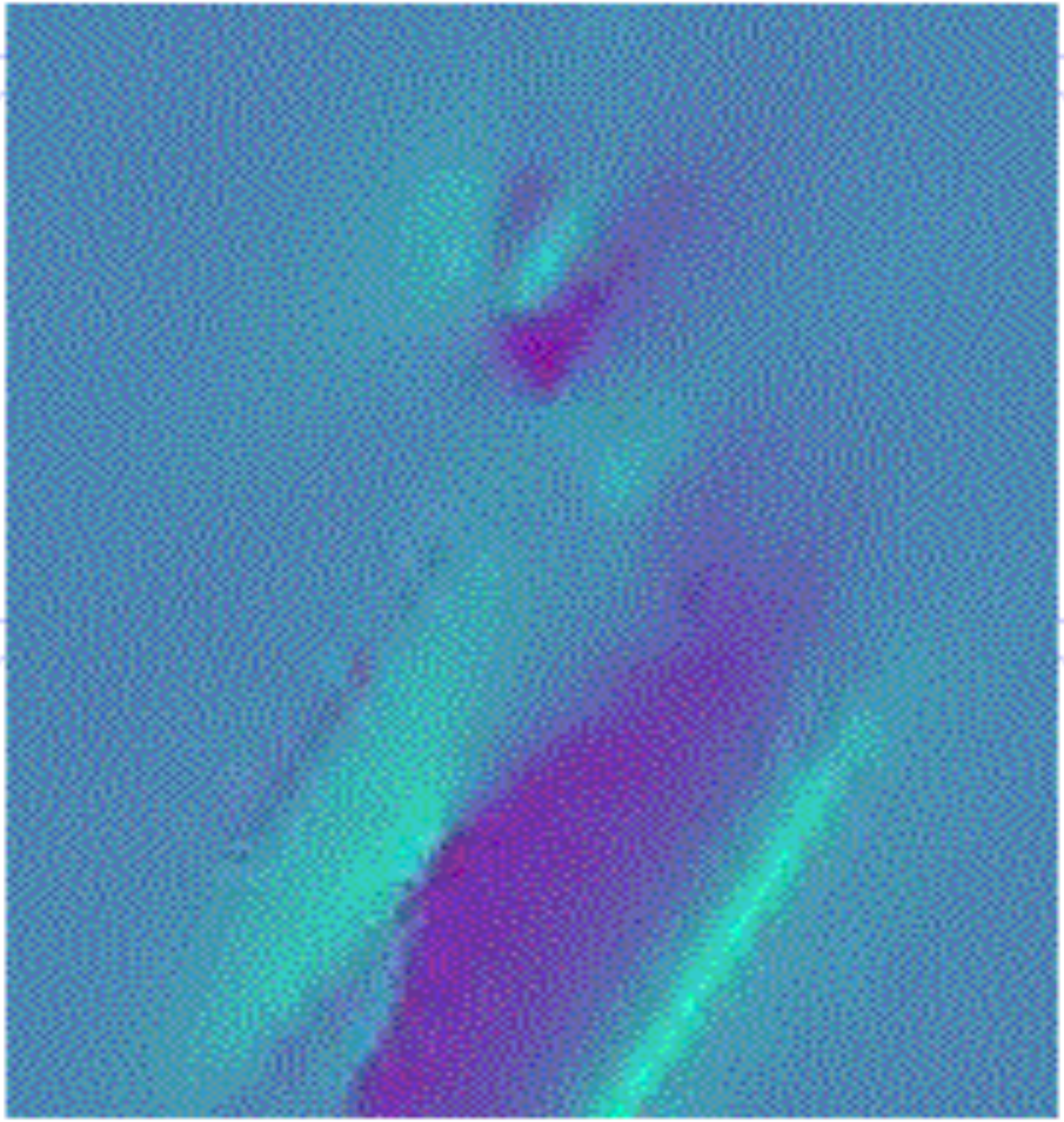


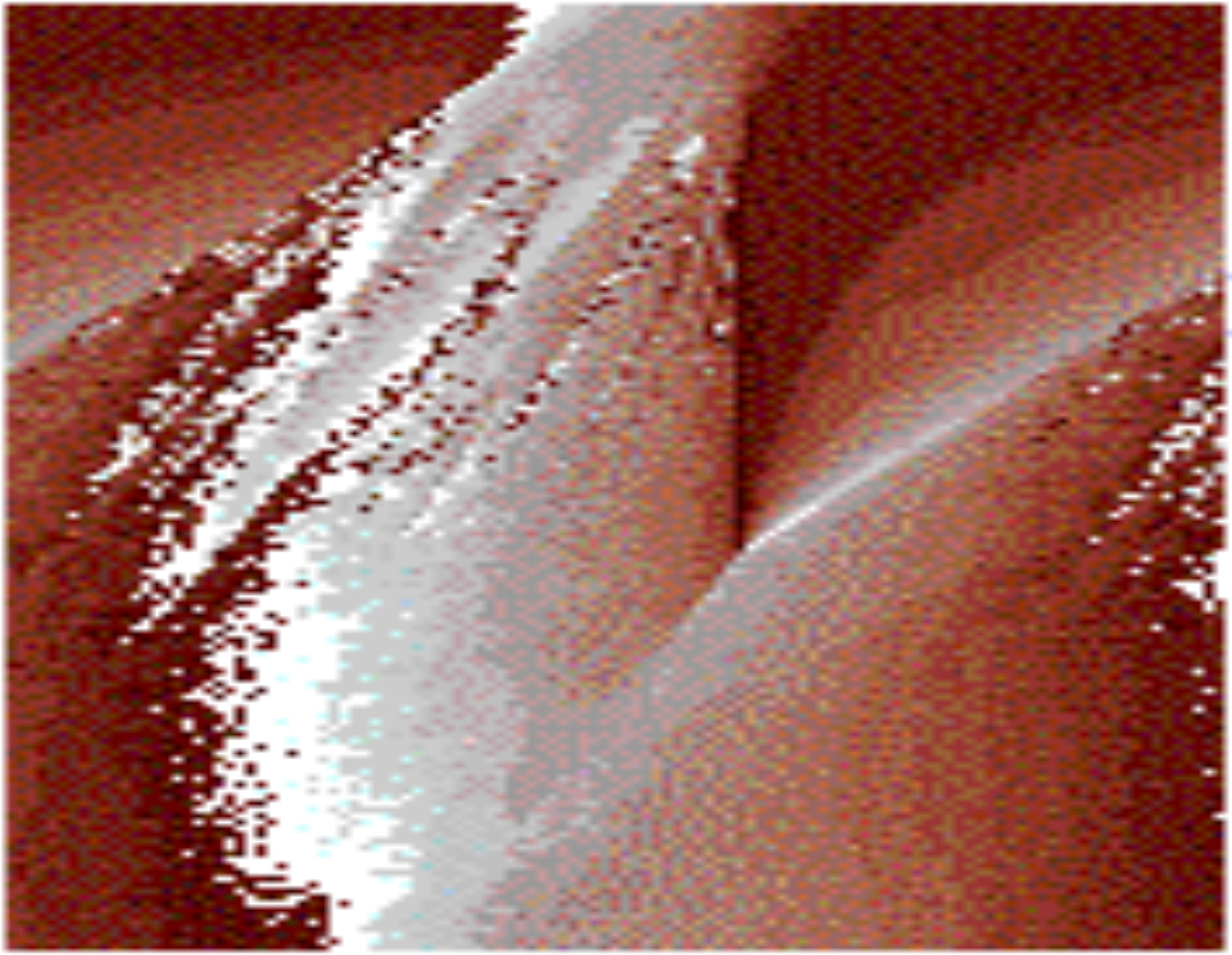
Surfactant Formation

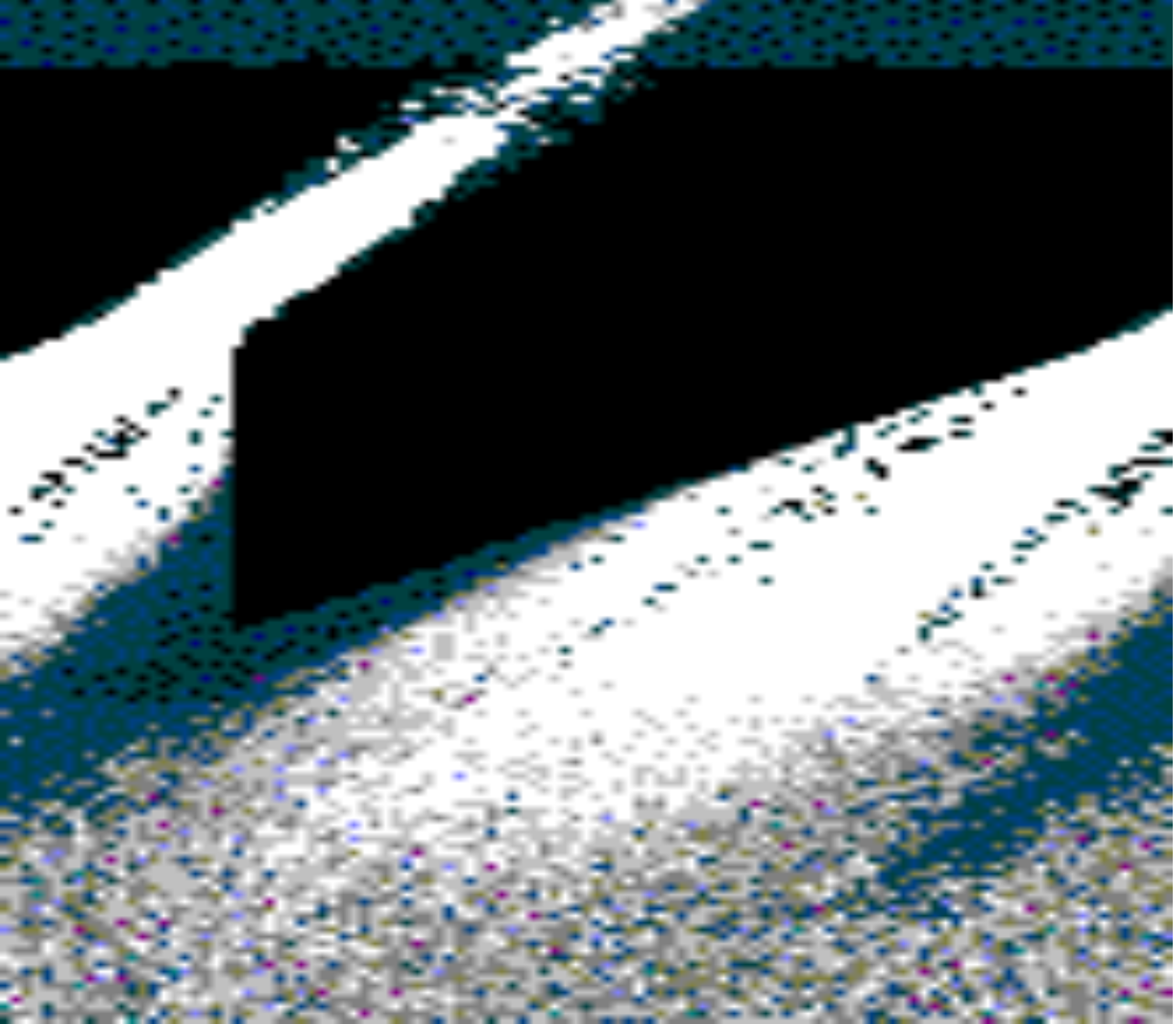
(Bruce Boghosian, Boston Univ.)









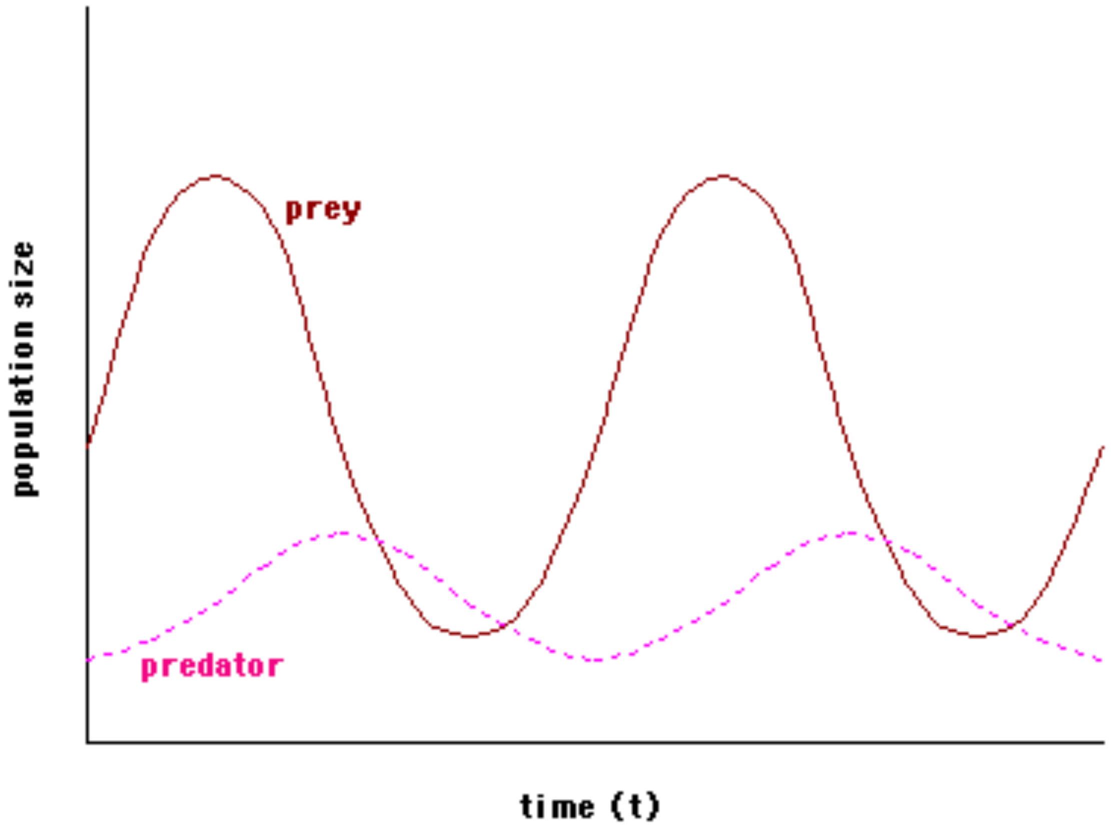


Other Application of CA: Consumer-resource interactions

- **Key:** One species exploits the other in some way

- Predator-prey
- Herbivore-plant
- Host-parasite/parasitoid
- Host-Pathogen

Predator-Prey (Lotka-Volterra)



Predator-Prey - Basics

- Applicable to other problems:
  - herbivore-plant
  - parasitoid-host

- oscillations in the population size of both predator and prey

- peak of the predator's oscillation slightly behind the peak of the prey's oscillation

Predator-Prey - Formulas

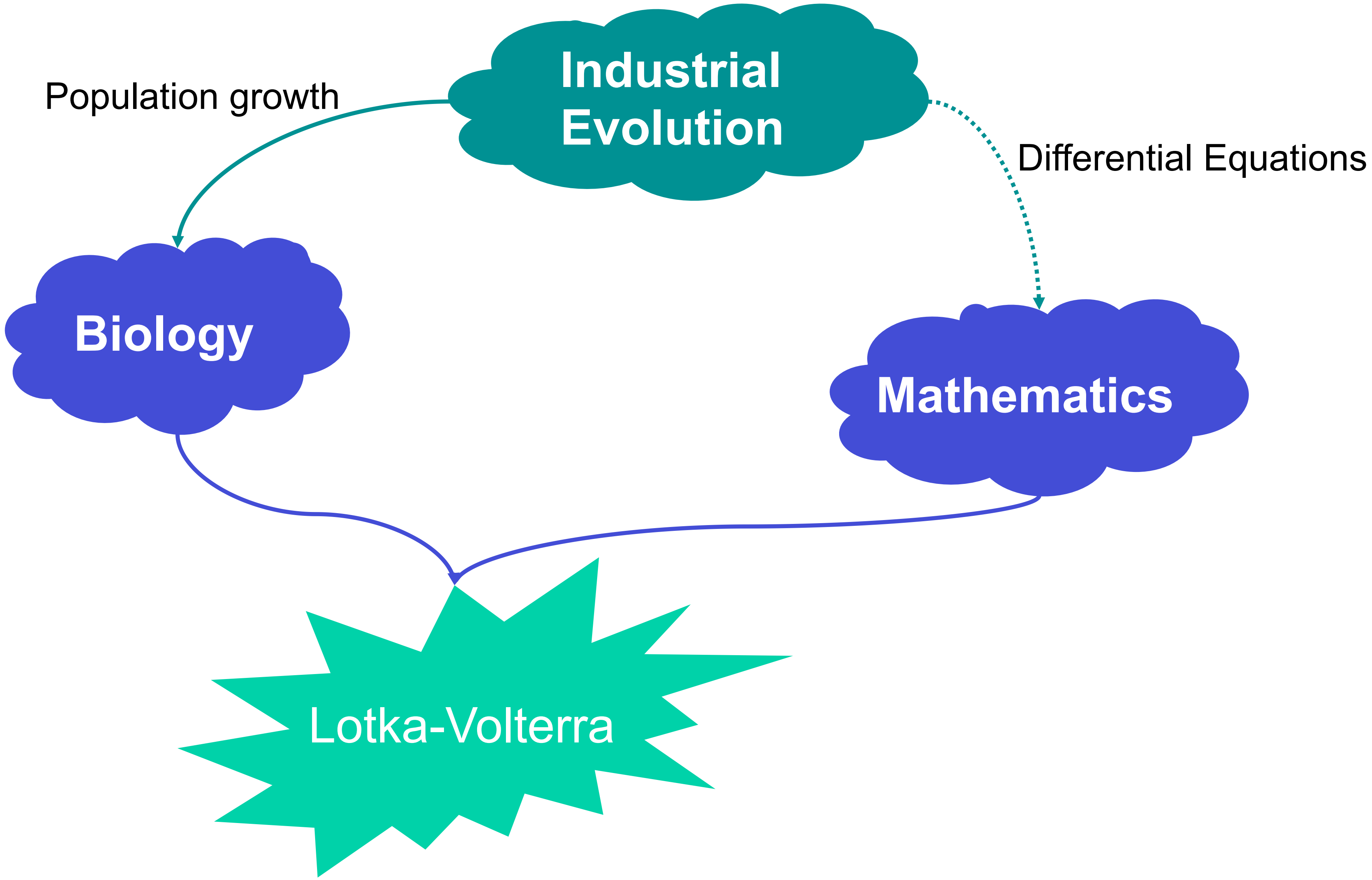
$$dP/dt = c_0P - d_0PR$$

$$dR/dt = c_1R + d_1PR$$

P = Prey

R = pRedator

Predator-Prey - History



Predator-Prey - Assumptions

- population will grow exponentially when the predator is absent
- predator population will starve in the absence of the prey population
- predators can consume infinite quantities of prey
- populations are moving randomly through a environment

Predator-Prey - History

- Alfred Lotka (american) and Vito Volterra (italian)
- Volterra wanted to model the fish populations in the Adriatic Sea
- Published his findings in Nature:

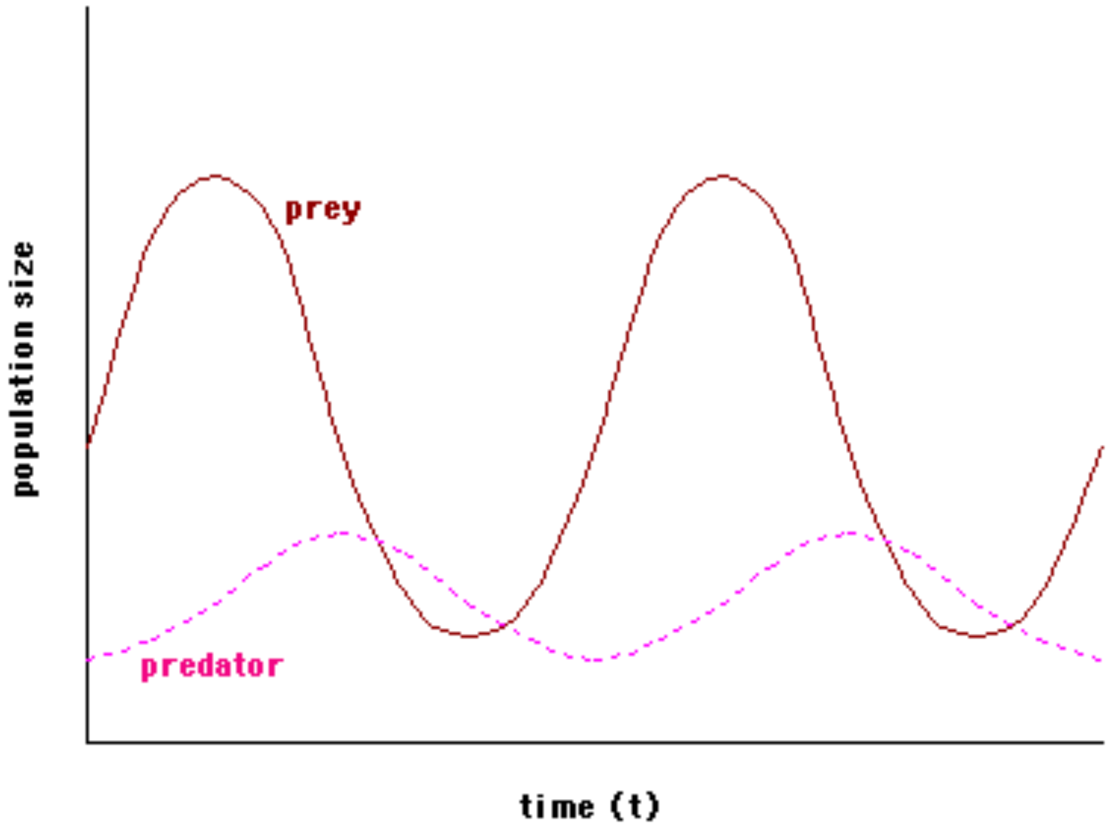
$$dP/dt = c_0P - d_0PR$$

$$dR/dt = c_1R + d_1PR$$

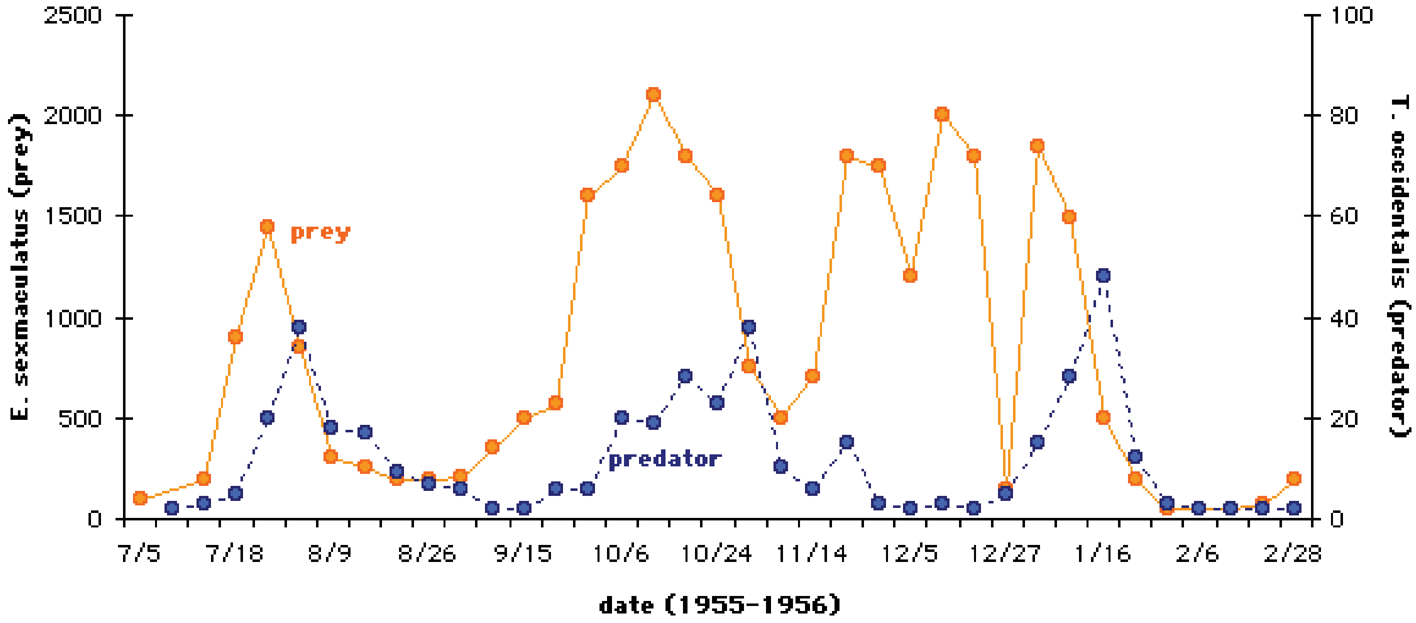
- Received letter from Lotka about previous similar publications of Lotka himself

- Developed model independently

Predator-Prey (ideal)

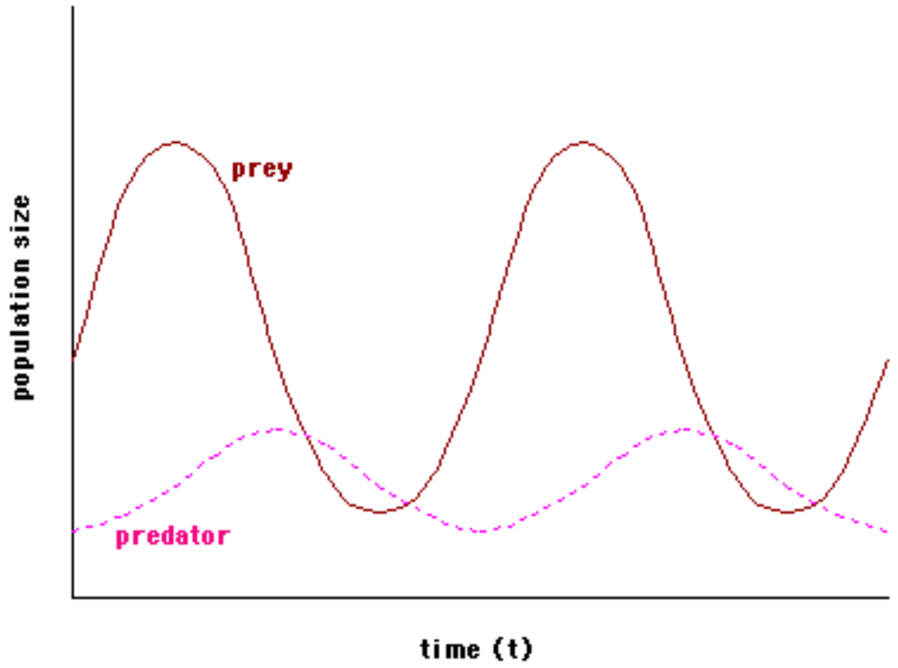


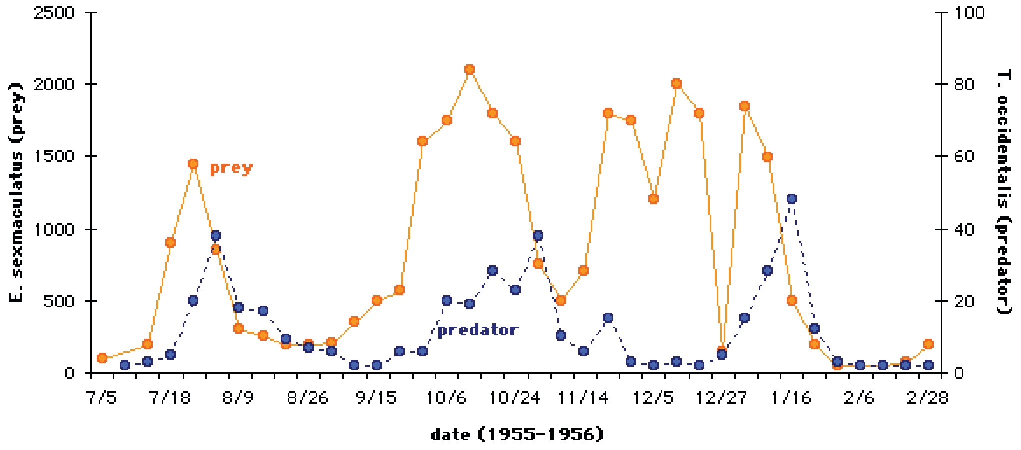
Predator-Prey (for real)



Huffaker (1958)

Predator-Prey - Correct?





On the side - About Models

- must be simple enough to be mathematically tractable, but complex enough to represent a system realistically.

- shortcomings of the Lotka-Volterra model is its reliance on unrealistic assumptions.
  - For example, prey populations are limited by food resources and not just by predation, and no predator can consume infinite quantities of prey.
  - Many other examples of cyclical relationships between predator and prey populations have been demonstrated in the laboratory or observed in nature, which are better explained by other models.

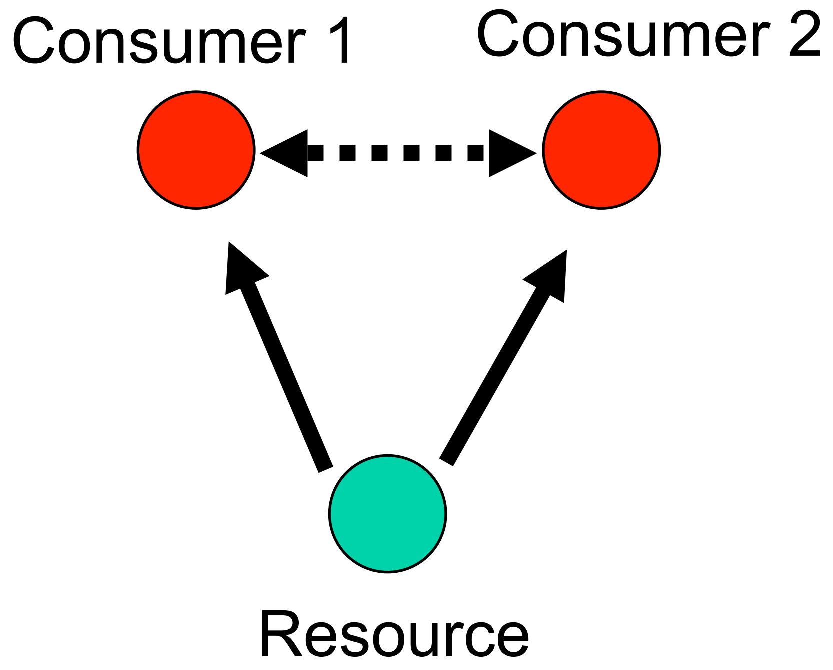
Characteristics of these types of problems

- Consumers can impact resources
- Consumers and resources are dynamic in space and time
- Models can result in complex dynamics

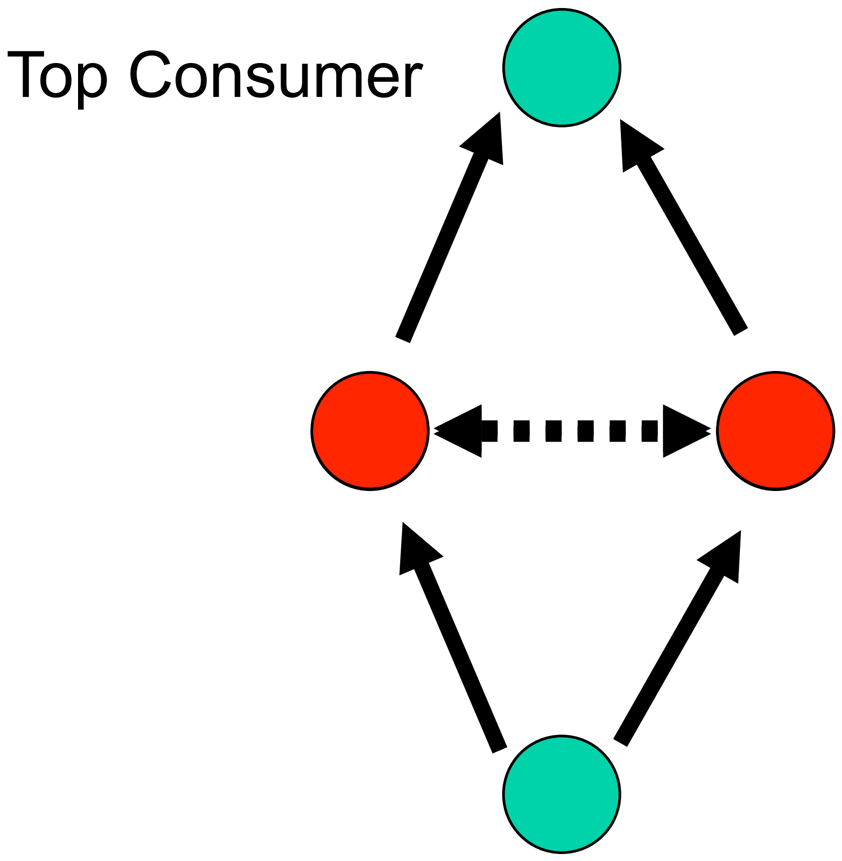
Complex web of interactions

Two consumers compete for resources

Competition **(true)**



Apparent Competition caused by consumer.



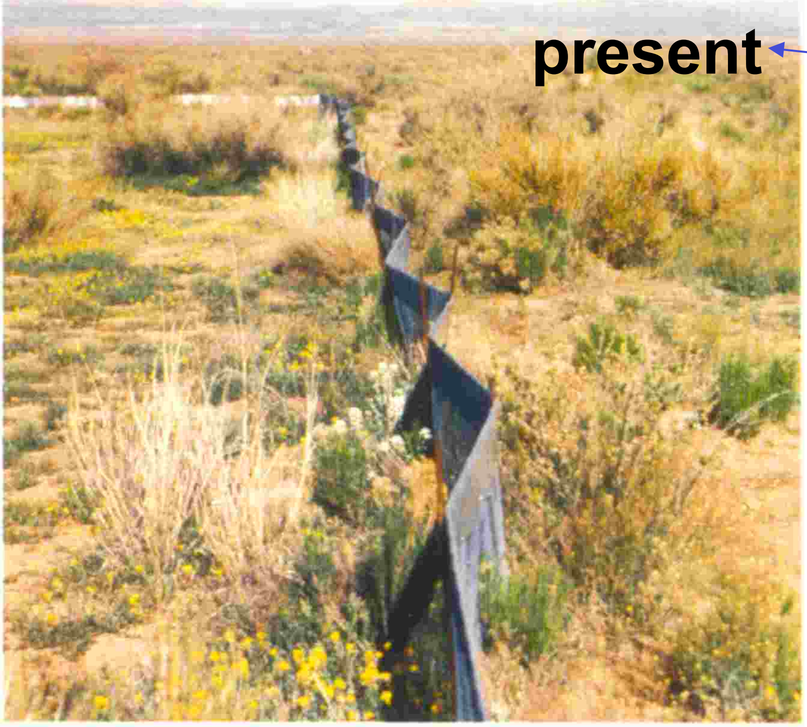
Predator-prey simulation

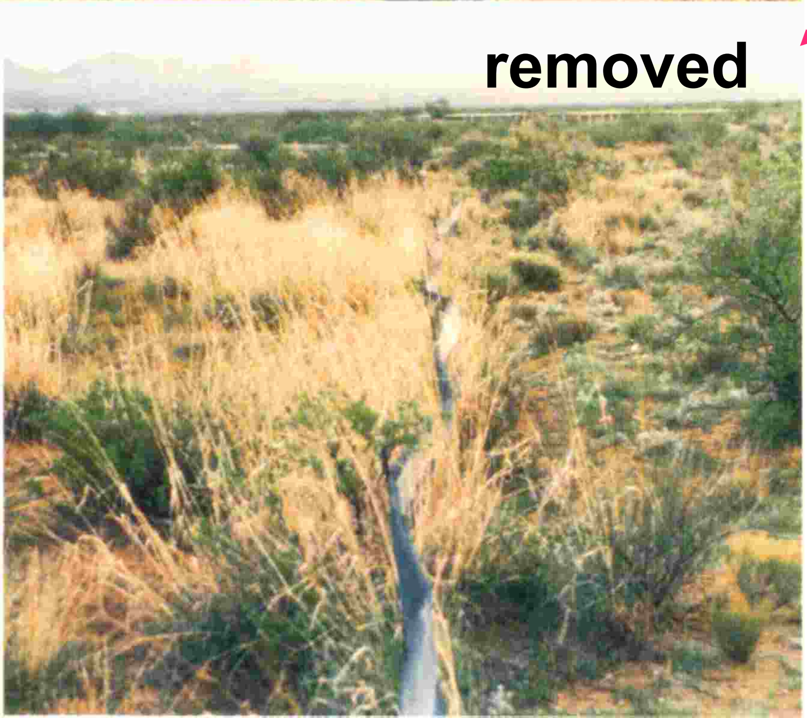
- Cellular Automata

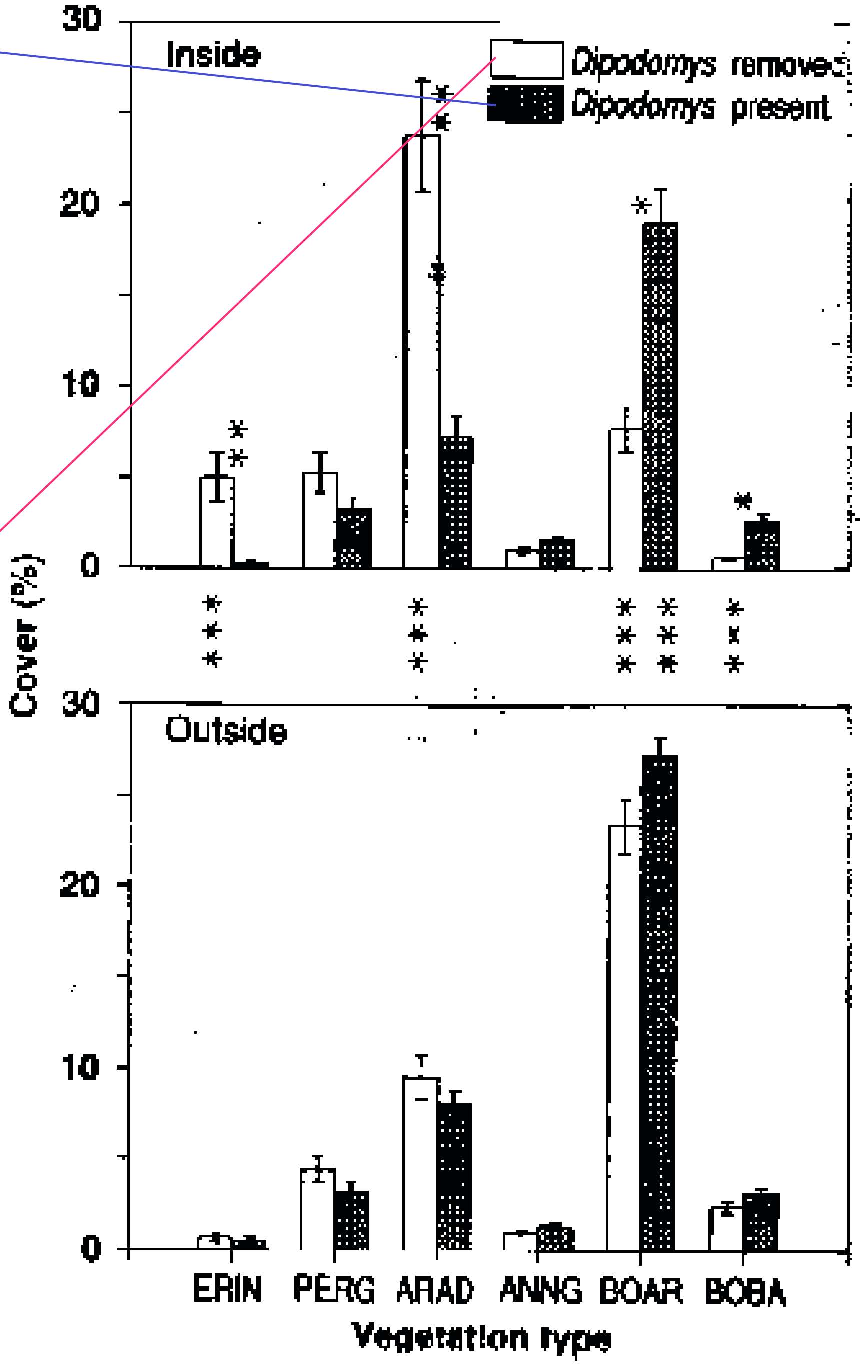
- http://www.stensland.net/java/erin.html

Examples: Impacts of predators

- Marine organisms.
  - Many lay 10,000 - 1,000,000 + eggs.
  - On average, all but one become an adult
- Desert/arid grassland transition.
  - Dominant vegetation type depends on small mammal seed predation.

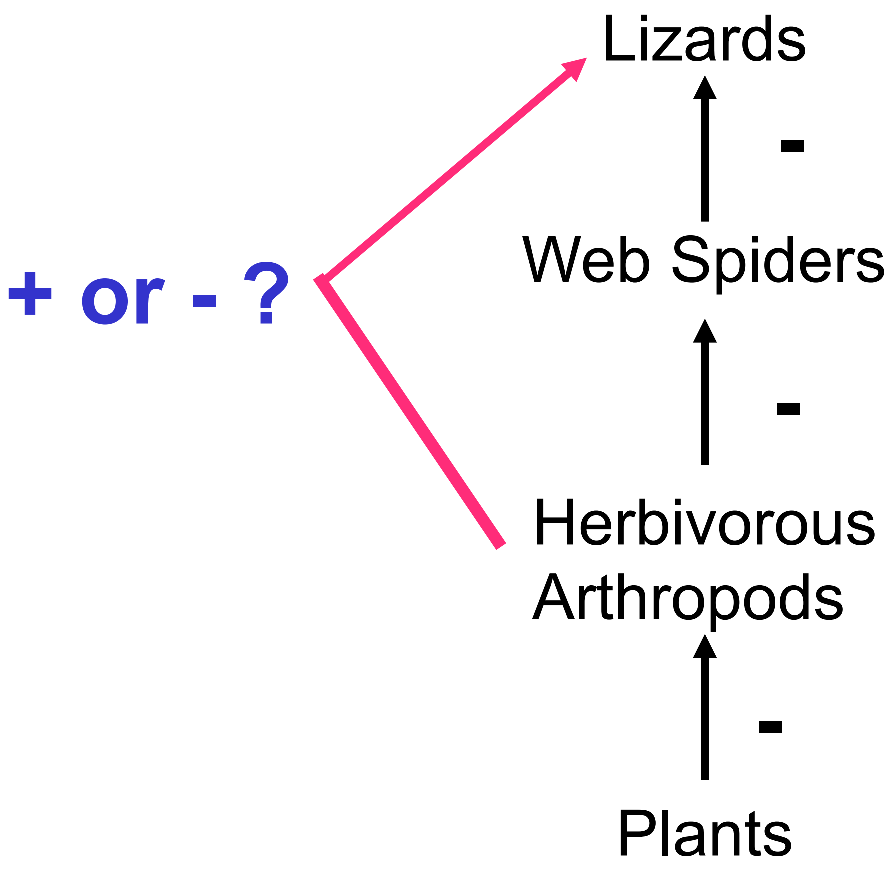

present


removed


Inside
Outside
Dipodomys removed
Dipodomys present
Cover (%)
30
20
10
0
ERIN
PERG
ARAD
ANNG
BOAR
BOBA
Vegetation type

Cascading Impacts of predators

Islands in the Bahamas. Simple food webs



Lizards eat spiders, spiders eat arthropods, etc.

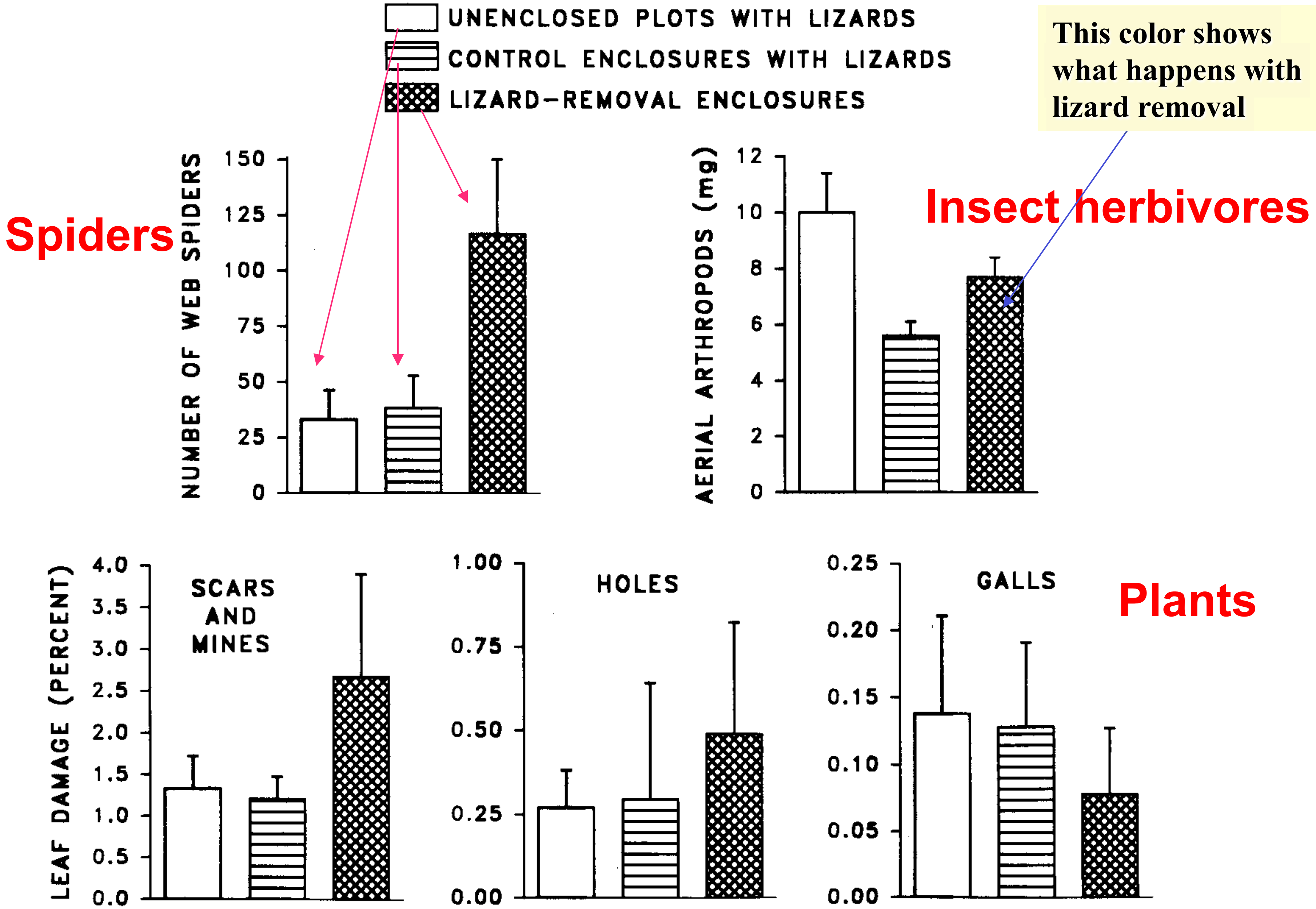
Less arthropods will cause less or more lizards?

Biomedical simulations: what can be done?

- Lizards feed at two trophic levels
  - They eat spiders (predators) and insect herbivores
  - Results in complex responses
- **Spiders increase by 3.1 times in absence of lizards**
  - predation and competition
- **Arthropods (control vs. removal).**
  - **Increase with lizards removed but not much…**
  - **Damage to plants depends on what predator is dominant**
  - **Lizards eat terrestrial bugs**
  - **Spiders eat flying insects**

Remove lizards, see what happens

This color shows what happens with lizard removal



Classification of Cellular Automata

- *dimension* 1D, 2D … nD
- *neighborhood* Neumann, Moore for 1D
  (2D => *r* is used to denote the radius)
- *number of states* 1,2,…, n

Cellular Automata - Types

- Symmetric CAs
- Spatial isotropic
- Legal
- Totalistic
- *Wolfram*

Cellular Automata - Steven Wolfram

I. Always reaches a state in which all cells are dead or alive

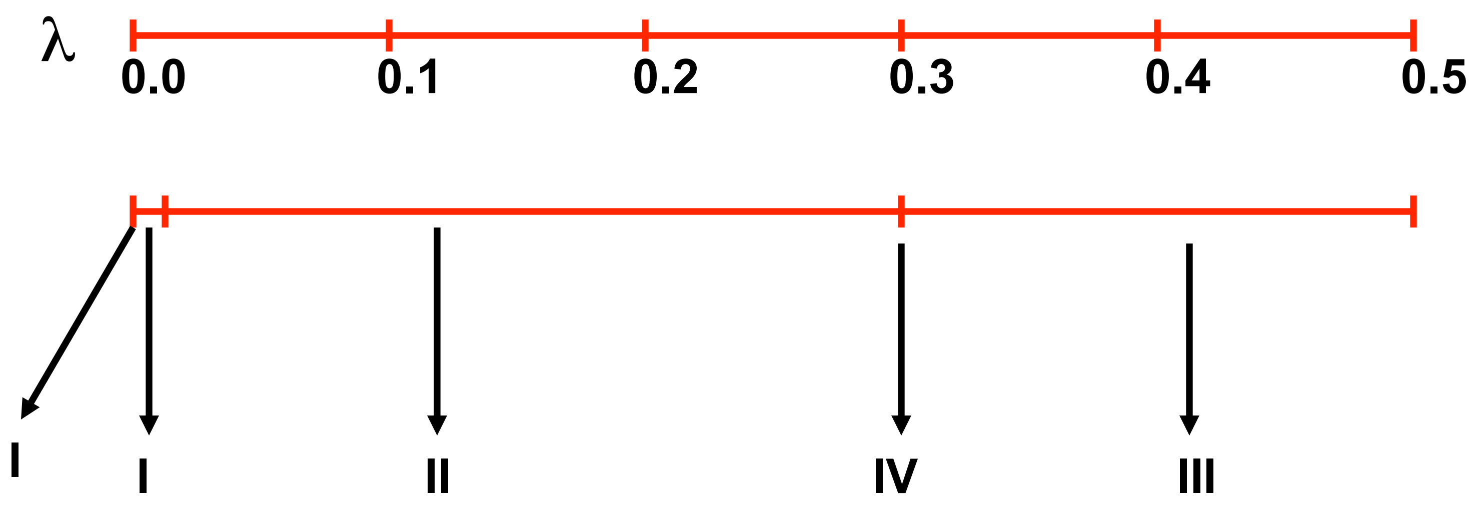
II. Periodic behavior

III. Everything occurs randomly

IV. Unstructured but complex behavior

Cellular Automata - Steven Wolfram

λ = chance that a cell is alive in the next state



What do these classes look like?

Cellular Automata - Complexity

- What is the total number of possibilities with CAs?
- Let's look at total number of possible rules
- For 1D CA:

  $2^3 = 8$ possible "neighborhoods" (for 3 cells)

  $2^8 = 256$ possible rules
- For 2D CA:

  $2^9 = 512$ possible "neighborhoods"

  2^{512} possible rules (!!)

Cellular Automata - Alive or not?

- Can CA or Game of Life represent life as we know it?

- A computer can be simulated in Life

- Building blocks of computer (wires, gates, registers) can be simulated in Life as patterns (gliders, eaters etcetera)

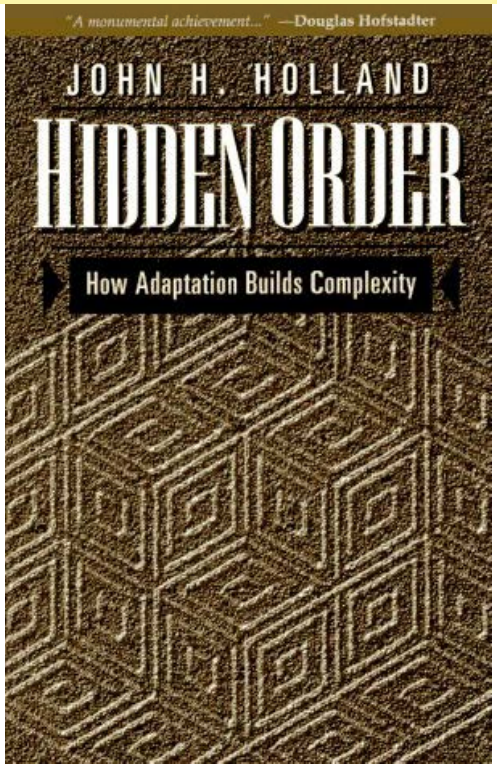
- Possible to build a computer, possible to build life?

Universal Machines - Cellular Automata

Stanislaw Ulam (1909 - 1984)

Universal Machines - Cellular Automata

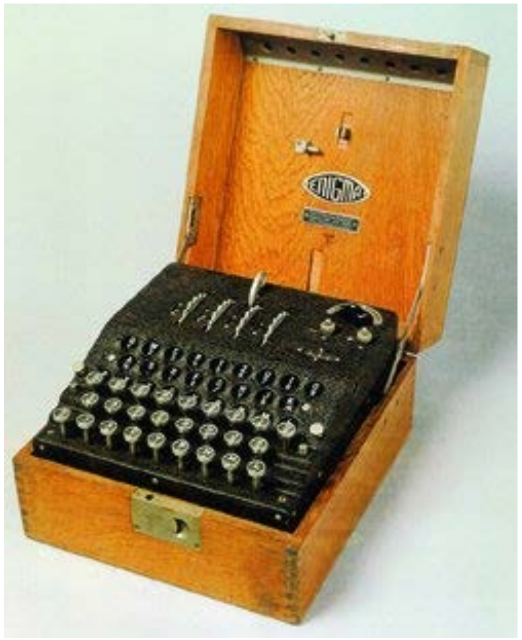
- conceived in the 1940s
- Stanislaw Ulam - evolution of graphic constructions generated by simple rules
- Ulam asked two questions:
  - can recursive mechanisms explain the complexity of the real?
  - Is complexity then only apparent, the rules themselves being simple?


"A monumental achievement..." —Douglas Hofstadter
JOHN H. HOLLAND
HIDDEN ORDER
How Adaptation Builds Complexity

ZOS Course book 1999/2000
Stanislaw Ulam Memorial Lectures

Universal Machines - Turing Machines

Alan Turing (1912-1954)



Universal Machines - Turing Machines

Data
(e.g., resignation letter)

Program
(e.g., Microsoft Word)

Are they really this different?

No, they're all just 0s and 1s!

Let's look at a Turing Machine!

Universal Machines - Neumann Machines

John von Neumann (1903- 1957)

Universal Machines - Neumann Machines

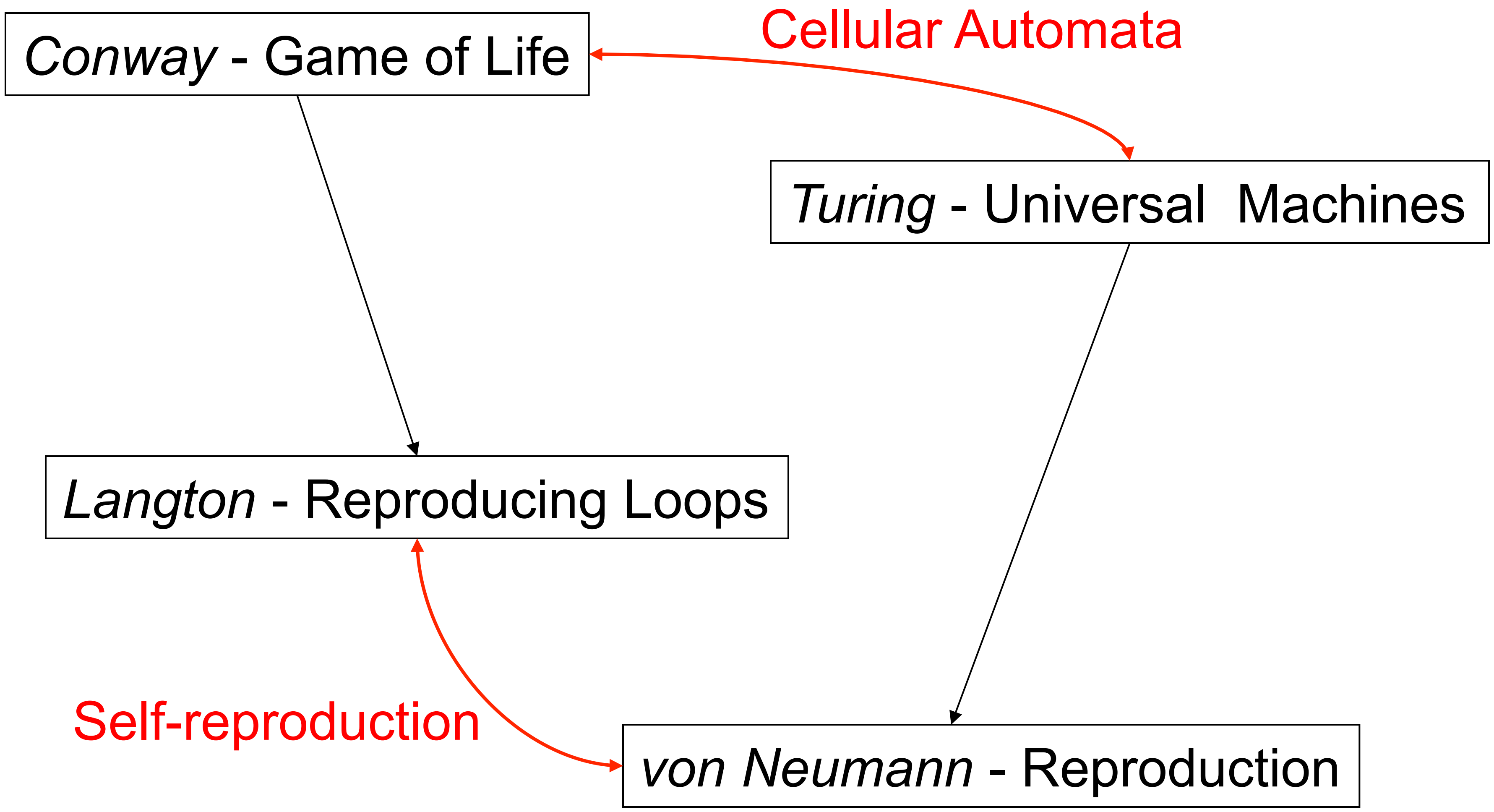
• John von Neumann interests himself on theory of self-reproductive automata

• worked on a self-reproducing “kinematon” (like the monolith in “2001 Space Odyssey”)

• Ulam suggested von Neumann to use “cellular spaces”

• extremely simplified universe

Self-Reproduction

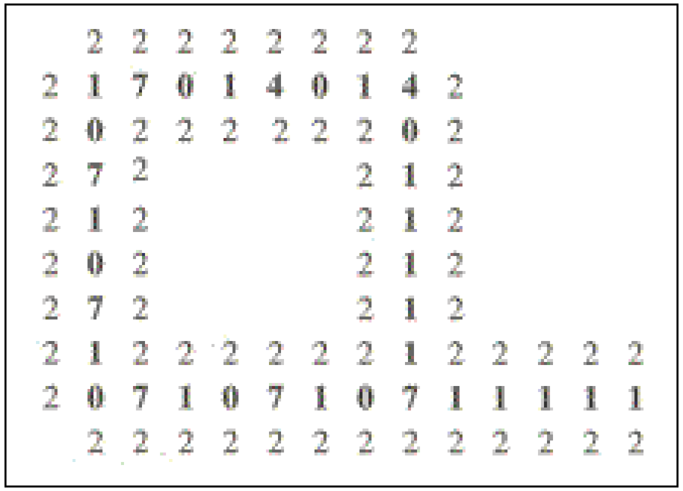


Cellular Automata
Conway - Game of Life
Turing - Universal Machines
Langton - Reproducing Loops
Self-reproduction
von Neumann - Reproduction

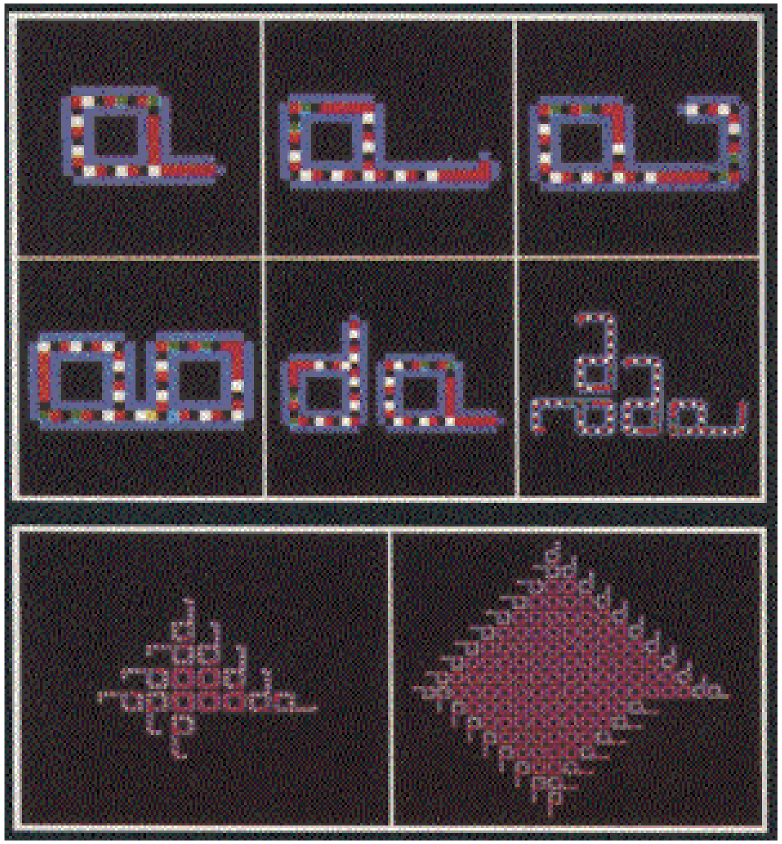
Self-Reproduction

An example of Self Reproducing Cellular Automata

Self-Reproduction



	2	2	2	2	2	2	2	2					
2	1	7	0	1	4	0	1	4	2				
2	0	2	2	2	2	2	2	0	2				
2	7	2					2	1	2				
2	1	2					2	1	2				
2	0	2					2	1	2				
2	7	2					2	1	2				
2	1	2	2	2	2	2	2	1	2	2	2	2	2
2	0	7	1	0	7	1	0	7	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2



Langton Loop’s

- 8 states
- 29 rules

Summary

- Ulam
- Turing
- von Neumann
- Conway
- Langton

- Universal Machines
- Turing Machines
- von Neumann Machines
- Game of Life
- Self Reproduction

Sources

1. Seth Copen Goldstein, CMU
2. David E. Culler, UC. Berkeley,
3. Keller@cs.hmc.edu
4. Syeda Mohsina Afroze
and other students of Advanced Logic Synthesis, ECE 572, 1999 and 2000.
5. Russell Deaton, The University of Memphis
6. Nouraddin Alhagi, class 572
7. Schut, Vrije Universiteit, Amsterdam

Computers, Life and Complexity

http://ivytech7.cc.in.us/mathsci/complexity/life/index.htm

Allen Shotwell
Ivy Tech State College

Overview

- Start Specific - Game of Life
- Globalize - Cellular Automata
- Speculate - Analogies and Stuff

Game of Life Overview

- History
- The Rules
- "Manual" Life
- Computerized Life
- Emergent Behavior
- Parameter Effects

FOR MORE INFO...

http://www.reed.edu/~jwalton/gameoflife.html

Game of Life - History

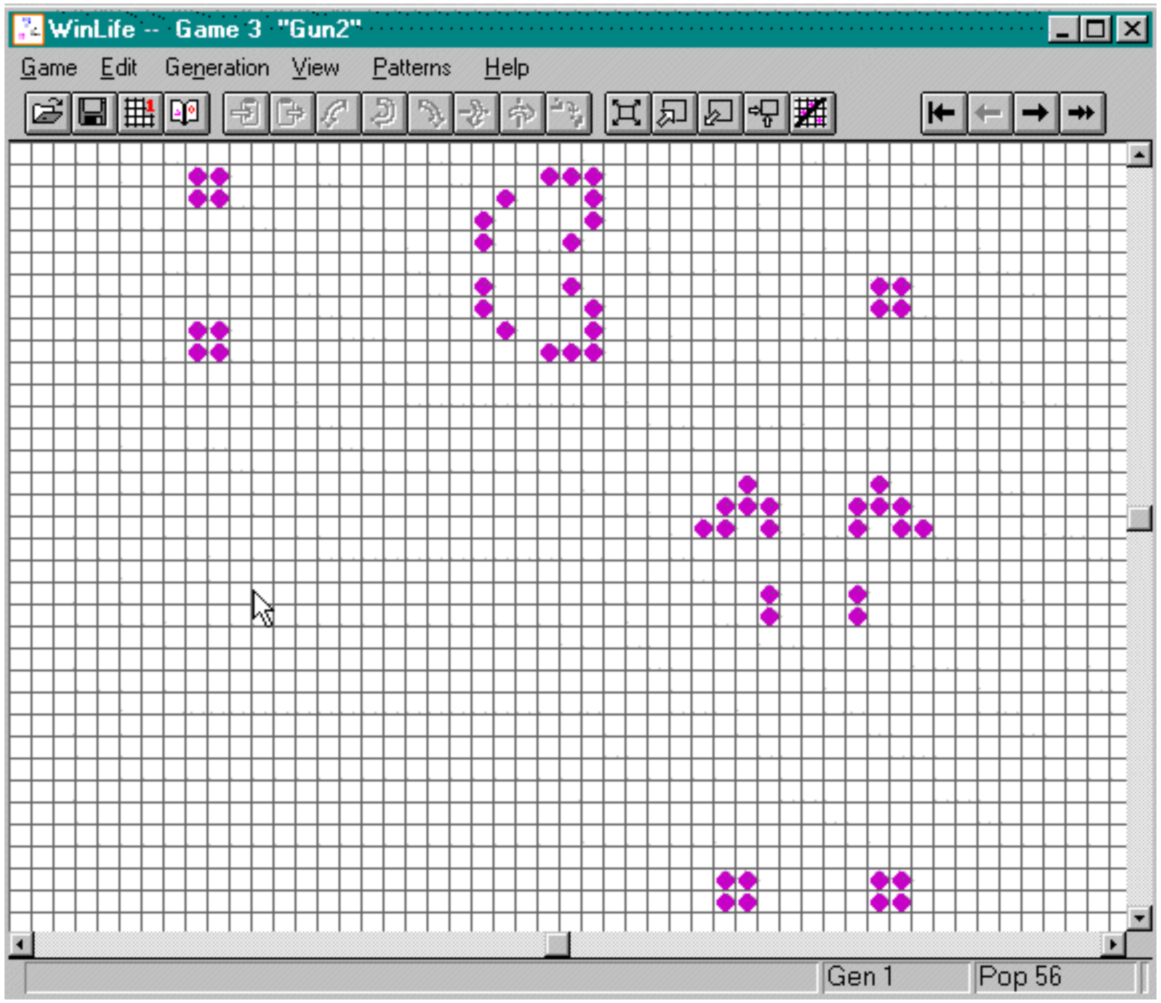
- Ulam and Von Neumann
- John Horton Conway
- April 1970, Scientific American, Martin Gardner
- Past time for computer programmers and mathematicians

The Rules

Each cell has eight possible neighbors (four on its sides, four on its corners).

The rules for survival, death and birth are as follows:

- survival: if a live cell has two or three live neighbors, it survives.
- death: if a live cell has less than two or more than three live neighbors, it dies.
- birth: if a dead cell has exactly three live neighbors, it is born.


WinLife -- Game 3 "Gun2"
Game
Edit
Generation
View
Patterns
Help
Gen 1
Pop 56

“Manual” Life

- Try it

Computerized Life

- Java - http://remus.rutgers.edu/~kenn/java/Life/
- DOS - ftp://ftp.cs.jhu.edu/pub/callahan/conways_life/life16.zip
- X Windows - ftp://ftp.cs.jhu.edu/pub/callahan/conways_life/xlife-3.0.tar.gz
- Windows 95 - http://www.mindspring.com/~alanh/Life32/
- Windows 3.1 - ftp://ftp.cs.jhu.edu/pub/callahan/conways_life/wlife.zip

Emergent behavior

A. Simple Life Forms

B. The Glider

C. The Eater

D. Methuselahs

E. Glider Guns and Puffer Trains

F. Spaceships

FOR MORE INFO...

http://www.reed.edu/~jwalton/gameoflife.html

A. Simple Life Forms

- Simple Life forms can generally be grouped into two categories: still-lifes and blinkers.
- Still-lifes are stable forms that do not change over successive generations unless disturbed by other live cells.

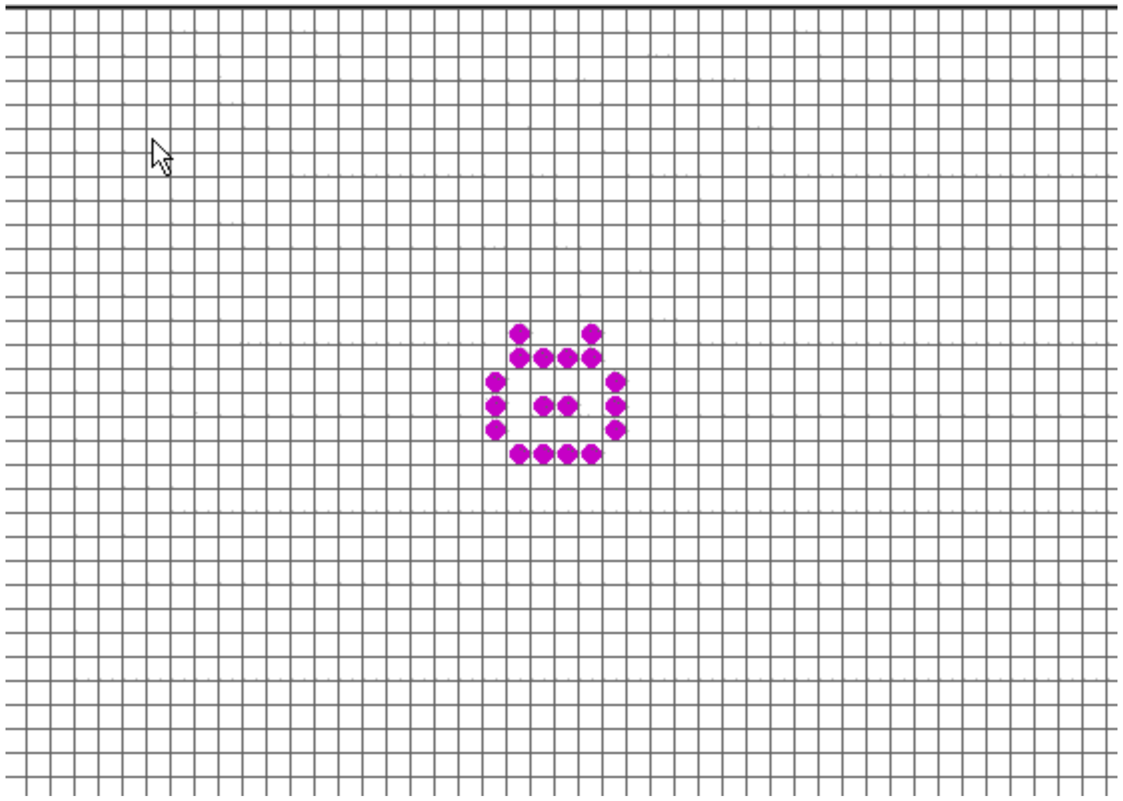
- The block and the beehive are two common forms of still-lifes.
- Blinkers are periodic Life forms which have predictable behavior.
- The traffic light is a quite common form of blinker that has a period of two.

B. The Glider

- The glider is a unique Life form.
- Technically, it is a blinker, having a period of four.
- However, it is not a stationary blinker, but one that travels a single diagonal cell ever four generations.
- If a glider is capable of escaping the milieu of a changing pattern, it travels off into the distance forever.

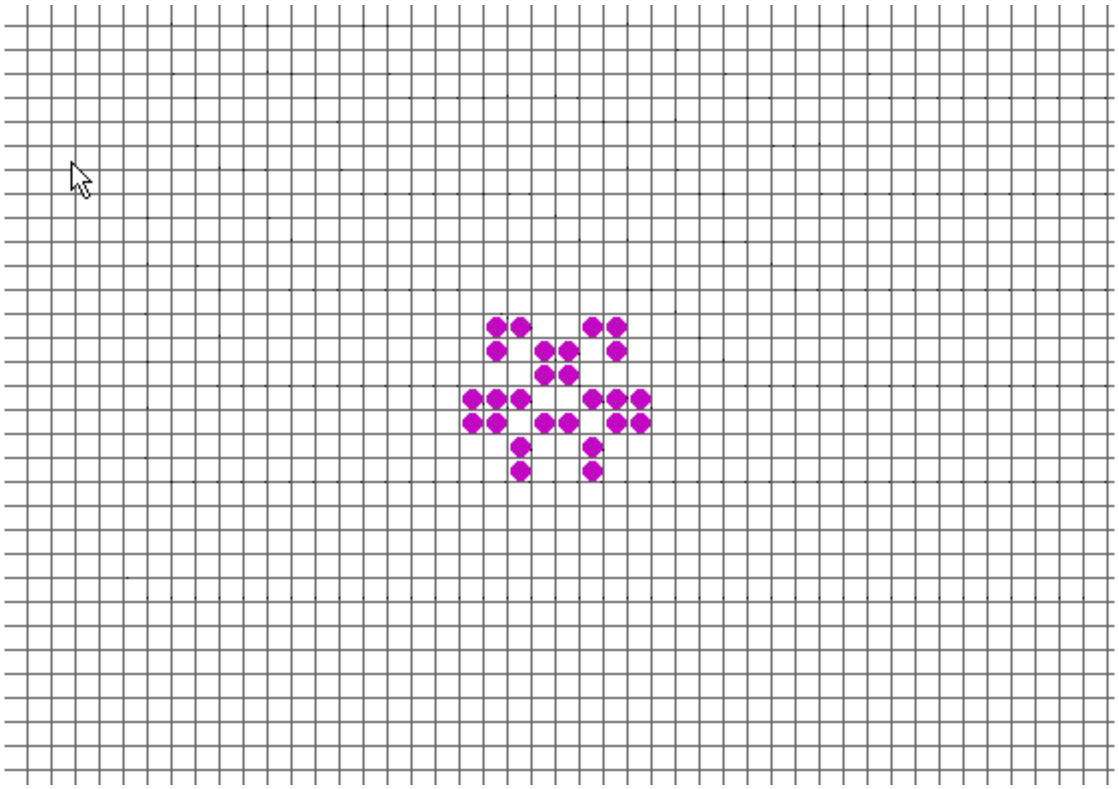
Cheshire:

Generation 1:



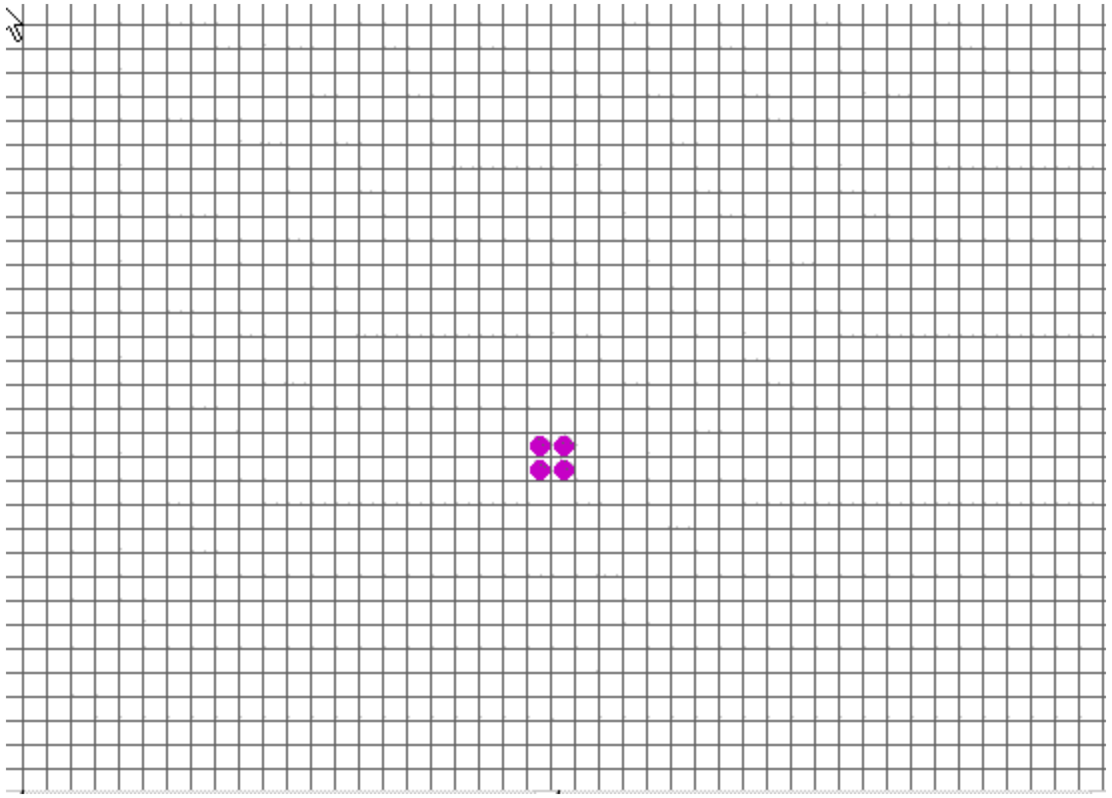
Cheshire:

Generation 3



Cheshire:

Generation 8 and on

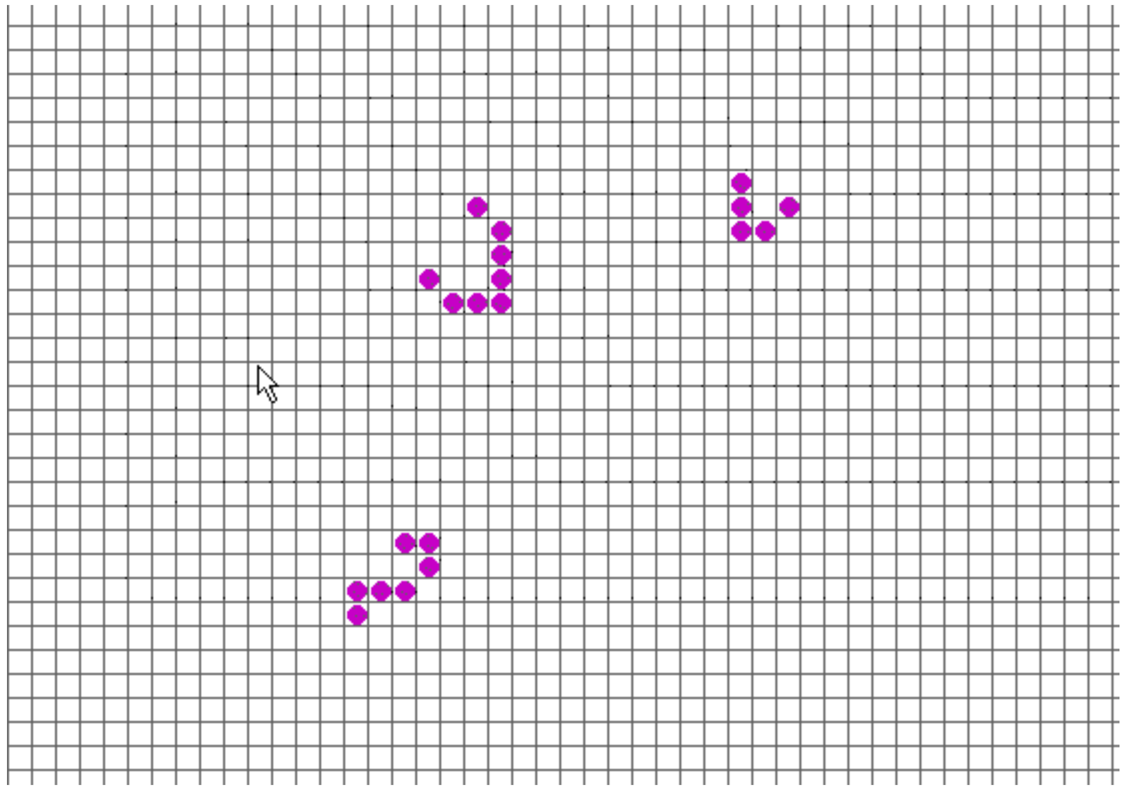


C. The Eater

- The eater is a still-life that is remarkable for it's capacity to devour gliders, provided that the glider approaches it at a certain angle.

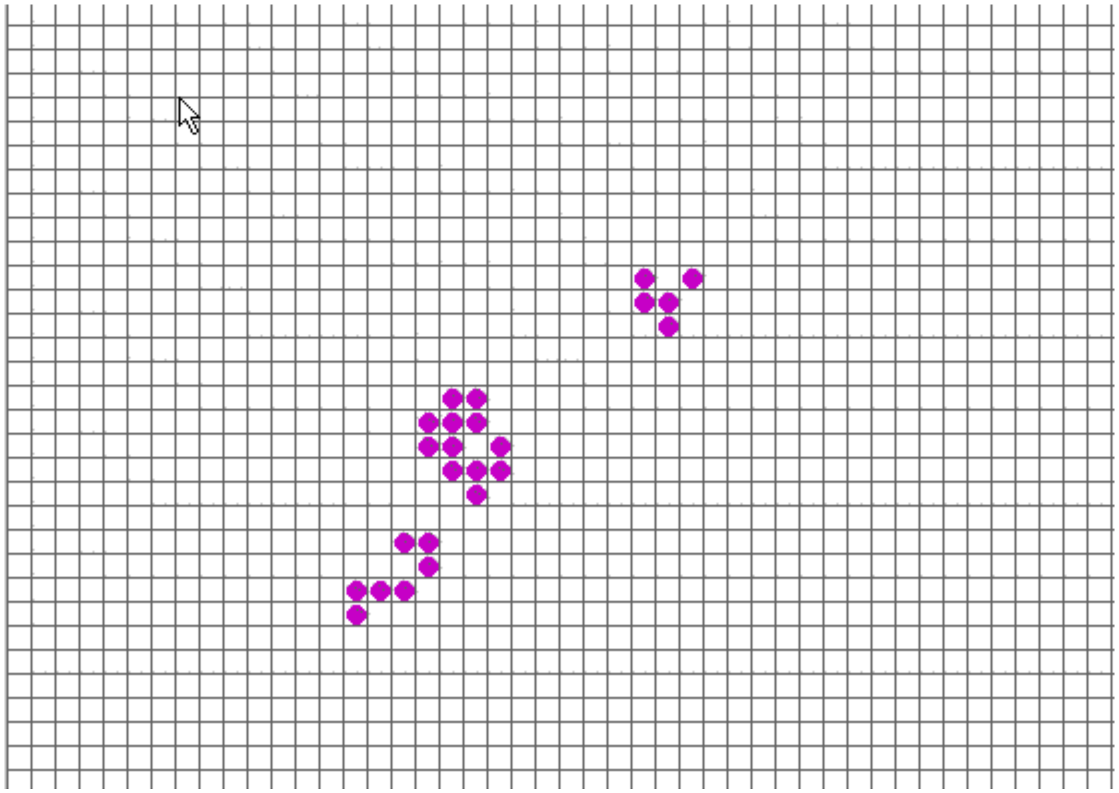
Eater:

Generation 1



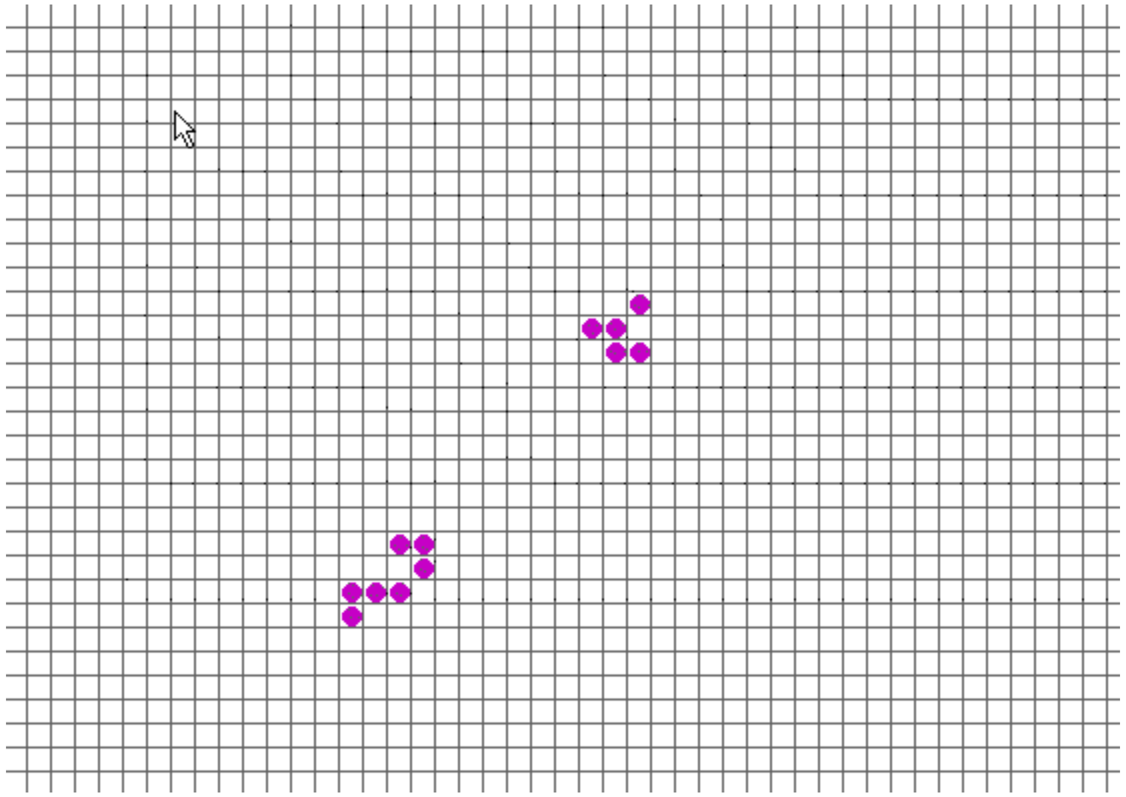
Eater:

Generation 16



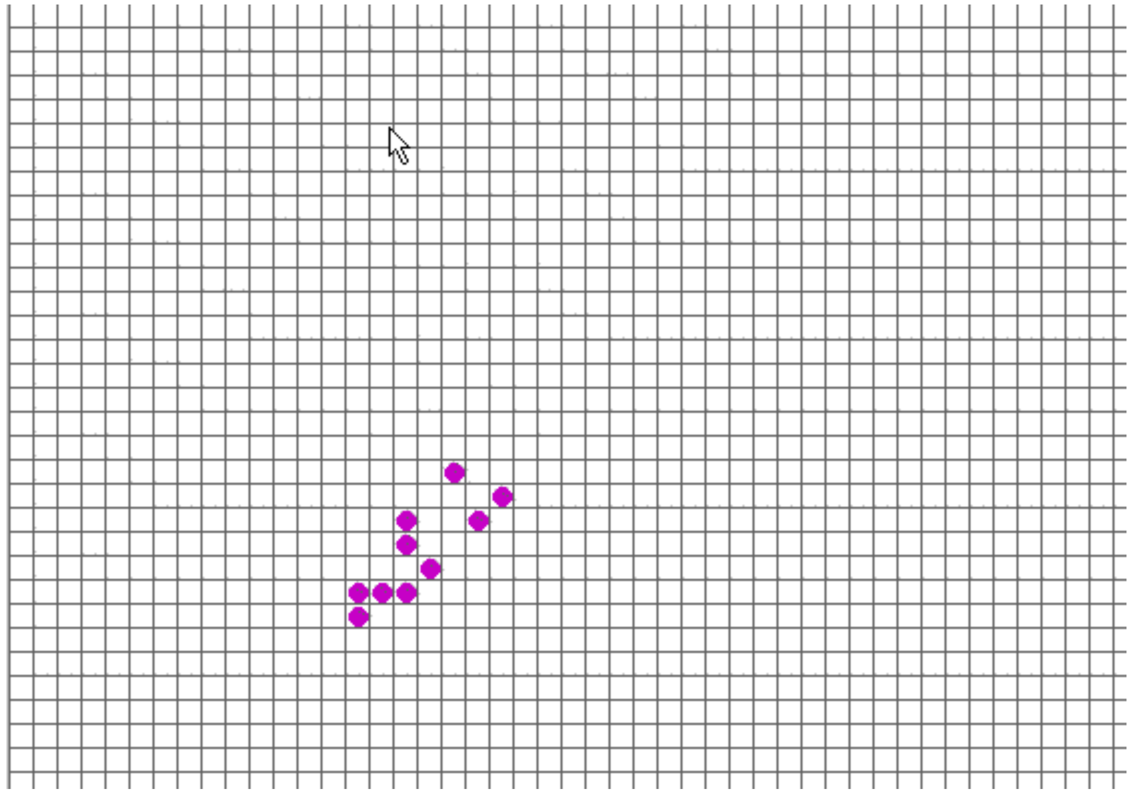
Eater:

Generation 22



Eater:

Generation 48

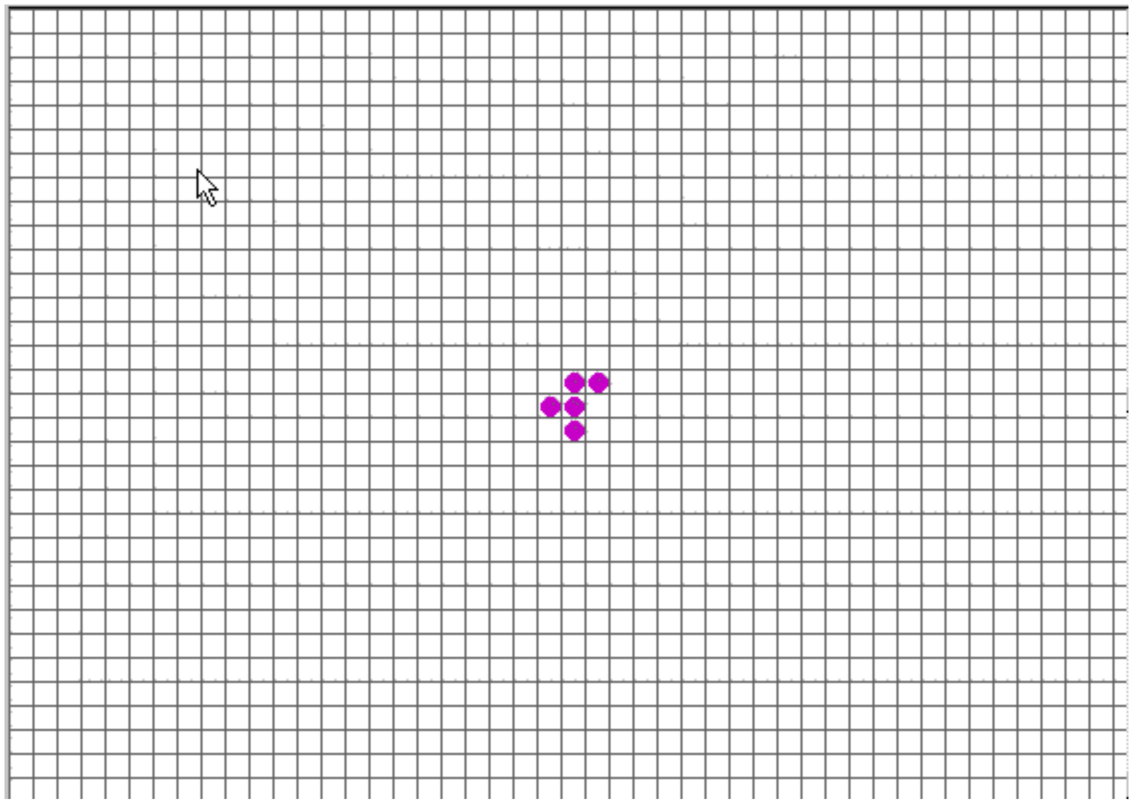


D. Methuselahs

- Methuselahs are initial patterns of live cells that require a large number of generations to stabilize.
- The smallest and perhaps most famous methuselah is the R-pentomino which takes 1103 generations before it reaches a stable (periodic) state.
- The acorn is perhaps the longest lived methuselah known, requiring over 5000 generations to reach periodicity.

R-pentomino

Generation 1:

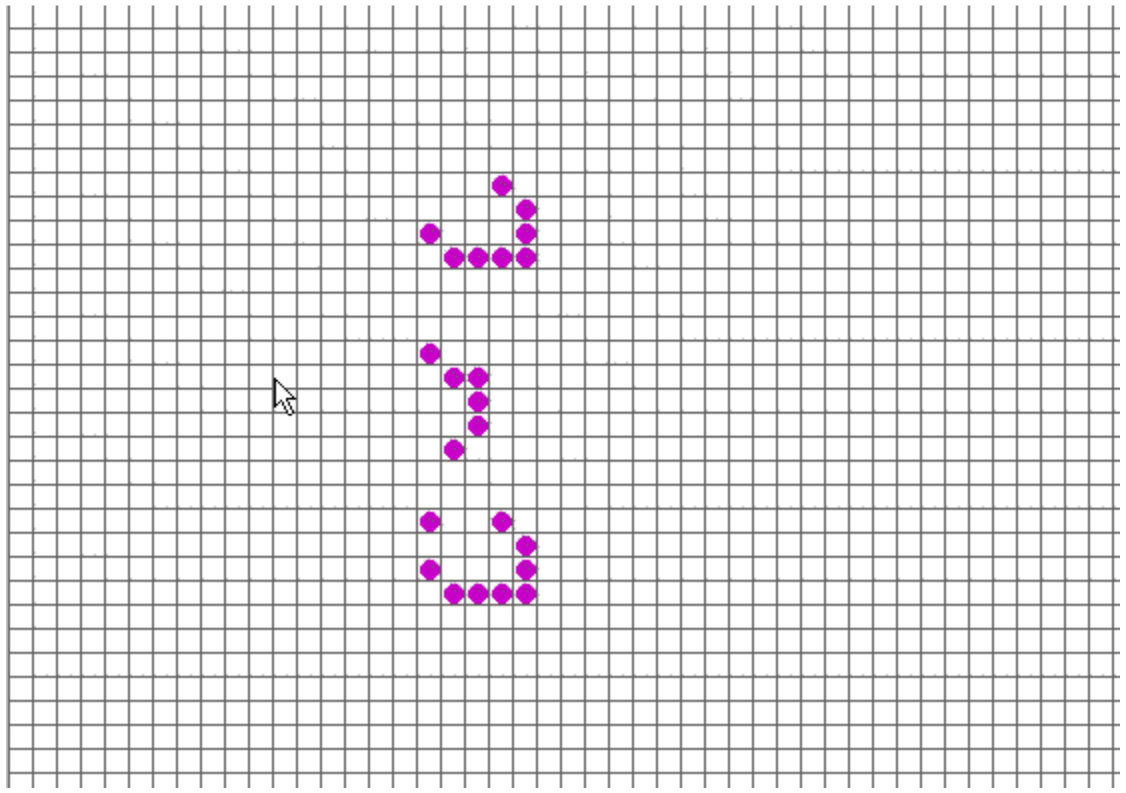


E. Glider Guns and Puffer Trains

- Glider guns and puffer trains are two unique Life forms that produce populations of live cells that grow without limit.
- The glider gun simply continues to produce gliders
- The puffer train travels endlessly in a horizontal direction, leaving a trail of smoke behind it.
- Both were discovered by Bill Gosper of the Massachusetts Institute of Technology.

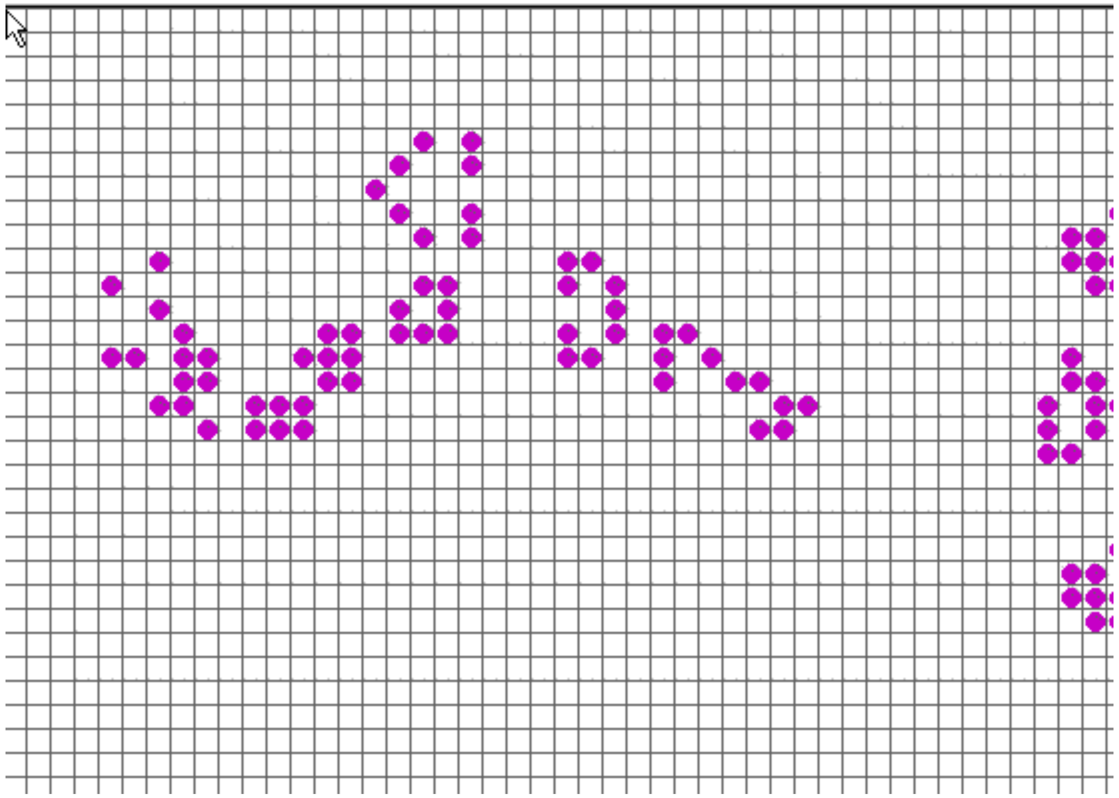
Puffer Train:

Generation 1



Puffer Train:

Generation 54



F. Spaceships

- A spaceship is a finite pattern of live cells in Life that after a certain number of generations reappears, but translated in some direction by a nonzero distance.

- Translation is measured by the shortest king-wise connected path between two cells.

- So two cells adjacent to each other are one cell apart, whether or not the cells are adjacent orthogonally or diagonally.

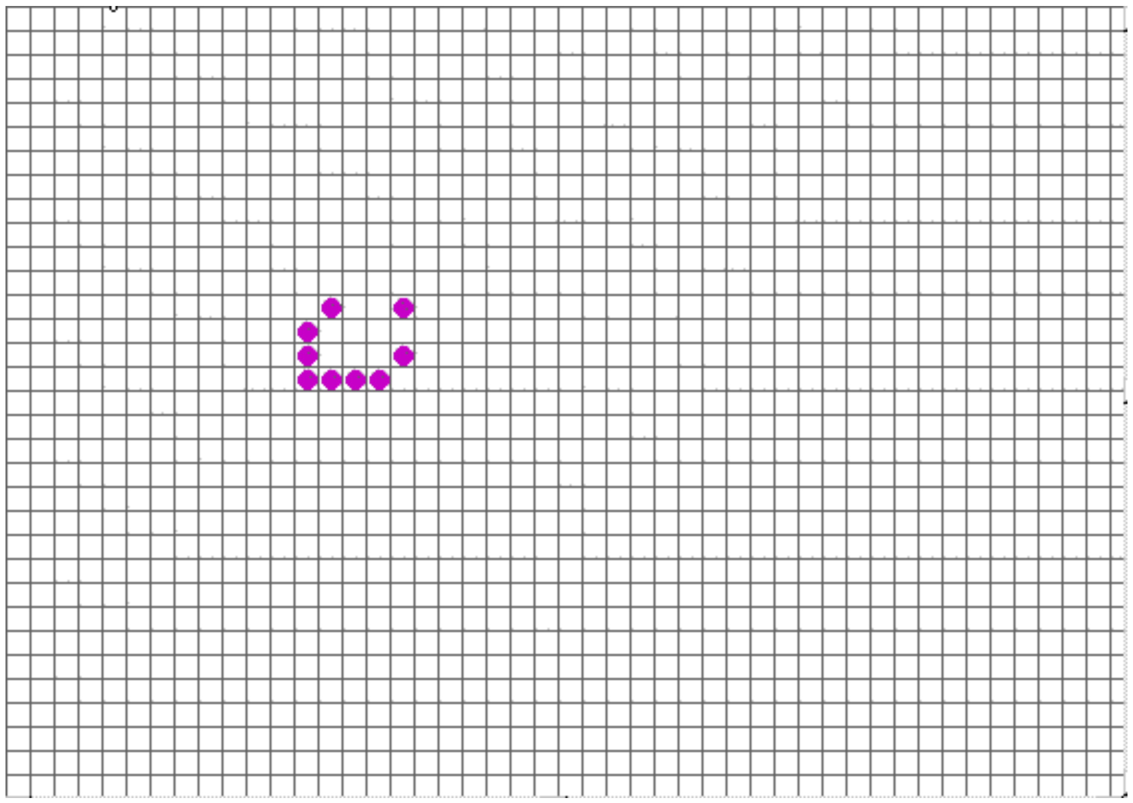
- The process of translating after a certain number of generations is called moving or traveling.
- The number of generations before the pattern reappears (but translated) is called the period of the spaceship.
  - (This is analogous to the period of stationary oscillators.)

F. Spaceships (Cont)

- Many spaceships appear to be the same pattern after a number of generations,
  - but on closer examination are not really the same.
- Instead of being merely translated, they are also reflected (or flipped) as in a mirror.
- After twice that number of generations, the spaceship does reappear in its original form, with just a translation.
- The period always measures this full number of generations.
- Spaceships which show their mirror image after half their period are called glide- reflection spaceships.

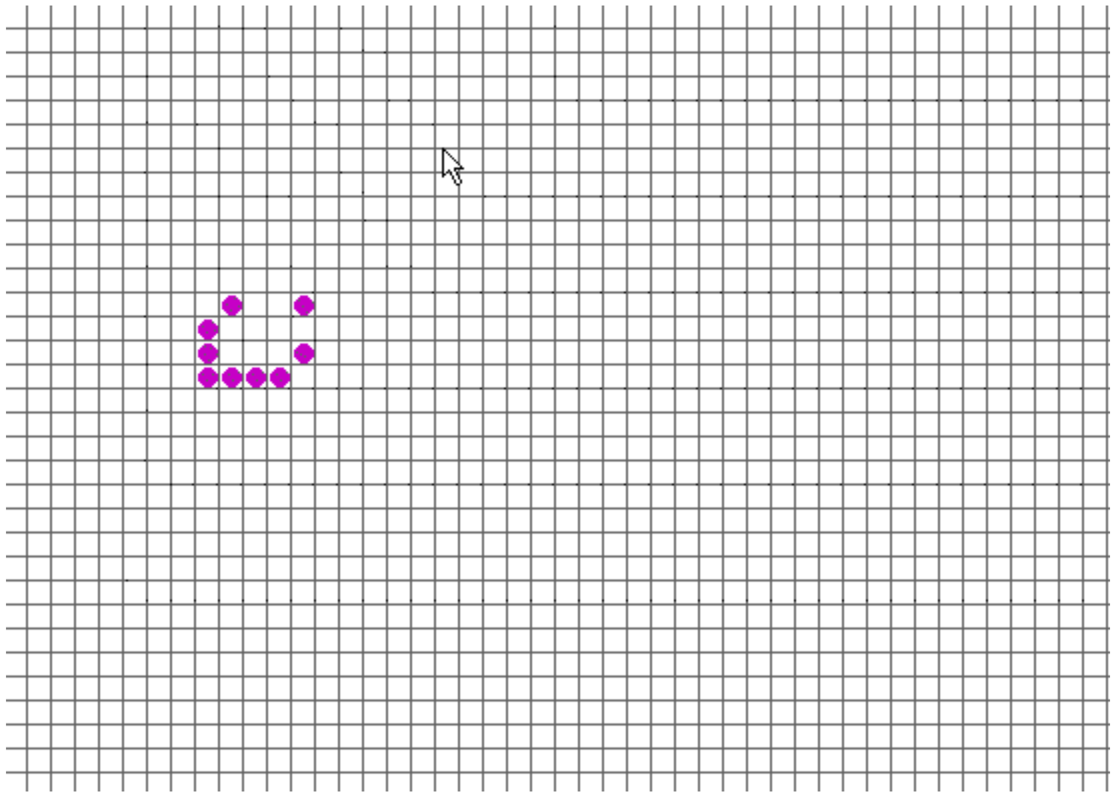
Example Space Ship:

Generation 1



Example Spaceship:

Generation 5



Additional Life/CA Forms

There are many other variations on these themes and many methods (in some cases) of ways of producing the ones shown here.

•ANTS (http://math.math.sunysb.edu/%7Escott/ants/)

•High Life (http://www.cs.jhu.edu/~callahan/altrule.html)

•Day and Night (http://www.cs.jhu.edu/~callahan/altrule.html)

•L-systems

FOR MORE INFO...

http://www.cs.jhu.edu/~callahan/lexiconf.htm

Cellular Automata

- Extension of “Life”
- Different Rules
- Varying Dimensions

FOR MORE INFO...

http://alife.santafe.edu/alife/topics/cas/ca-faq/ca-faq.html

Cellular Automata and Emergence

- Mandelbrot' s Fictitious Example
  - Sierpinski gasket arising from a grid of spins. N=-1 if S(t-1, n-1) = S(t-1,n+1) else n=+1
- 1D CA and Attractor Types

Analysis of Emergence in CA

- Wolfram's Classification
- Mean Field Theory
- λ Parameter
- Holland's CGP

FOR MORE INFO...

http://www.santafe.edu/~hag/complex1/complex1.html

Wolfram’s Classifications

Four Classifications

- Class I Very Dull
- Class II Dull
- Class III Interesting
- Class IV Very Interesting

FOR MORE INFO...

http://alife.santafe.edu/alife/topics/cas/ca-faq/classify/classify.html

S. Wolfram, Statistical Mechanics of CA, Rev. Mod. Phys. 55:601-644 (1983)

Class I Very Dull

- All Configurations map to a homogeneous state

Class II Dull

- All configurations map to simple, separated periodic structures

Class III Interesting

- Produces chaotic patterns (impossible to predict long time behavior)

Class IV Very Interesting

- Produces propagating structures, may be used in computations (The Game of Life)

Mean Field Theory

- Statistical Picture of performance
- Based on
  - The probability of a cell being in a certain state
  - The probability of a block (certain specified states in specified locations) at a given time.
  - No correlation between cell probabilities
- Approximation which improves in large time or dimensional limits

λ Parameter

- Measure of the distribution of state transitions (alive to dead/dead to alive)
- Fraction of the rule table containing neighborhoods with nonzero transitions.
- Low λ, low Wolfram class
- High λ, higher Wolfram class
- Works bests in the limit of infinite number of states.

Holland's CGP

- Constrained Generating Procedures - models
- Rules that are simple can generate emergent behavior
- Emergence centers on interactions that are more than the summing of independent activities
- Persistent emergent phenomena can serve as components of more complex emergent phenomena

FOR MORE INFO...

Holland, Emergence, Helix Books (Addison Wesley), 1998

CA as Turing Machines

- Glider Guns as Logic Gates
- Universal Turing Machines

FOR MORE INFO...

http://cgi.student.nada.kth.se/cgi-bin/d95-aeh/get/umeng

Life and Music Generation

- Brian Eno
- http://www.cs.jhu.edu/~callahan/enoexcerpt.html

Life and Universal Computing

- BIT REPRESENTATION
- NOT Gate
- AND GATE
- NAND GATE
- UNIVERSAL GATE
- BASIC DIGITAL CIRCUITS

BIT REPRESENTATION

- The Glider can be used to represent a ‘1’
- The absence of a Glider can be used to represent a ‘0’
- A string of Gliders with equal spacing is a binary number.

The NOT Gate

- The Collision of two gliders at the proper angle eliminates them.
- The Gun can be used as a source of continuous, equally spaced gliders
- A stream of gliders representing bits can be placed so that it intersects with the Gun's output to produce a NOT Gate

The AND Gate

- The Output of a Note Gate whose in input is A (NOT A) is combined with a second stream, B
- The Collisions of B and Not A are the same as B AND A

THE NAND Gate

- The output of B AND A shot through another gun stream produces B NAND A

BASIC DIGITAL CIRCUITS

- The Adder

Artificial Life

Artificial Life ("AL" or "Alife") is the name given to a new discipline that studies "natural" life by attempting to recreate biological phenomena from scratch within computers and other "artificial" media.

Alife complements the traditional analytic approach of traditional biology with a synthetic approach in which, rather than studying biological phenomena by taking apart living organisms to see how they work, one attempts to put together systems that behave like living organisms.

Simulators

The phrase Artificial Life is somewhat all-encompassing including things such as:

artificial intelligence,

simulation of natural life,

etc.

To narrow the discussion some, we will focus on Artificial Life Simulations.

Fundamental Algorithms of ALife Simulation

Artificial life simulation can be subdivided into categories based on the algorithm used.

The subdivisions are

- Neural Networks
- Evolutionary Algorithms
- Cellular Automata

Neural Networks

Input-output “neurons” organized into highly connected networks.

Used for higher-order processes such as learning.

(Brain Model)

Evolutionary Algorithms

Iterative algorithms that contain a population of individuals that compete. Each iteration produces survivors who compete the best.

There are several methods.

- Genetic Algorithm
- Genetic Programming
- Evolutionary Programming
- Classifier Systems
- Lindenmeyer Systems

Genetic Algorithm

- By J. Holland.
- A population of individuals is chosen at random.
- The “fitness” of the individuals is determined through a defined function.
- Fit individuals are kept and unfit ones are not.
- The new population undergoes the process.
- Individuals are in practice arrays of bits or characters.

Genetic Programming

Similar to the GA except the individuals are computer programs in a lisp environment.

Evolutionary Programming

Start with a random population.

Mutate the individuals to produce the new population.

Assess fitness of offspring (new population)

Classifier Systems

J. Holland. Early application of Gas.

CFSs use evolutionary algorithm to adapt their behavior toward a changing ENVIRONMENT.

Holland envisioned a cognitive system capable of classifying the goings on in its ENVIRONMENT, and then reacting to these goings on appropriately.

So what is needed to build such a system? Obviously, we need

(1) an environment;

(2) receptors that tell our system about the goings on;

(3) effectors, that let our system manipulate its environment; and

(4) the system itself, conveniently a "black box" in this first approach, that has (2) and (3) attached to it, and "lives" in (1).

Lindenmeyer Systems

A system of rules used to model growth and development of organisms.

Cellular Automata

Discrete system of cells that iterate based a set of rules (game of Life, etc)

A Life Simulations

Many Simulations based on the algorithms described have been produced.

These Simulations can be divided into two types.

Simulation of a particular life form using one or more algorithms, and *Instantiation* of artificial life (or the bottom-up approach).

Simulation

Examples of simulations of life forms include

- ANTS
- BOIDS
- Gene Pool
- Biotopia

ANTS

Simulation of Ant behavior using a CA where each cell is either a left or right turn.

Ants move through the cells following the turn commands and changing the value of the cell to its opposite.

Instead of attempting to represent a living organism or process, natural laws are invoked in an artificial biosphere.

- Tom Ray's Tierra is an example.
- Avida is another.

BOIDS

Each boid has direct access to the whole scene's geometric description, but reacts only to flockmates within a certain small radius of itself.

The basic flocking model consists of three simple steering behaviors:

1. Separation: steer to avoid crowding local flockmates.
2. Alignment: steer towards the average heading of local flockmates.
3. Cohesion: steer to move toward the average position of local flockmates.

BOIDS - Cont.

In addition, the more elaborate behavioral model included predictive obstacle avoidance and goal seeking.

Obstacle avoidance allowed the boids to fly through simulated environments while dodging static objects.

For applications in computer animation, a low priority goal seeking behavior caused the flock to follow a scripted path.

Gene Pool

Uses a genetic algorithm for animation of creatures based on a response to a stimulus. In the gene pool, creatures try to reproduce by contacting other creatures.

Those creatures who move around better contact more of their fellow creatures and reproduce more.

Biotopia

Similar to Gene Pool except creatures thrive on their ability to find food.

If they don't find enough, they "lose energy" and die.

They reproduce by finding "life" cells which they collect until they have enough to produce offspring.

Instantiation (bottom-up approach)

Instead of attempting to represent a living organism or process, natural laws are invoked in an artificial biosphere.

Tom Ray's Tierra is an example.

Avida is another.

Tierra

"Organisms" in Tierra are machine-language programs.

The organism/program is executed by moving through it's list of instructions.

Instructions in an organism may be altered through mutation or swapping instructions with other organisms and the organism will still be executable.

Tierra - cont.

There are 32 instructions available. Each is specified by a 5 bit number (any possible 5 bit number is therefore valid). Mutation can be caused by “flipping” a bit.

Tierra - cont.

The Tierra system sets up a virtual computer with a CPU and memory. Organisms use the CPU to execute their instructions and reside in portions of the memory.

Tierra - cont.

The process normally starts with a single organism which is self reproducing. It is executed over and over producing copies of itself which are stored in the memory. The Tierra user introduces mutations in some of offspring.

Tierra - cont.

Most mutations produce organisms that don’t reproduce at all. However, occasionally, mutations produce organisms that are better at reproduction.

Tierra - cont.

When the memory of the virtual computer is full, some organisms must be removed. The system removes organisms based on how well the work. For example, it is possible to “define” error conditions for certain executions and remove those organisms that produce errors.

A Life and Evolution

– *Alife a "living system" The distinction between it and carbon based systems is a matter of semantics. As a result, the study of an instantiation of a-life is not a simulation for the purposes of drawing analogies to other living systems.*

Statistical Mechanics and A life Evolution

Observation of the “entropy” of an a-life system allows some understanding of the self criticality of life.

Consider a set of N strings of Ng types such that Ng<N

- *Number of strings(n_i) (at some time, t+1) of a particular type is a function of the rate of replication, probability of string mutating to a different type and probability of a string of a different type mutating into this one.*
- $n_i(t+1)-n_i(t) = (\varepsilon_I - < \varepsilon > - Rl)n_i + (N/N_g)Rl$

where ε is the replication rate, Rl is the mutation rate and $(N/N_g)Rl$ is the rate at which other strings mutate into this one.

There is one type that has the best growth rate (increases its number better than others).

$\varepsilon_{\text{best}}$

Running such a system on Avida, measure its "Shannon" entropy. The entropy related to information theory which can be best defined as the measure of the information content of the system. More simply, a measure of the number of different types of strings.

$S = -\Sigma(n_i/N)\log(n_i/N)$

Entropy over time looks like attached graph.

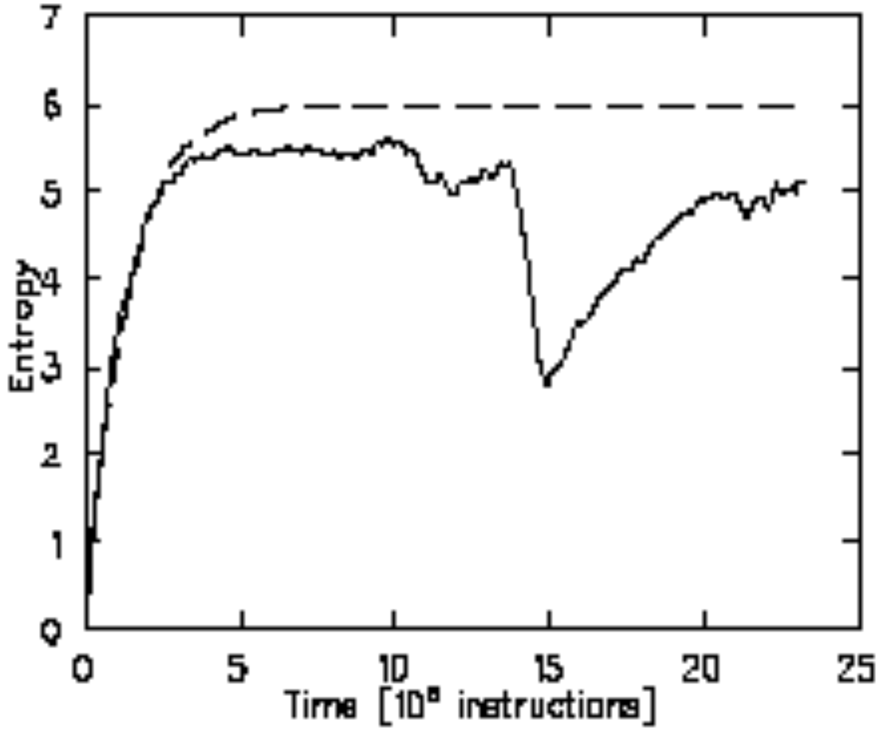


FIG. 1 Entropy of a population of 400 self-replicating (solid line) and non-replicating (dashed line) strings started in the uniform state.

Interpretation: System is in equilibrium except when occasional mutations produce a new type that has a better growth rate that the best so far. Under these conditions, the entropy drops because the new string dominates the system. After a time, the new string produces mutated offspring with similar or lesser growth factors.

This change of state can be considered the equivalent of a sand pile's avalanche and a result of self organized criticality.

Entropy does not always return to its original value. Interpretation: A string can have "cold Spots" and "hot spots". Cold spots are parts of the string that, when mutated, do not increase the growth factor. Hot spots are parts of the string that, when mutated, do increase the growth factor.

When a new string is produced and causes a "change in state", the new string has attained new information for producing a better growth rate.
This information is stored in cold spots for the string.
Since there is more information than in previous, less effective genes, the number of hot spots is reduced.
Information within the string is added at the expense of entropy (information or differences in string types for the whole system).

Fractals and Zipf's Law and A Life

– *Power law distributions are known to exist within certain quantities in living systems. For example, frequency distribution of number of taxa with a certain number of sub taxa. Also, frequency distribution of proteins of different lengths and of codons. Power laws can be indicative of self-organized criticality.*

The taxa example can be expected if there is no time scale governing the length of time a species dominates a population and the number of subfamilies is proportional to the length of time a certain species dominates. In addition, the taxa example is found to have fractal dimensions.

In general, if the distribution of waiting times between events follows a power law, the system is in a self-organized critical state.

- *Analogy: the taxa study. Another example, wait times between entropy drops (changes in state) is measured in a A-life simulation using Tierra (shown in attached graph).*

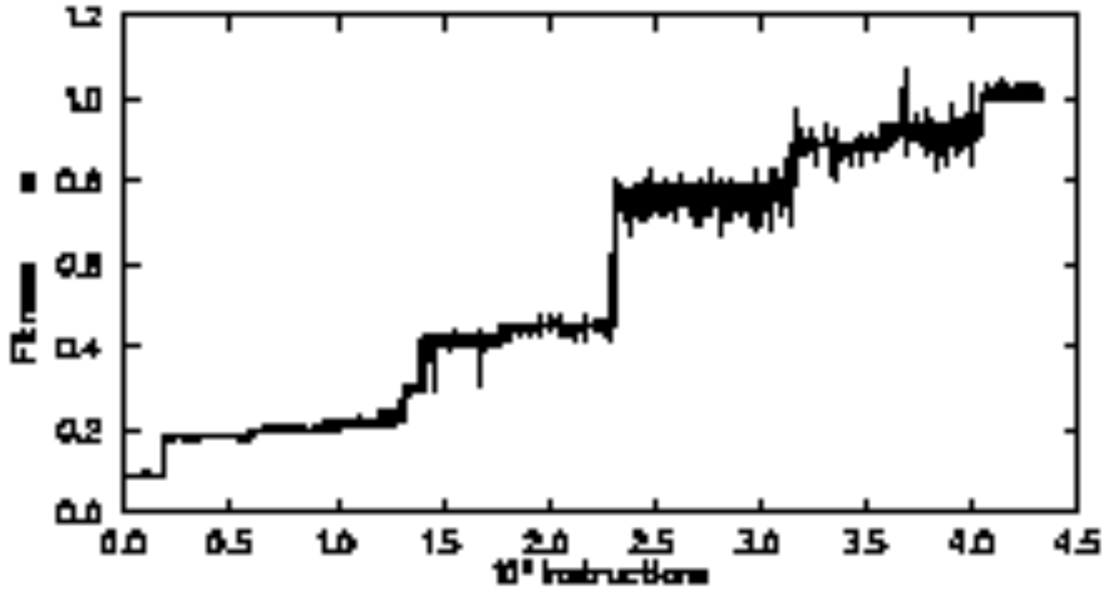


FIG. 2 Fitness curve for a typical run. The (normalized) fitness α of the most successful (i.e., most populous) genotype is plotted as a function of time measured every million instructions for a mutation rate $R = 0.65 \times 10^{[illegible]}$.

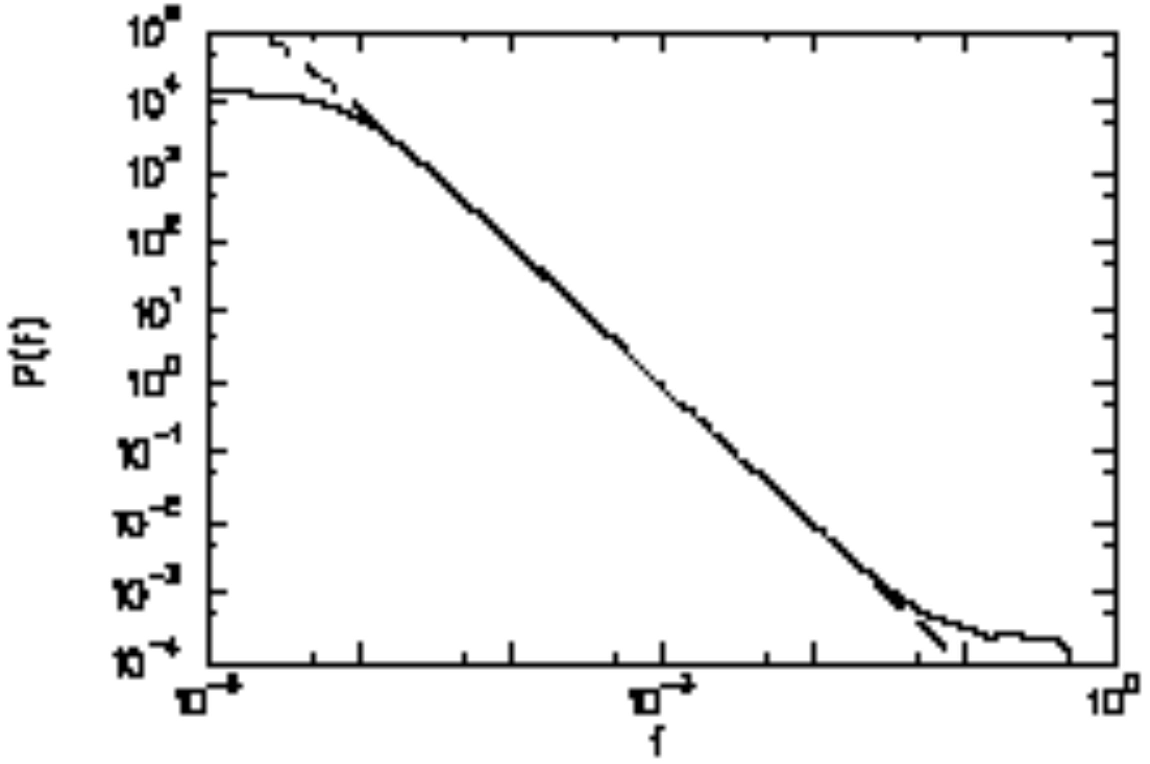


FIG. 3 Power spectrum $P(f)$ of a typical fitness curve $a(t)$ (Fig. 2). The dashed line is a fit to $P(f) \sim f^{-\beta}$ with $\beta = 2.0 \pm 0.05$.

If this power law distribution (in alife example) is fractal, then the same results could be expected for a much longer running simulation. This means the system can make huge jumps and magnitudes and still retain its same structure. In other words, a large number of strings could be wiped out very quickly without external interference.

Resources

- *On Modeling Life, Chris Adami*
  - http://xxx.lanl.gov/abs/adap-org/9405002
- *A Mathematical Theory of Communication. C.E. Shannon*
  - **http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html**
- *Self Organizing Criticality in Living Systems*
  - http://xxx.lanl.gov/abs/adap-org/9401001