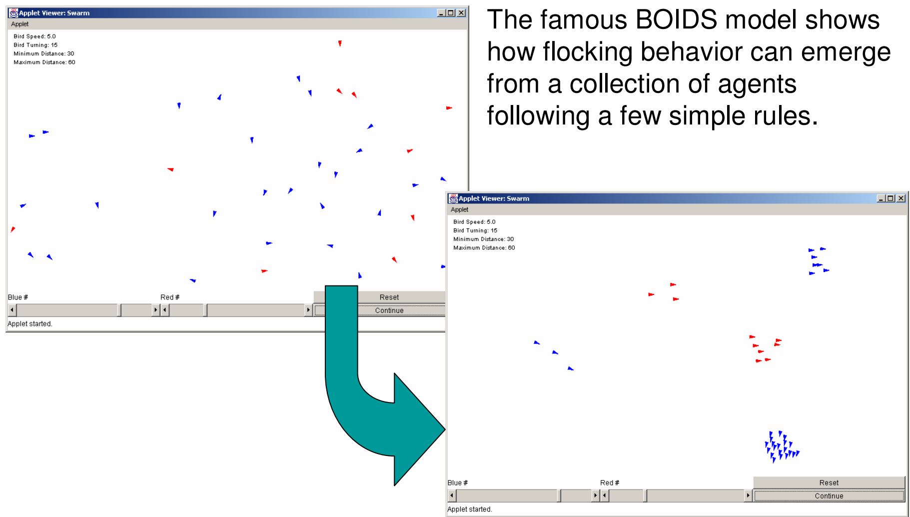
Introduction to Artificial Life and Cellular Automata

CS405

Cellular Automata

- A cellular automata is a family of simple, finite-state machines that exhibit interesting, **emergent** behaviors through their interactions in a population

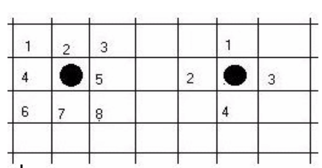
Emergent Behavior



The famous BOIDS model shows how flocking behavior can emerge from a collection of agents following a few simple rules.

Game of Life

- The best known CA is John Horton Conway's "Game of Life".
  - Invented 1970 in Cambridge.
- Objective: To make a 'game' as unpredictable as possible with the simplest possible rules.
- 2-dimensional grid of squares on a (possibly infinite) plane. Each square can be blank (white) or occupied (black).



Moore Neighborhood

von Neumann Neighborhood

Game of Life

- The grid is populated with some initial dots
- Every time tick all squares are updated simultaneously, according to a few simple rules, depending on the local situation.
  - For any one cell, the cell changes based on the current values of itself and 8 immediate neighbors

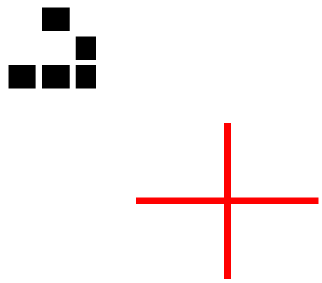
Game of Life Update Rules

- Stay the same if you have exactly two “On” (black) neighbors
- Switch or stay “On” (black) if you have exactly three “On” neighbors
- Otherwise switch to “Off” (white) on the next time step

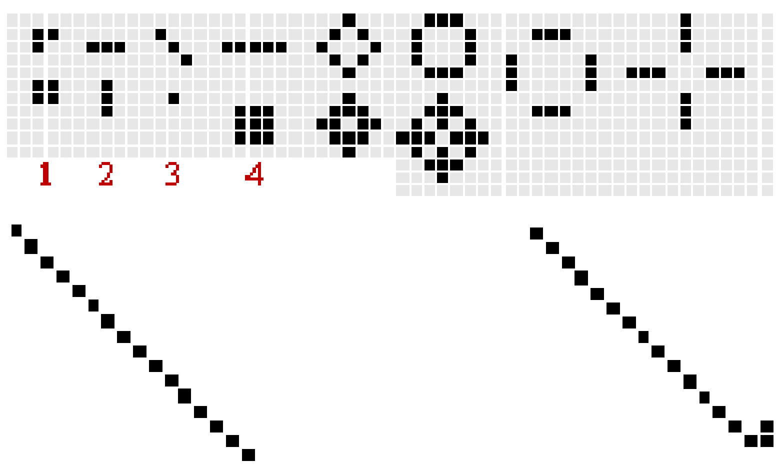
Alternative (equivalent) formulation of Game of Life rules:

0,1 nbrs	= starve, die	2 nbrs	= stay alive
3 nbrs	= new birth	4+ nbrs	= stifle, die

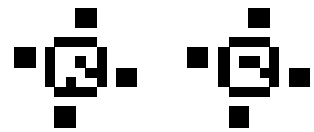
Glider



Sequences



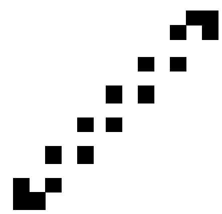
More



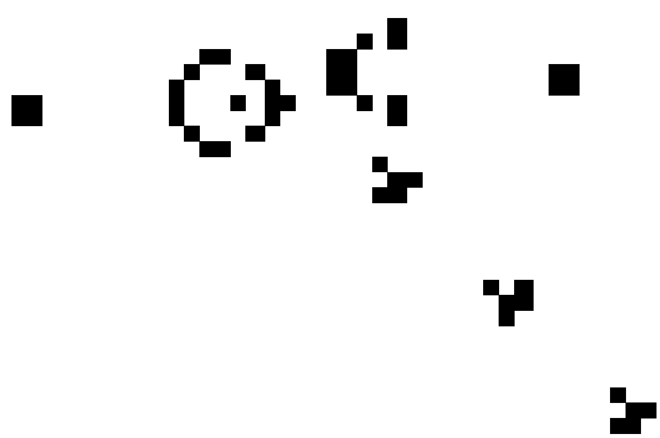
Sequence leading to Blinkers

Clock

Barber's pole



A Glider Gun



More Formal Cellular Automaton

- A set I called the Input Alphabet
- A set S of states that the automaton can be in
- A designated state s_0 , the initial state
- A next state function: $N: S \times I \rightarrow S$, that assigns a next state to each ordered pair consisting of a current state and a current input

- A lattice (e.g. grid)
- of finite automata (e.g. cells)
- each in a finite state (e.g. white or black)

Game of Life - implications

Typical Artificial Life, or Non-Symbolic AI, computational paradigm:
- bottom-up
- parallel
- locally-determined

Complex behaviour from (... emergent from ...) simple rules.

Gliders, blocks, traffic lights, blinkers, glider-guns, eaters, puffer-trains ...

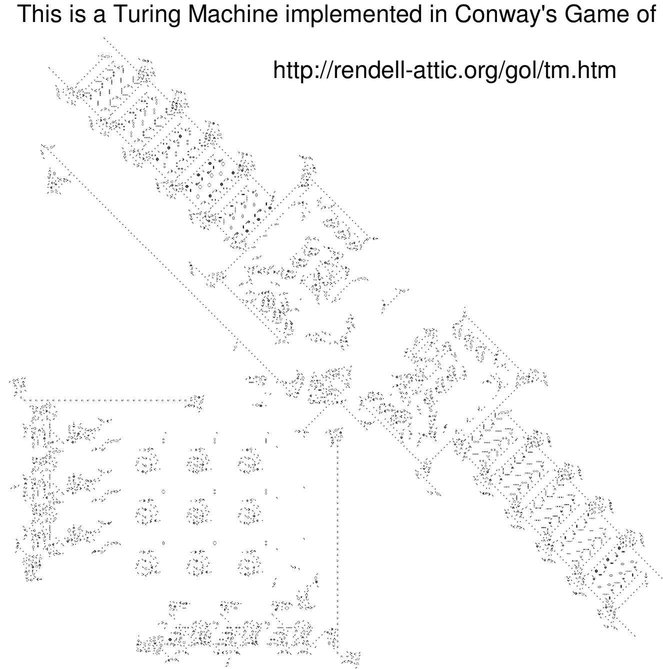
Game of Life as a Computer ?

Higher-level units in GoL can in principle be assembled into complex 'machines' -- even into a full computer, or Universal Turing Machine.

'Computer memory' held as 'bits' denoted by 'blocks' laid out in a row stretching out as a potentially infinite 'tape'. Bits can be turned on/off by well-aimed gliders.

This is a Turing Machine implemented in Conway's Game of Life.

http://rendell-attic.org/gol/tm.htm



Self-Reproducing CA’s

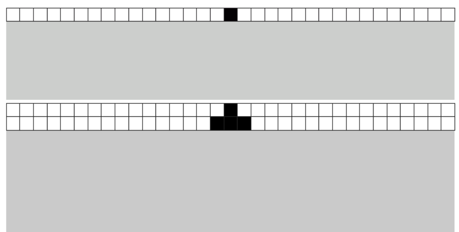
- von Neumann saw CAs as a good framework for studying the necessary and sufficient conditions for self-replication of structures.
- von Neumann’s approach: self-representation of abstract structures, in the sense that gliders are abstract structures.
- His CA had 29 possible states for each cell (compare with Game of Life 2, black and white) and his minimum self-rep structure had some 200,000 cells.

Self-Representation and DNA

- This was early 1950s, pre-discovery of DNA, but von Neumann's machine had clear analogue of DNA which is:
  - Interpreted to determine pattern of 'body‘
  - Contains instructions to copy itself directly
- Simplest general logical form of reproduction (?)
- How simple can you get?

One-Dimensional CA's

- Game of Life is 2-D. Many simpler 1-D CAs have been studied
- For a given rule-set, and a given starting setup, the deterministic evolution of a CA with one state (on/off) can be pictured as successive lines of colored squares, successive lines under each other



Wolfram's CA classes 1,2

From observation, initially of 1-D CA spacetime patterns, Wolfram noticed 4 different classes of rule-sets. Any particular rule-set falls into one of these:-:

CLASS 1: From any starting setup, pattern converges to all blank -- **fixed attractor**

CLASS 2: From any start, goes to a limit cycle, repeats same sequence of patterns for ever. **-- cyclic attractors**

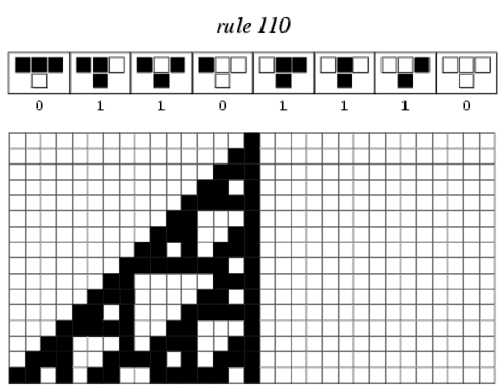
Wolfram's CA classes 3,4

CLASS 3: Turbulent mess, chaos, no patterns to be seen.

CLASS 4: From any start, patterns emerge and continue continue without repetition for a very long time (could only be 'forever' in infinite grid)

Classes 1 and 2 are boring, Class 3 is messy, Class 4 is **'At the Edge of Chaos'** - at the transition between order and chaos -- where Game of Life is!.

Wolfram Rule 110

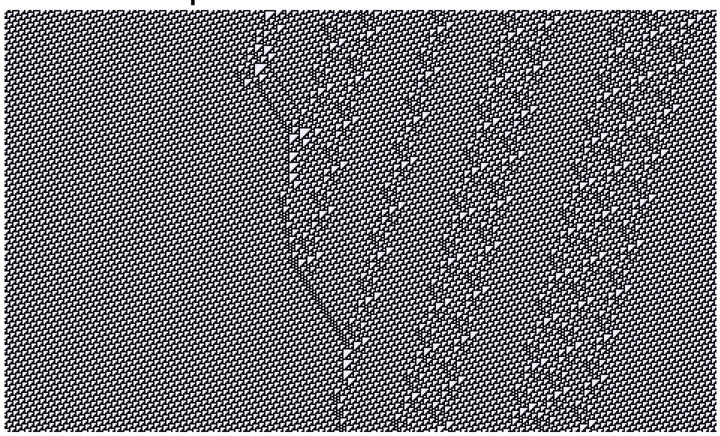


Proven to be Turing Complete - Rich enough for universal computation

interesting result because Rule 110 is an extremely simple system, simple enough to suggest that naturally occurring physical systems may also be capable of universality

Rule 110 Example

- Requires potentially infinite dimensions for general computation



A-Life Applications?

- Tool for mathematically studying emergence from simple, inanimate components
  - “atoms” of an a-life system are defined and physical interactions emerge
- Modeling biological entities, chemistry, pharmacology
  - Chemical multi-cellular morphogenesis

Chemical Morphogenesis Project - 2004

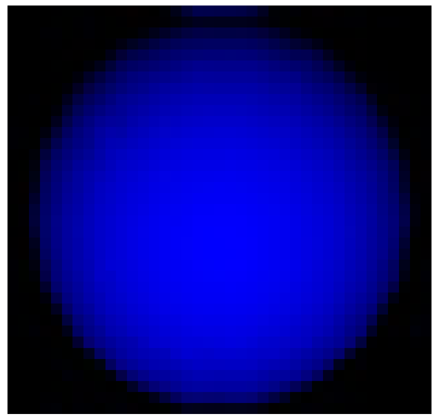
- Three subteams
  - Computer Science: Dr. Mock, Nick Armstrong, and Heather Koyuk
  - Biology: Dr. Gerry Davis
  - Chemistry: Dr. Jerzy Maselko, Heidi Geri
- Three subprojects
  - Implement a 3-D simulation and theoretical model
  - Relate the chemical system to biological systems
  - Implement the chemical system in the laboratory

The Project

- Create a computer simulation capable of modeling multi-cellular chemical and biological growth
- Should model biological and chemical systems as accurately as possible
  - Cells as spherical objects
  - Cells bud or grow in spherical (non-discrete) directions
  - Use both context-free and context-sensitive growth
    - Easy to write a program that ‘simulates’ growth
    - Harder to use grammars to create a specific unique pattern

The Agents

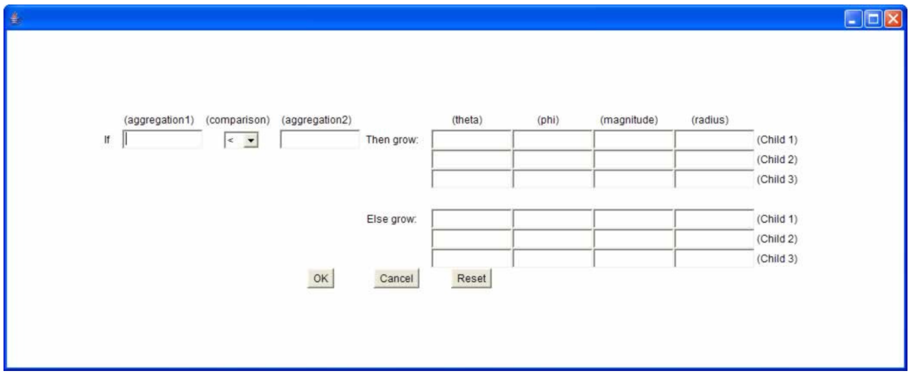
- 3D spheres, uniform radii
- Magnitude (state)
- Spherical growth vectors
- Current model:
  - Sessile, rigid
  - Die/become dormant after budding
- Not limited to the above!



The Rules and Actions

- Rules comprise a grammar
- Context-free
  - Unaware of neighbors; behavior based on state
- Context-sensitive
  - Behavior based on state & state of neighbors
- Actions:
  - Implemented: Budding
  - Working on: Cell Division
  - Others: Motility, growth, non-uniform shapes, etc.
- Dynamic rule creation (via user interface)

Dynamic Rules Creation



Research Overview

- Morphogenesis
  - Lots of plant morphogenesis research: L-systems, etc.
  - Chemical morphogenesis: Mostly chemical reaction/diffusion

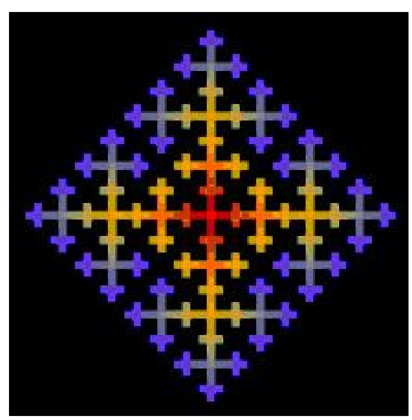


Image source: Fowler, D., and Prusinkiewicz, P. “Maltese Cross.” 1993. Visual Models of Morphogenesis/ Algorithmic Botany at the University of Calgary. 4/14/05. <http://algorithmicbotany.org/vmm-deluxe/Section-07.html>.

Research Overview

- Cellular Automata
  - Begin with grid of cells
  - Usually 1-D, some 2-D
  - Binary/discrete state variables (‘on’ or ‘off’)
  - Cells change state based on their current state and state of immediate neighbors
- Our cells:
  - Do not fill grid
  - 3-Dimensional and can grow in any direction
  - Continuous state variables (not discrete)

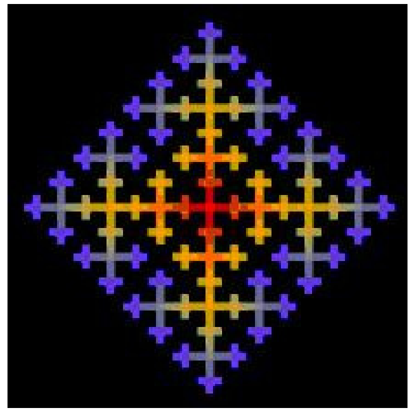
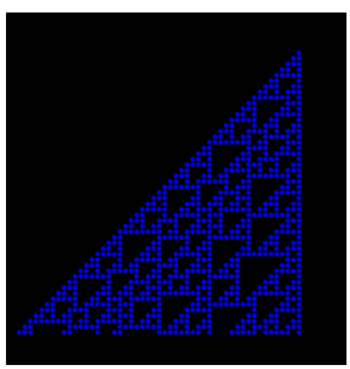


Image source: Fowler, D., and Prusinkiewicz, P. “Maltese Cross.” 1993. Visual Models of Morphogenesis/ Algorithmic Botany at the University of Calgary. 4/14/05. <http://algorithmicbotany.org/vmm-deluxe/Section-07.html>.

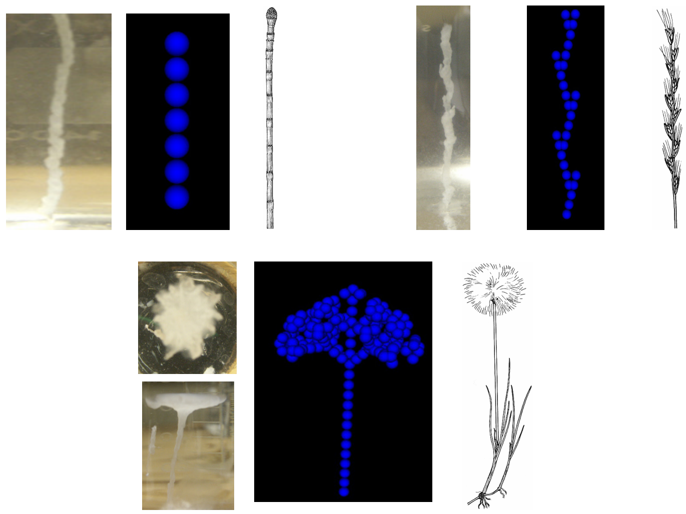
Cellular Automata

- Our cells are capable of everything a cellular automaton is, and more!

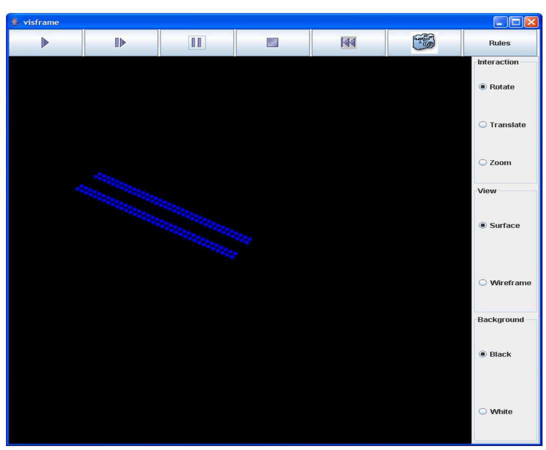


Wolfram’s Rule 110

Context-Free

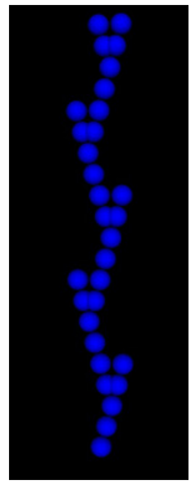


Context-Sensitive



Problems/Questions

- Infinite search space for possible rules
  - How to narrow down and find interesting ones?
- Dynamic rule specification
  - Entails specifying, executing a grammar during run-time
- Backward problem
  - For a given macrostructure, how to define a rule set to produce that structure?
- Expand code functionality
  - Budding/Cell Division, Cell Growth/L-Systems, Motility, Pliability



Future Directions/Answers

- Create a language for specifying rules
- Use genetic algorithms to find interesting rules, and to solve backward problem
- Examine division/budding, motility, cell growth, L-Systems, and pliability separately and in great depth
- Keep trying to reproduce basic biological structures (e.g. developing embryo) in model and in lab

Conclusion

- This project has widespread implications
  - Biology
  - Chemistry
  - Computer science
  - Complexity
- We've laid the groundwork
- But we've only scratched the surface!