Introduction
Computational Science

Introduction Computational Science

by Lera (Valeria) Krzhizhanovskaya

V.Krzhizhanovskaya@uva.nl

L4. 2-dimensional CA, Game of Life

Computational Science

L1. Intro Computational Sci (recap)

- **The 3rd pillar of science**
  - Experiment
  - Theory
  - Modelling & simulation
- **Why model? and what?**
- **System, experiment, model, simulation**
  - Only an idiot uses simulation in place of <.?.>
  - Don't fall in love with your model! Danger!
- **Validation, verification**
- **Types of models**

L2. Cellular Automata. 1D (recap)

- **What is CA? Why study? Applications?**
- **Range *r*, Neighbourhood *N=2r+1*, *k* States Σ,**
- **Input alphabet α, Size $m=k^N$**
- **k^m Transition functions Δ (a.k.a. Rules)**
- **Rule numbering by "Wolfram code"**

  Rule 30: $30 = 00011110_2$

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

- **Wolfram classes**

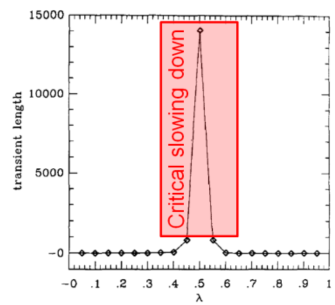
  1 homogeneous, 2 stable pattern, 3 chaos, 4 complex

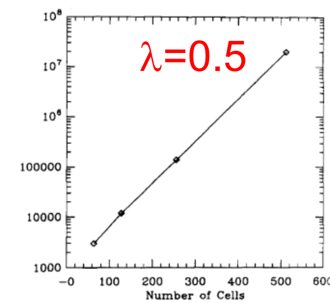
- **Transients and cycles** 001 → 011 → 101 → 110 → 011 → 101 →…

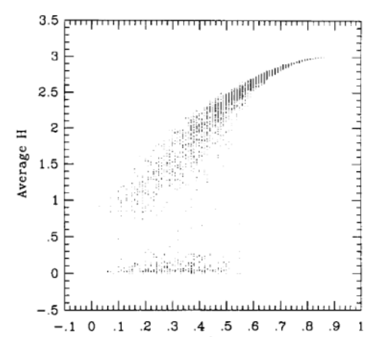
L3. CA quantify complexity (recap)

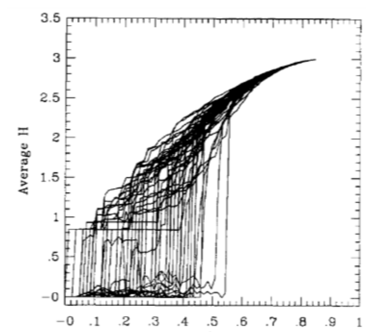
- **Langton parameter** $$\lambda(\Delta) = \frac{k^N - n}{k^N}$$
  - Limits: λ=0, λ=1, λ = 1-(1/k) equally represented
  - 2 sampling methods: random & walk-through
  - Where is the complexity observed?
- **Complexity measures**
  - Transient length
    - Critical slowing down
  - Shannon entropy

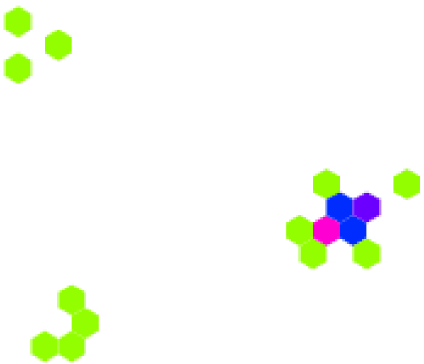
$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

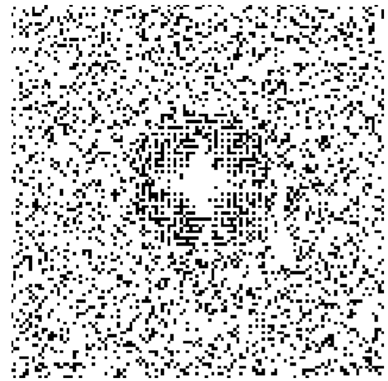


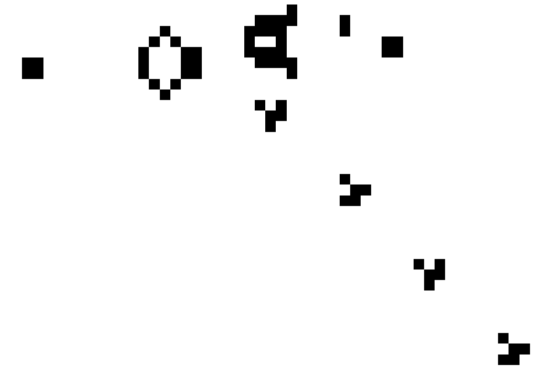




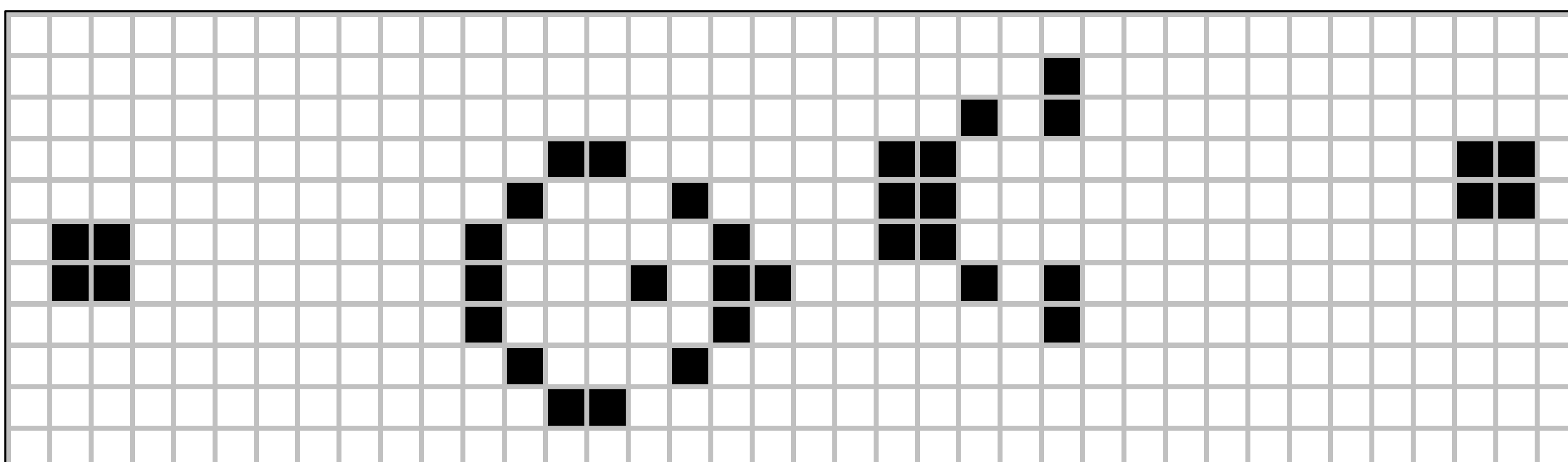








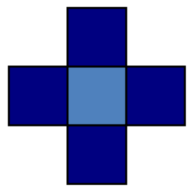
2-dimensional CA.
Game of Life.
Universal Computation



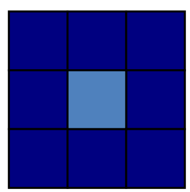
Computational Science

Two-dimensional CA

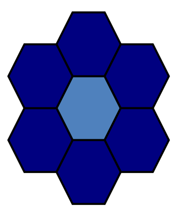
- **Lattice is a 2D grid.**
- **Commonly-used neighborhoods:**



von Neumann

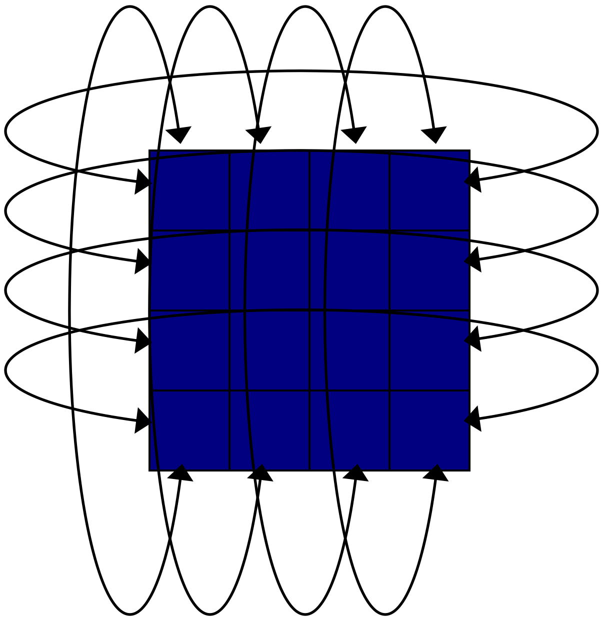


Moore

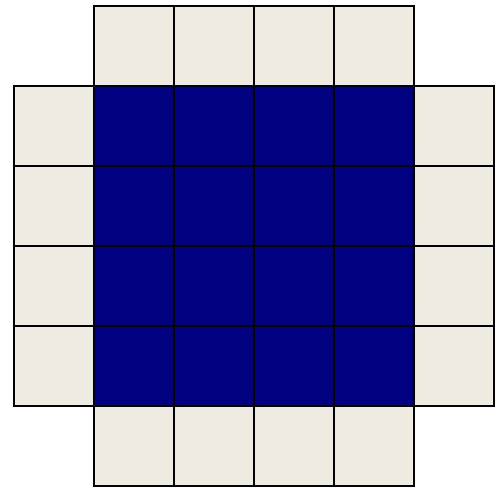


Hexagonal

Boundary conditions



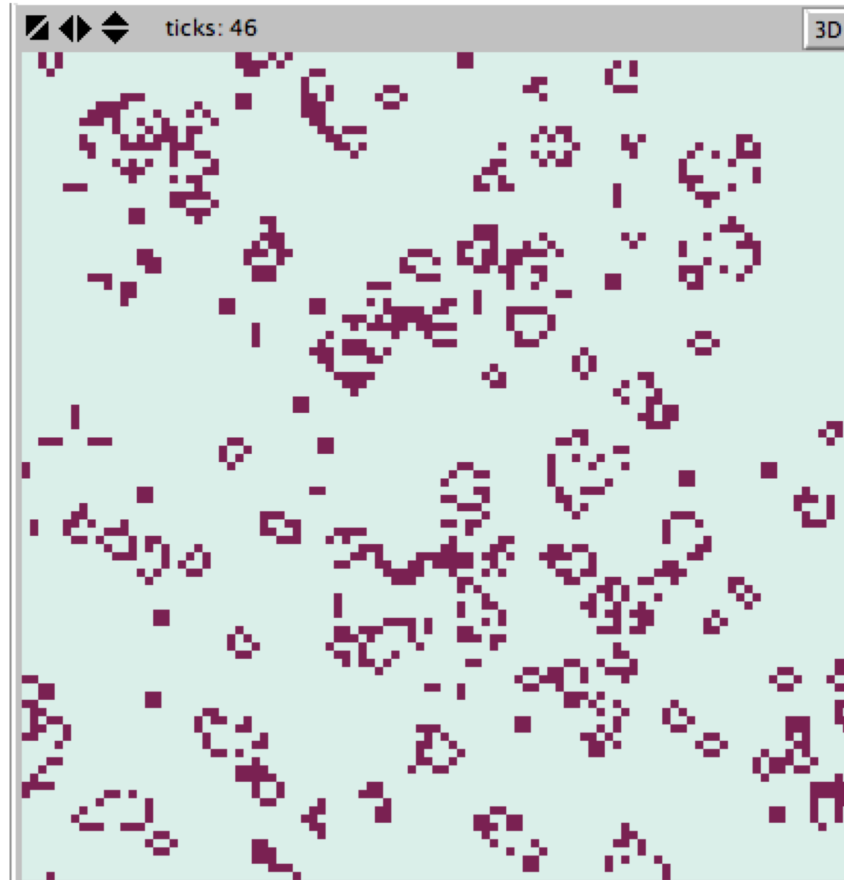
Periodic



Blocking

Conway's Game of Life

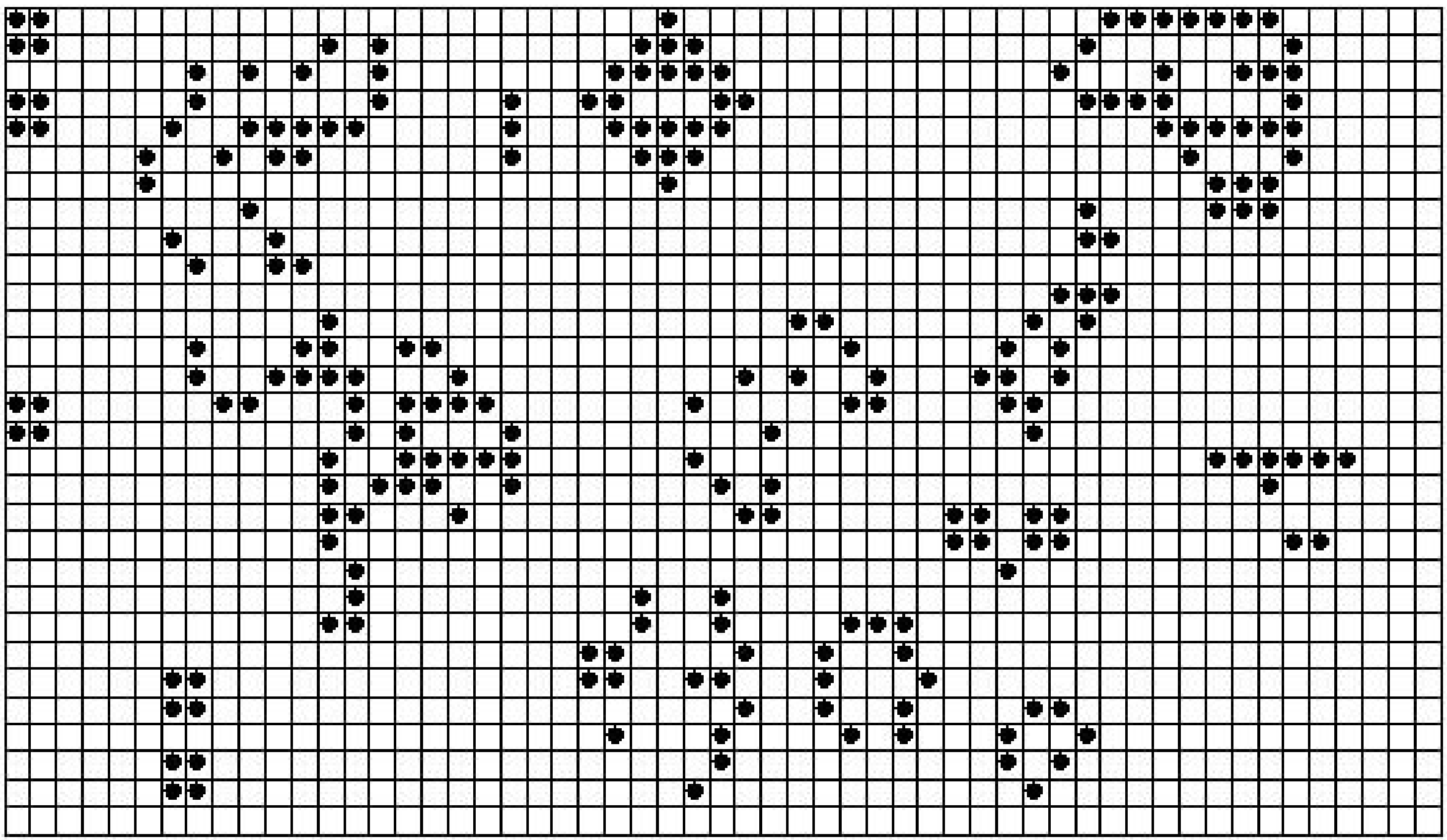
John H. Conway



Life.nlogo

Conway's Game of Life

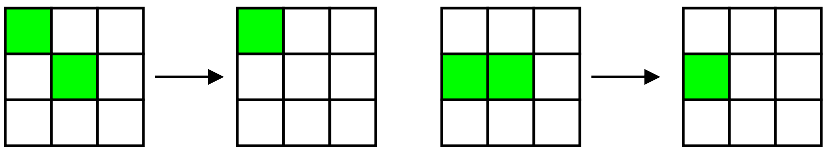
- **Moore neighborhood**
- **Cells are alive or dead**



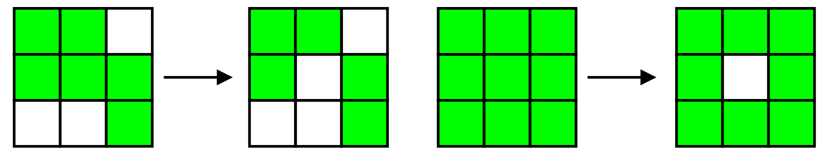
Computational Science

The Rules

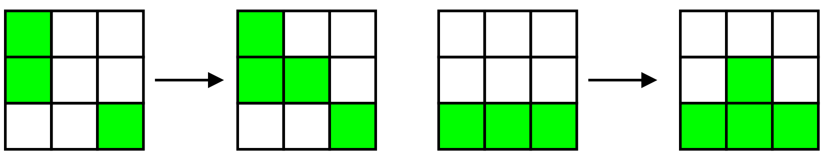
Loneliness: < 2 neighbors alive



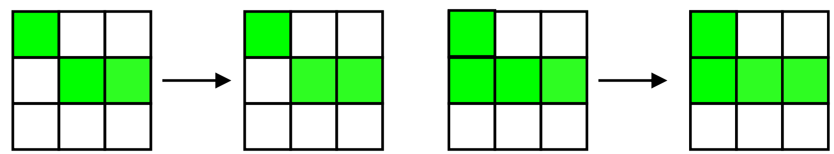
Over crowding: > 3 neighbors alive



Birth: = 3 neighbors alive



Survival: = 2 or 3 neighbors alive



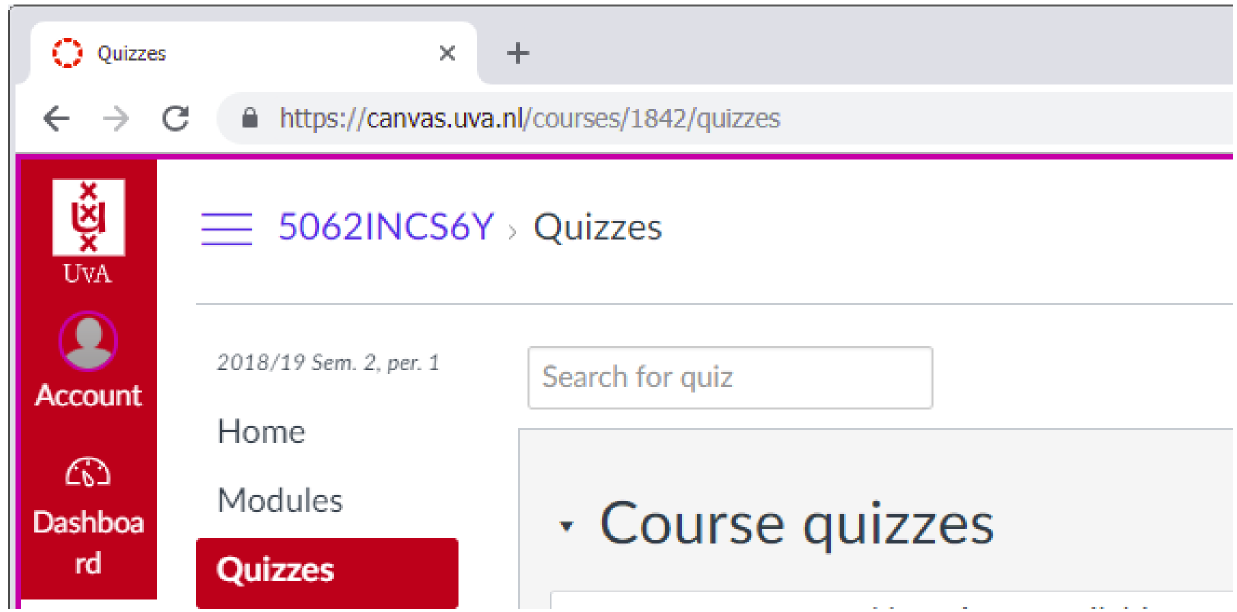
Quiz: L4Q1, Q2

Q1

1	1
0	1

Q2

0	1	0
0	1	0
0	1	0



Why is Life so interesting?

- Early example of emergence, self organization
- Elaborate patterns/dynamics come from simple rules
- "Unlike most computer games, the rules themselves create the patterns, rather than programmers creating a complex set of game situations."

http://conwaylife.com/

http://www.conwaylife.com/wiki/Conway's_Game_of_Life

Where did Life come from?

- **Ideas from John Von Neuman – interested in machines that could replicate themselves.**
- **Colonize mars (planets) by first sending machines that farm iron, build replicas of themselves, get more iron etc.**
- Is this possible, or do you need more sophisticated machines to build simpler machines, etc.

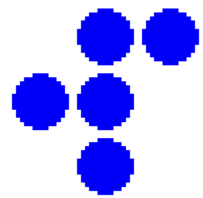
- Tackled this as a mathematician – not an engineer.

The man who finds Game of Life Boring…

JOHN H. CONWAY

Complex Emergence

- Hard to predict behavior mentally:

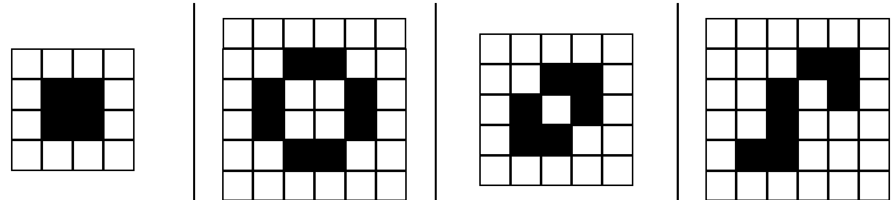


- R-Pentomino – how will this evolve?
- Conway simulated this by hand… does it become stable?

Life.nlogo

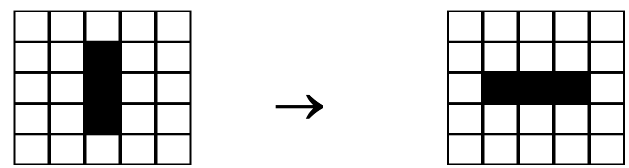
Patterns of Game of Life

- You can observe **patterns** in GOL:
  - Static/Still life



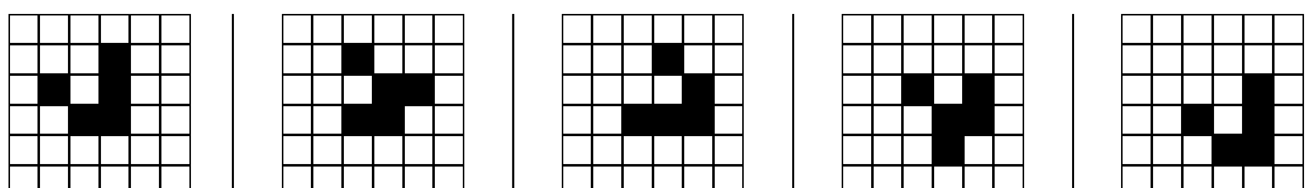
block | pond | ship | eater

  - Periodic/Oscillators



time = 1 → time = 2

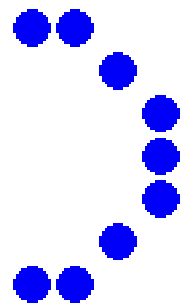
  - Moving

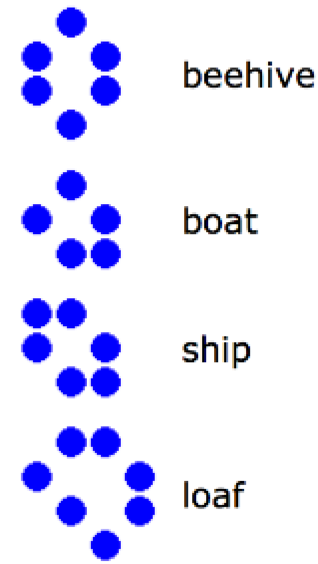


time = 1 | time = 2 | time = 3 | time = 4 | time = 5

The Queen Bee Shuttle

- **Hard to predict behavior mentally:**

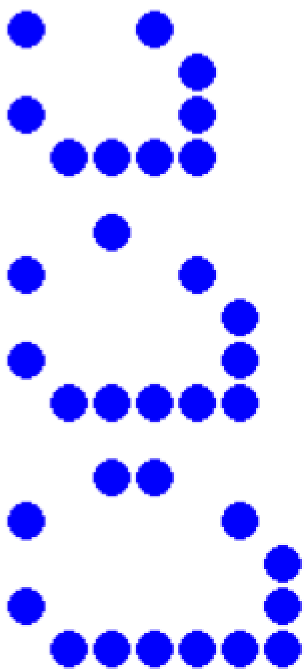




- **Conway called this queen bee shuttle**

Life.nlogo

More interesting early discoveries

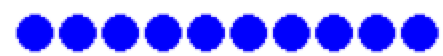


light weight

medium weight

heavy weight

pentadecathlon:

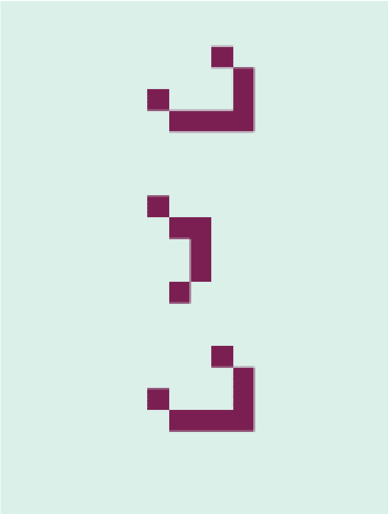


Life.nlogo

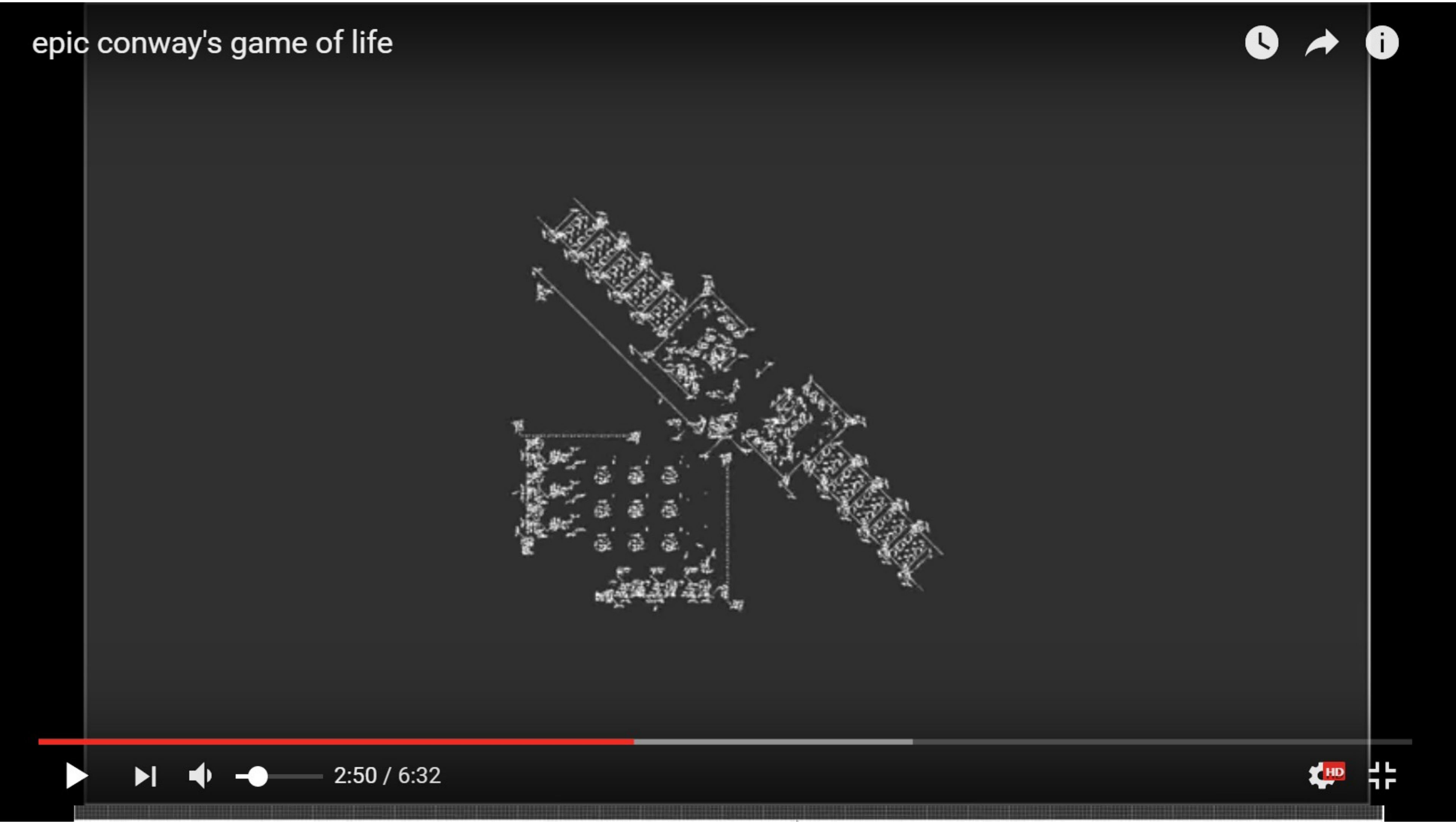
All stabilize?

- **Conway offered a prize for example patterns that go on forever.**

- **The Puf Train**



Some people spend their weekends…



Computational Science

source: https://youtu.be/C2vgICfQawE?t=69

Game of Life and Turing Machines

UNIVERSAL COMPUTATION

Universal Computation?

- **Show that Life can emulate a Universal Turing machine - then life can calculate all algorithms!**
- **Show that life can create:**
  - A finite-state control (with clock)
  - A tape (with memory)
  - A tape head

Building parts…

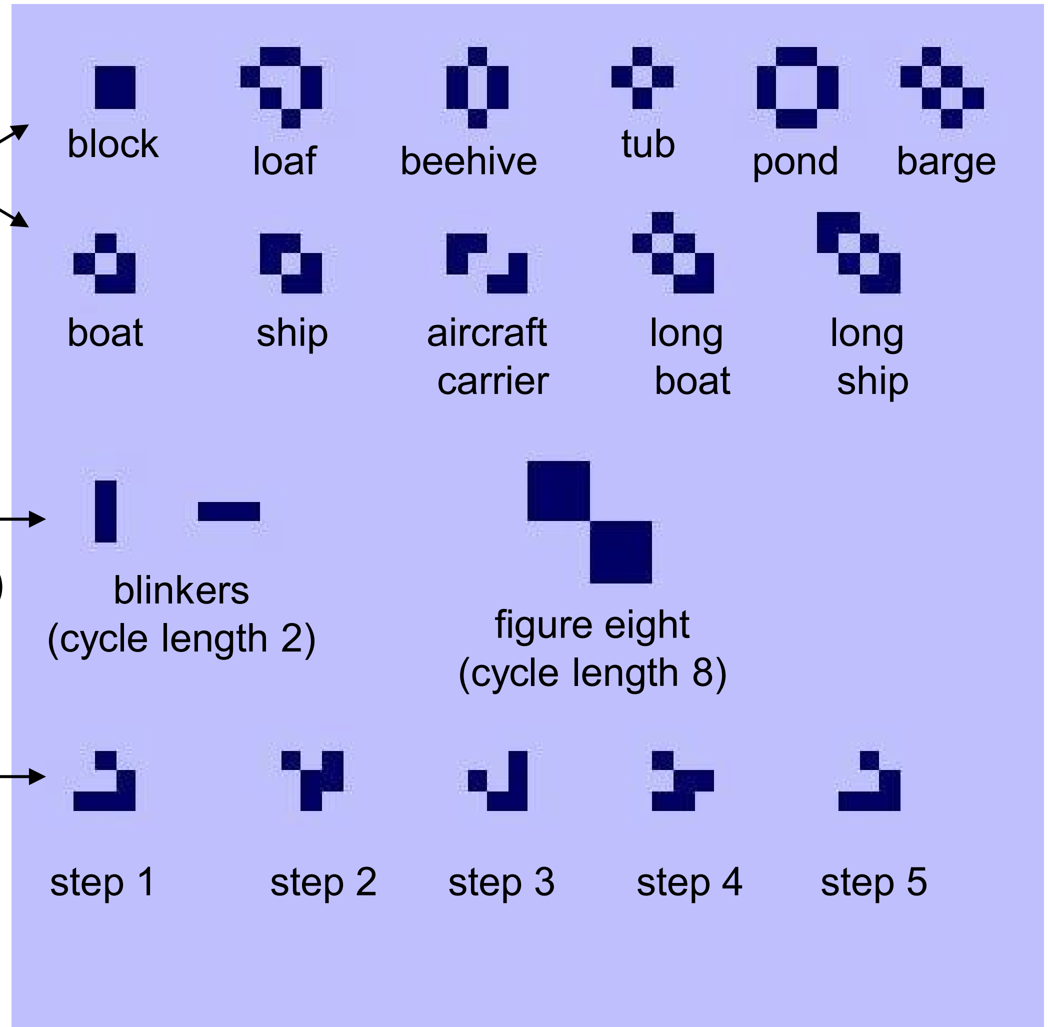
- Need configurations that can be used as clocks, memory, etc.
- We choose a few configurations to help us…

Stable and Glider…

Stable Life
(cycle length of 1)

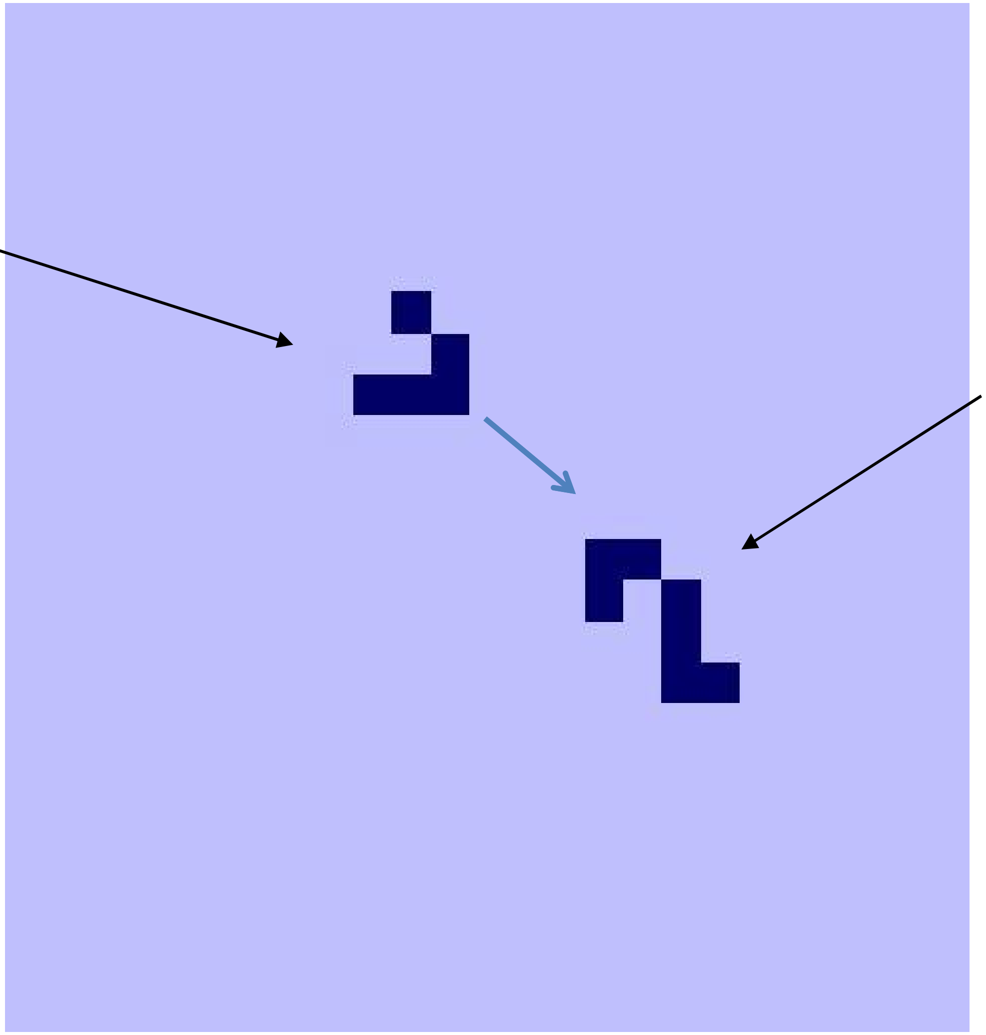
Periodic Life
(other cycle lengths)

Gliders
(translates position
down and to the
right every
4 steps)



Glider Eater

Glider which will move southeast towards the "eater".



A glider "eater". A stable configuration that swallows the glider and then reconstructs itself.

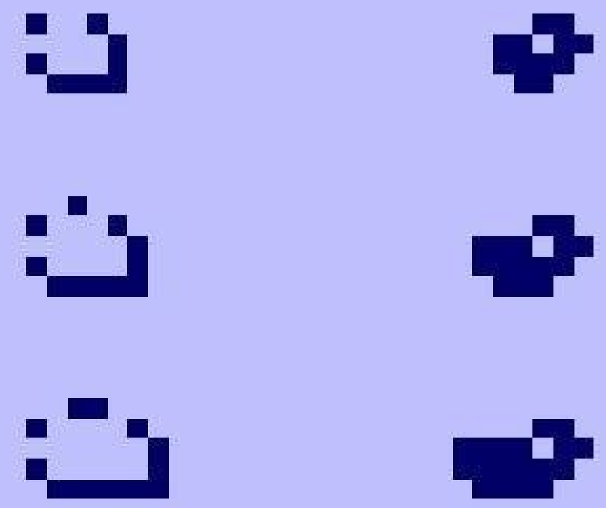
Spaceships

Spaceships move to the right!

lightweight spaceship →

middleweight spaceship →

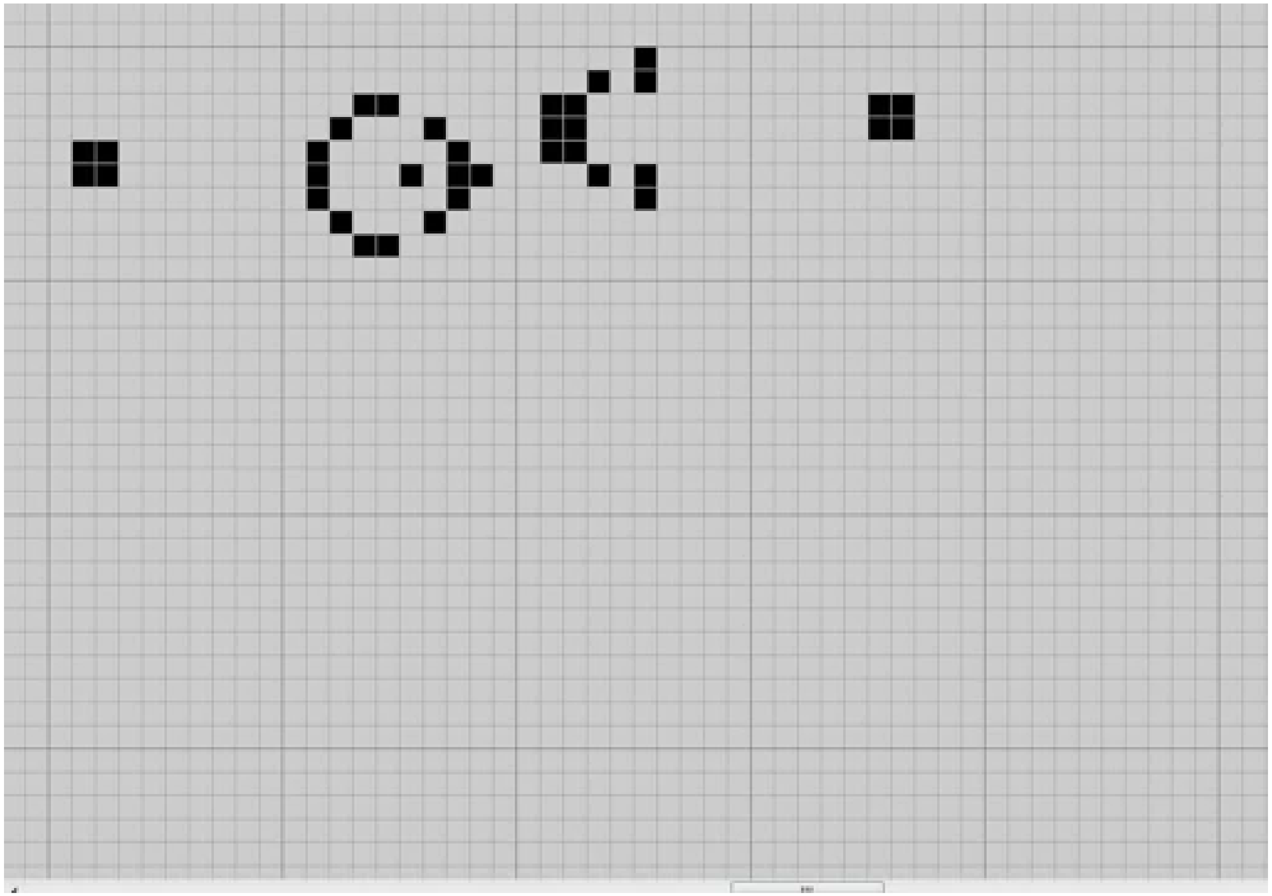
heavyweight spaceship →



step 1 step 2

All have cycle length of 4.

Glider Gun



This produces an endless stream of gliders, one every 30 steps

Building Blocks (to build a TM)

– A finite-state control (with clock)
– A tape (with memory)
– A tape head

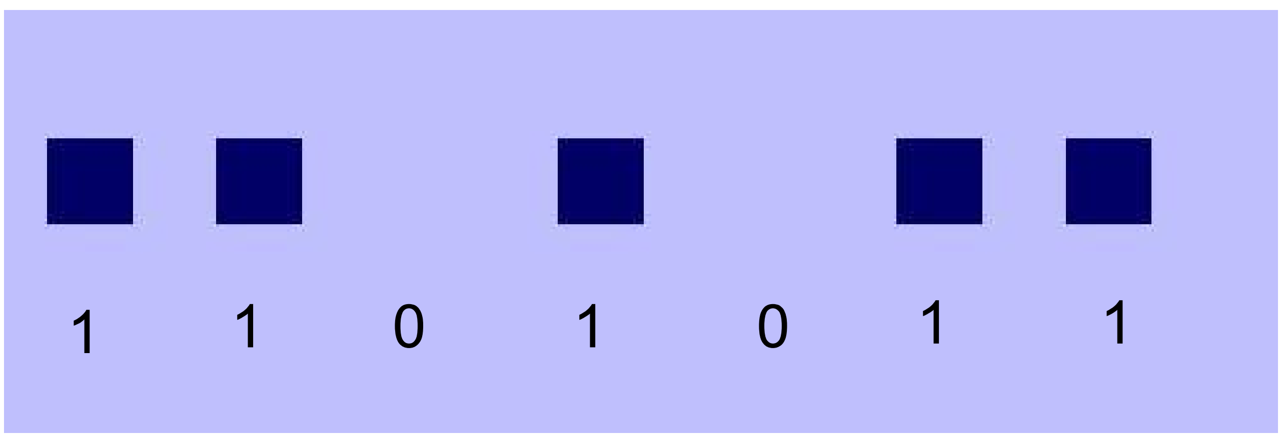
Functional requirements (in general)

- Arbitrary information **storage**
  - (whichever information, however long, at any time)
- Arbitrary information **transfer**
  - (from/to anywhere, at any time)
- Universal information **modification** (such as NAND-gate)
  - Combines with storage and transfer to construct any Boolean function

Memory…

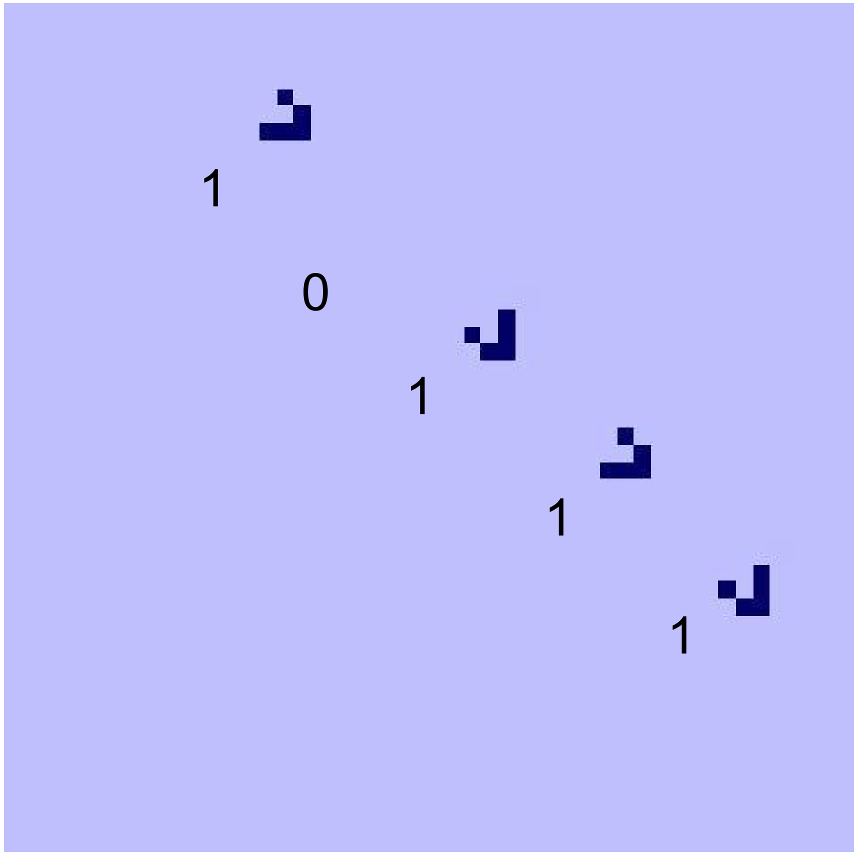
- **Arrange stable blocks or stable patterns to store/remember things**

- **Obvious step to binary storage…**



Memory…more

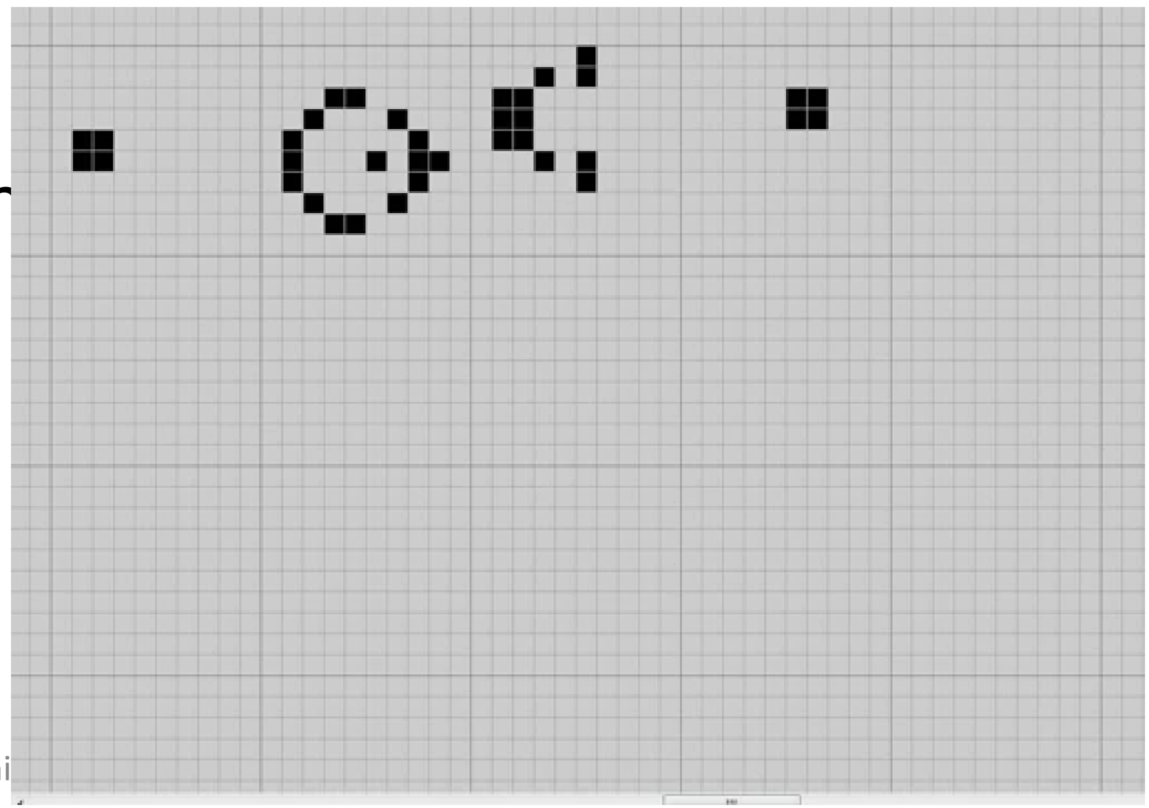
- Or we can represent memory with gliders, they also transmit information



Clock

- **We have the time of the CA (steps)**
- **Need to be able to move objects at a set rate… so can move the tape head and transmit information**

- Gliders and spaceships can move
- We need to generate them
- **Glider Gun**



Finite State Control

- The effect of finite state control in a Turing Machine is to take an input string and generate an output string on the tape
- We can consider binary strings without loss of generality
- So, the effect of finite state control is a Boolean function, maps binary to binary

Finite State Control

- **In the Game of Life we can create any Boolean function**
- **If we can prove that we can generate any Boolean function then we can create the function that gives a Universal TM finite-state control**

Boolean Functions

- **A function from a string of 0's and 1's to another of 0's and 1's**
  - $f : \{0,1\}^n \rightarrow \{0,1\}^m$

- **Possible to build a function**
  - $f: \{0,1\}^n \rightarrow \{0,1\}^m$, from a set of m functions $f_i:\{0,1\}^n \rightarrow \{0,1\}$
  - $f(x_1,x_2,\ldots,x_n) = (f_1(x_1,x_2,\ldots,x_n)\ ,\ f_2(x_1,x_2,\ldots,x_n)\ ,\ldots,\ f_m(x_1,x_2,\ldots,x_n))$

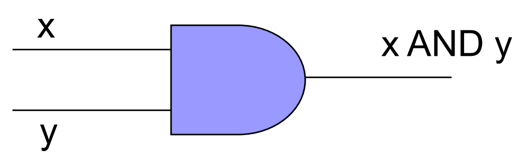
Boolean Functions: Logic gates

- **Real computer chips are built using Logic gate:**
  - AND, OR, NOT, XOR, NAND, NOR, etc.
    - All map $\{0,1\}^2 \rightarrow \{0,1\}$

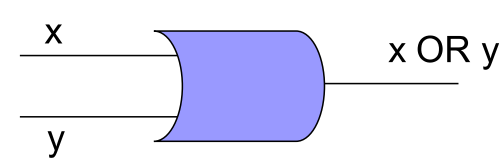
- **Most important ones:**
  - AND, OR, NOT

Boolean Functions: Logic gates

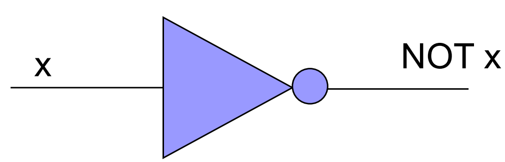
x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1



x	y	x AND y
0	0	0
0	1	1
1	0	1
1	1	1



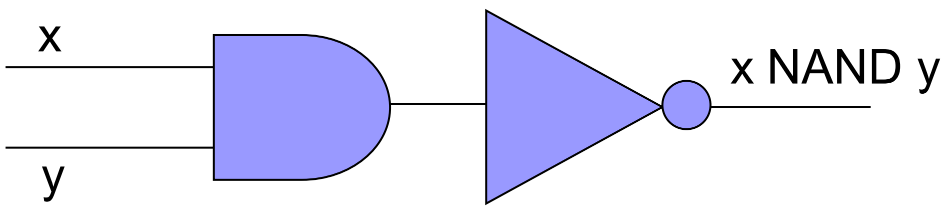
x	NOT x
0	1
1	0



All possible Boolean Functions?

- **Can combine…**
- **How to create NAND? (NOT AND)**

x	**y**	**x NAND y = NOT(x AND y)**
0	0	1
0	1	1
1	0	1
1	1	0



Universal gates: AND, OR, NOT

- **Theorem: AND, OR and NOT are sufficient for calculating all Boolean functions**

- **Proof: Start with example, consider a Boolean function *f* as shown in table of next slide.**

Universal gates: Proof

#	x_1	x_2	x_3	f
1	0	0	0	0
2	0	0	1	1
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1

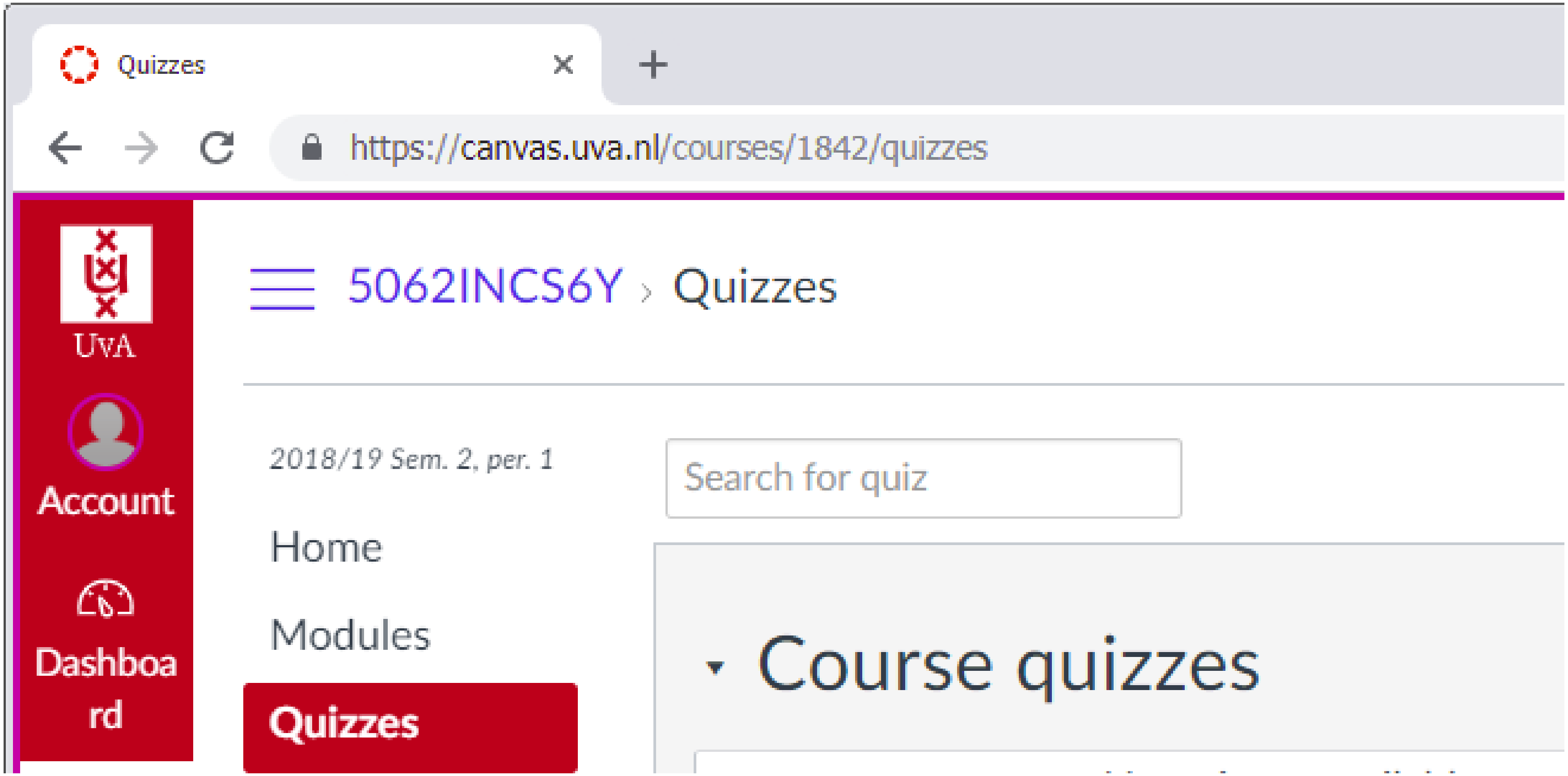
- **For each line where the output is a 1, we can construct a Boolean function that mimics *f***
- **Line 2 would be:**

  (NOT x_1) AND (NOT x_2) AND x_3
- **This will have value 1 when x_1= 0, x_2= 0, and x_3= 1.**
- **Line ??**

  x_1 AND x_2 AND x_3

  (NOT x_1) AND x_2 AND x_3

Quiz: L4Q3



Quizzes
https://canvas.uva.nl/courses/1842/quizzes
UvA
Account
Dashboa rd
5062INCS6Y › Quizzes
2018/19 Sem. 2, per. 1
Home
Modules
Quizzes
Search for quiz
Course quizzes

Universal gates: Proof

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

- **We can OR all true rules together to get *f*. *Disjunctive Normal Form* of *f***

((NOT x_1) AND (NOT x_2) AND x_3) OR
((NOT x_1) AND x_2 AND x_3) OR
(x_1 AND x_2 AND x_3)

We can generalize to an arbitrary Boolean function with n inputs. Now recall f : $\{0,1\}_n \rightarrow \{0,1\}_m$ can be constructed from a set of m functions $f_i : \{0,1\}_n \rightarrow \{0,1\}$ (n=3 here). And each f_i can be constructed in disjunctive normal form (as shown above). Therefore, the entire function can be built from a collection of disjunctive normal forms which are just a collection of NOT, AND, and OR gates.

Universal gates: NAND

- **Theorem: NAND is a universal gate by itself. We actually need only 1 gate: NAND, then we just need to show GOL can build a NAND!**
- **Proof: We can build AND, OR, NOT from NAND.**

x AND y = (x NAND y) NAND (x NAND y)

x OR y = (x NAND x) NAND (y NAND y)

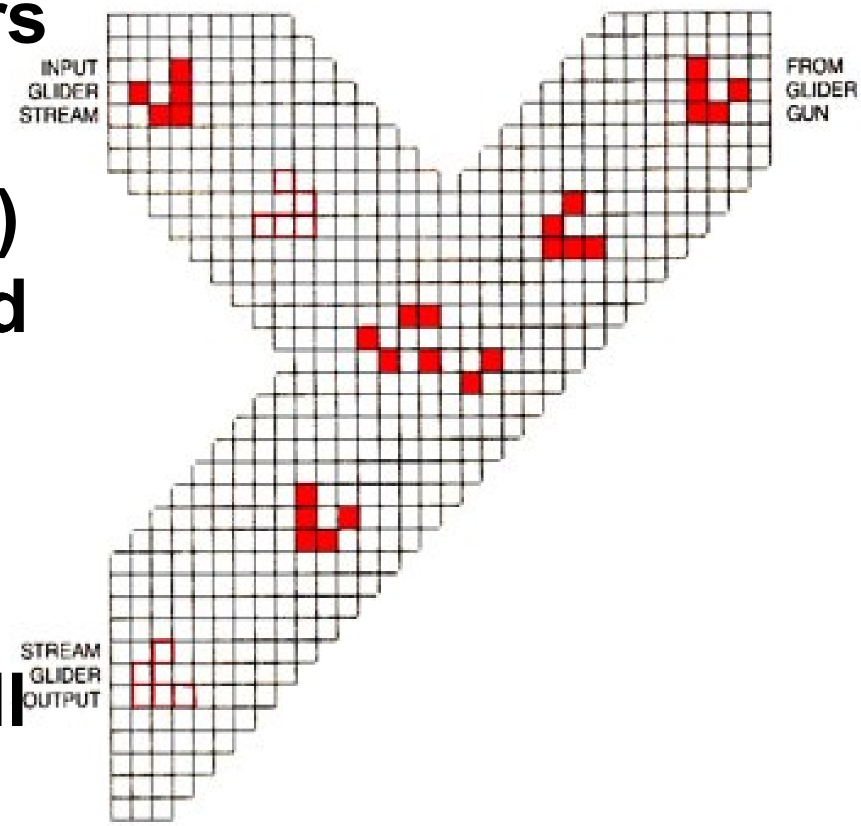
NOT x = x NAND x

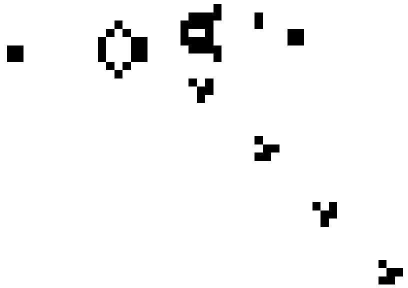
Next steps?

- **We now know that a universal Turing Machine's finite state control can be implemented as a series of NAND gates.**
- **In order to prove GOL is equivalent to a UTM:**
  - Clock (periodic states)
  - Memory (Stable or moving)
  - Finite State Control (?)
- **This means we only now need to show that GOL can create NAND gates (NOT + AND).**

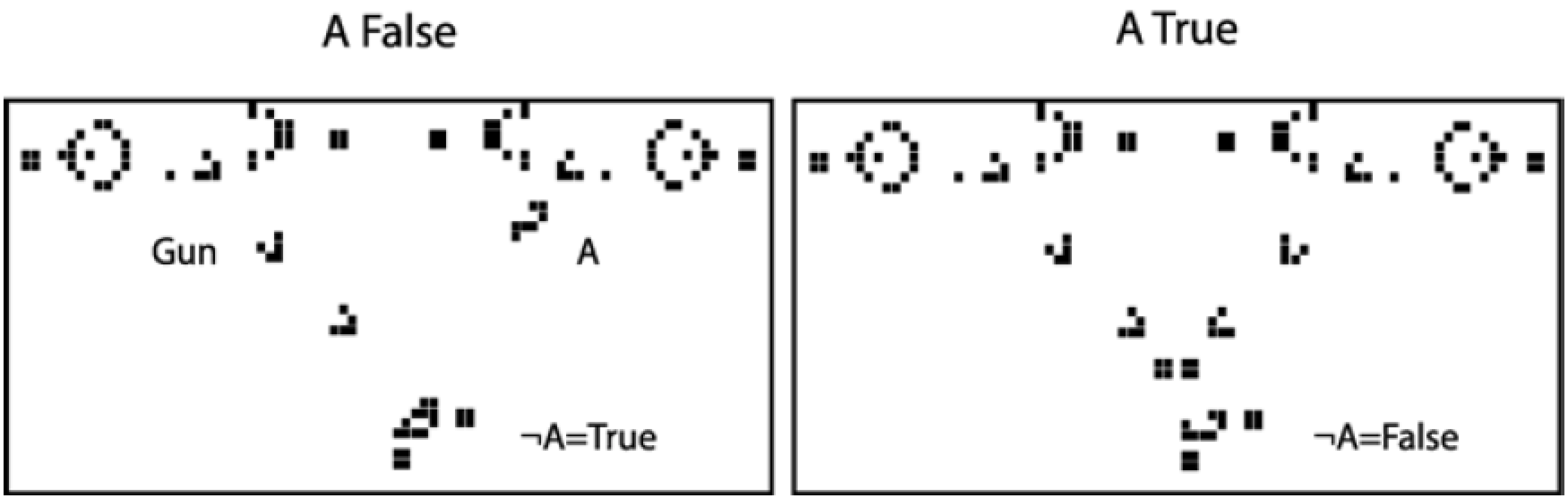
GoL “NOT” gate

- **Consider a stream (s1) of data constructed by gliders**
  - glider =1, no glider = 0
- **Create another stream (s2) of gliders that is all 1’s and orient at right angles.**
- **Two streams collide and annihilate each other**
- **Second stream (s2) will kill 1’s and 0’s of s1 will go through as 1’s**





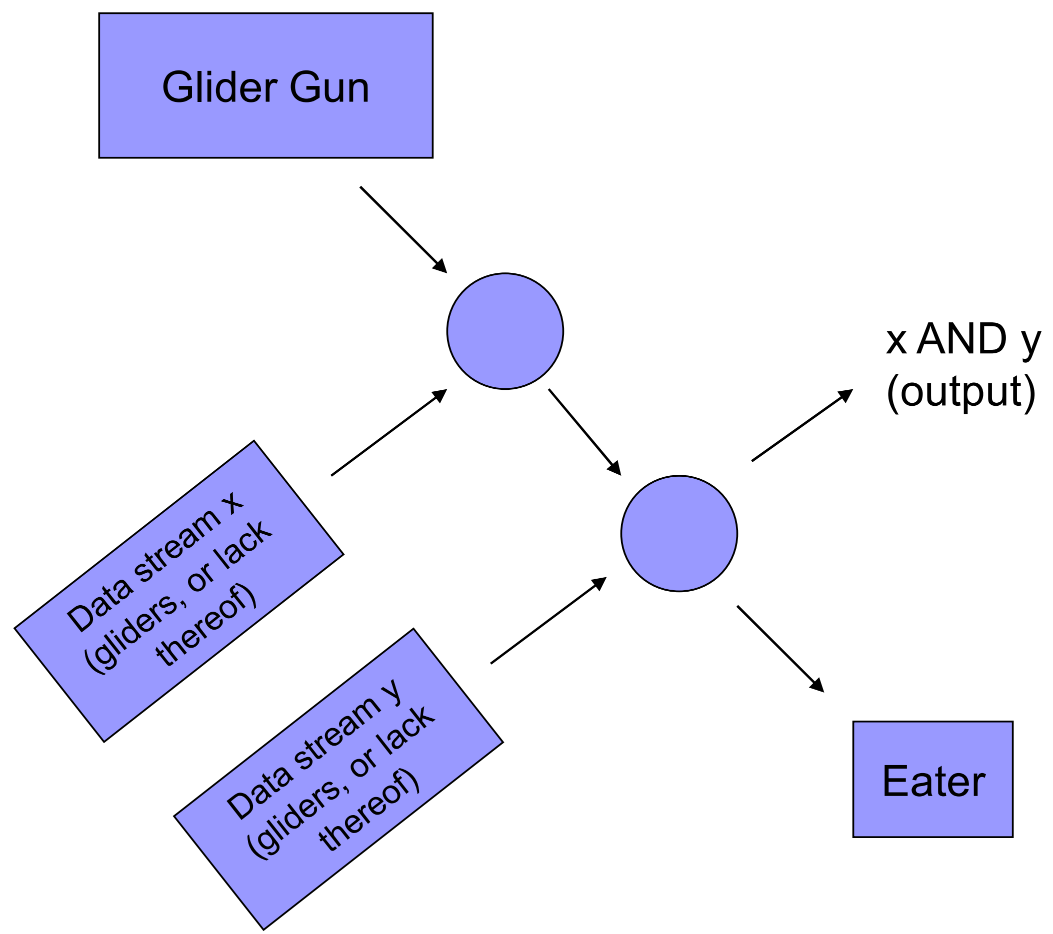
GoL “NOT” gate



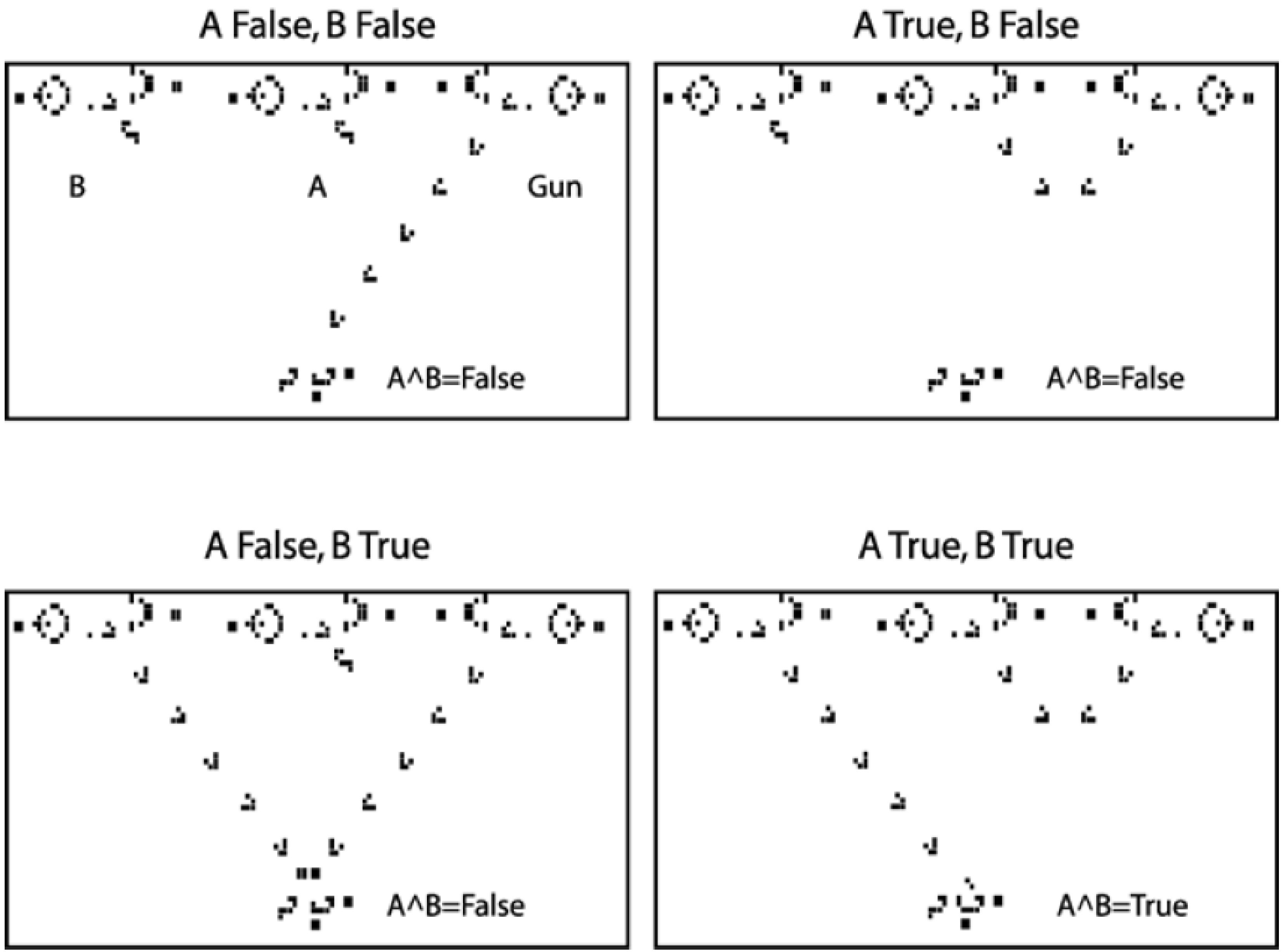
See file GOL_Gates.pdf in Canvas

GoL "AND" gate

- Two input glider streams x and y collide with all 1 glider gun.
- Circles are location of collisions
- Eater collects extras

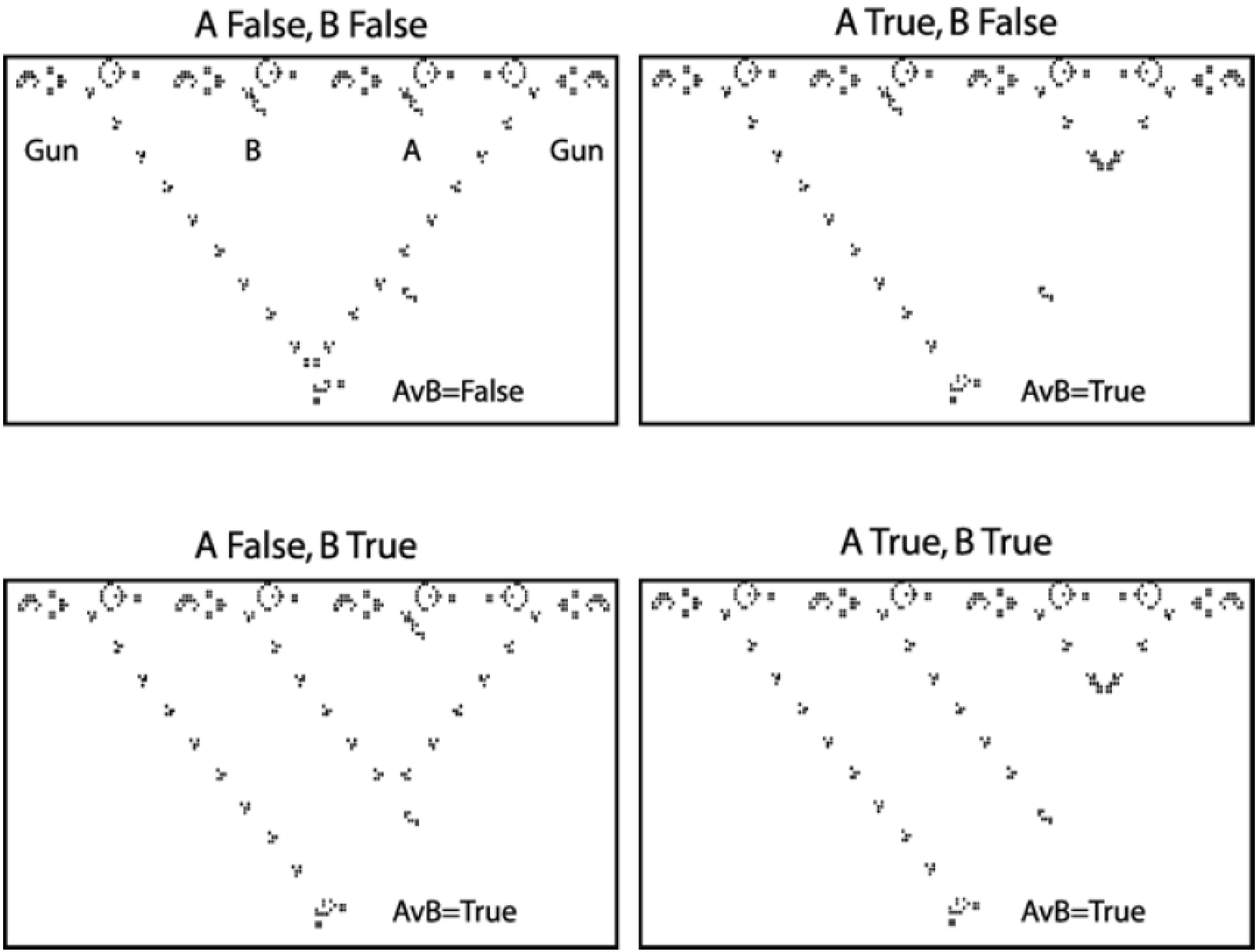


GoL “AND” gate



See file GOL_Gates.pdf in Canvas

GoL “OR” gate



See file GOL_Gates.pdf in Canvas

GoL is a Universal Computer!

- **Proof Outline:**
  - Have memory (stable/periodic blocks)
  - Have AND and NOT gates → NAND gate
  - NAND gates can build any Boolean function
  - Boolean functions provide finite state control in UTM
  - A UTM can run any other TM
  - All algorithms are TM's
  - GoL can run any algorithm → GoL is a universal computer!

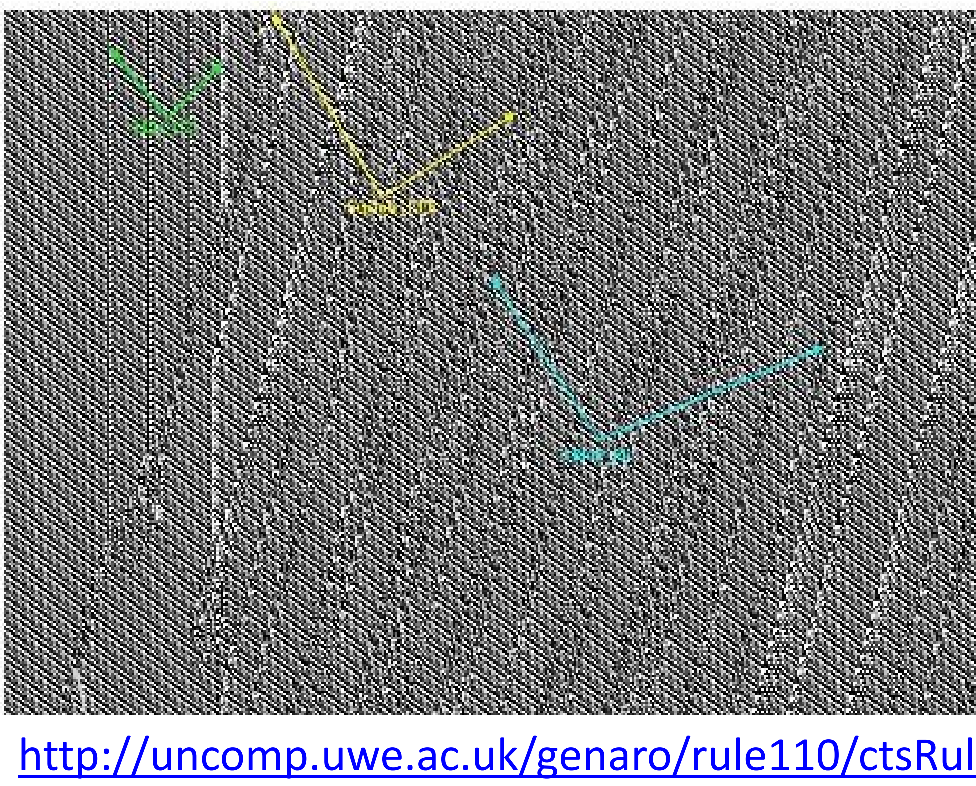
GoL is a Universal Computer!

- **This proof is abstract, but implementations have been done!**
  - Paul Rendell, 2001. "A Turing Machine in Conway's Game Life"
    - http://rendell-attic.org/gol/tm.htm
    - The implementation is huge.
  - Chapman, 2002, constructed a full blown Universal TM
    - http://www.igblan.free-online.co.uk/igblan/ca/
    - **268,096 initially active cells, 2,600 x 21,500 grid**

Computational Science

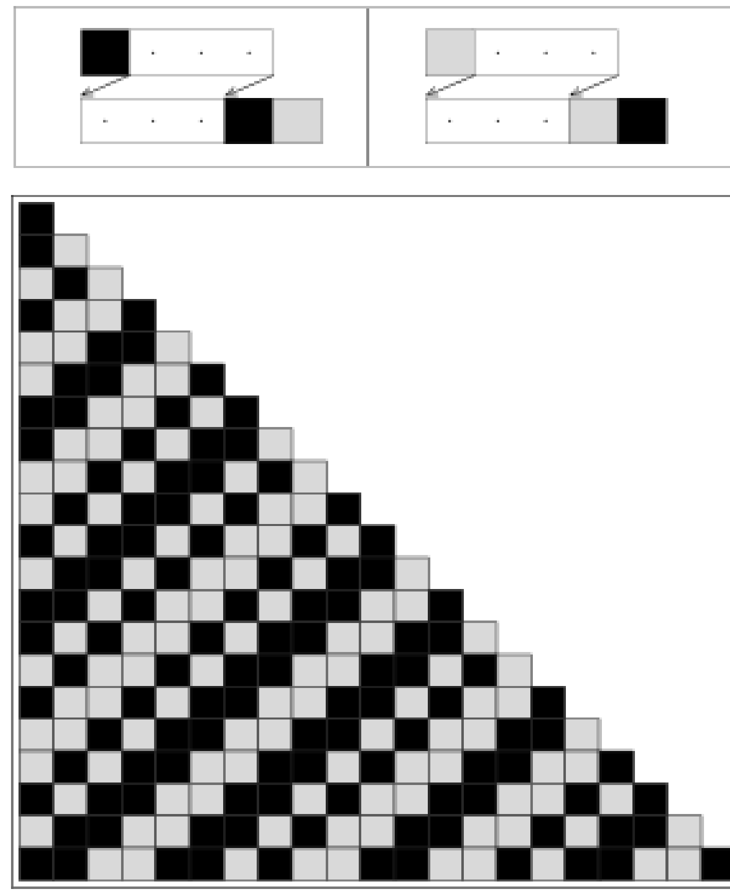
R110 is the simplest UTM

- **Proven by Matthew Cook in 1998/2000**
- **R110 emulates Cyclic tag system, which is UTM**



Tag system (not cyclic!)

$(1, \ldots) \to (\ldots, 1, 0) \quad (0, \ldots) \to (\ldots, 0, 1)$



adapted from Wolfram, S. *A New Kind of Science*. Wolfram Media, p. 93, 2002.

http://uncomp.uwe.ac.uk/genaro/rule110/ctsRule110.html

NEXT LECTURE: Grid-based models
NEXT LAB SESSION: CA3 Langton & entropy