A crash course in the theory of molecular computing

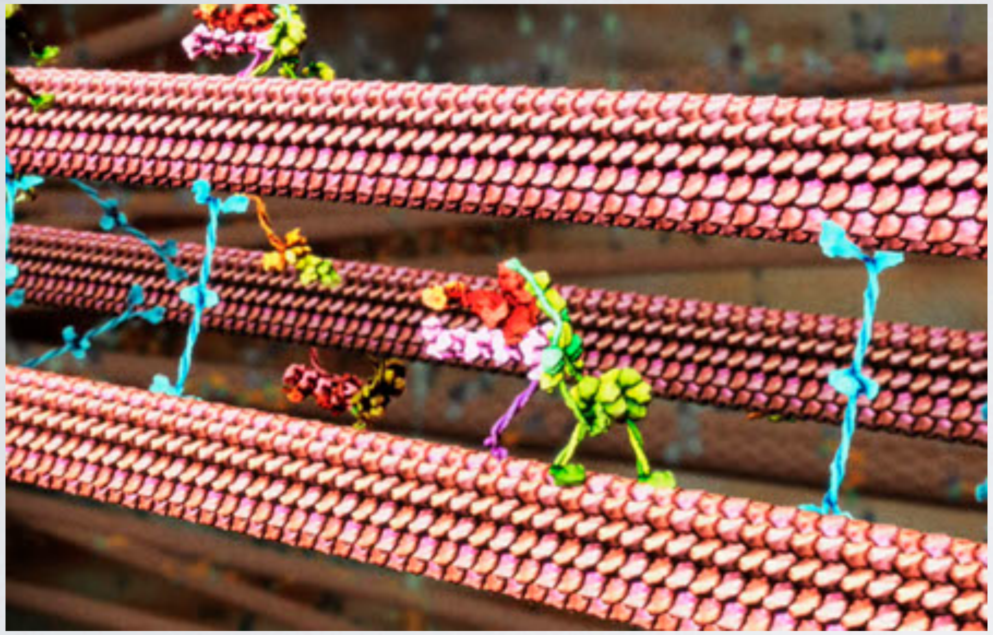
Damien Woods

Caltech

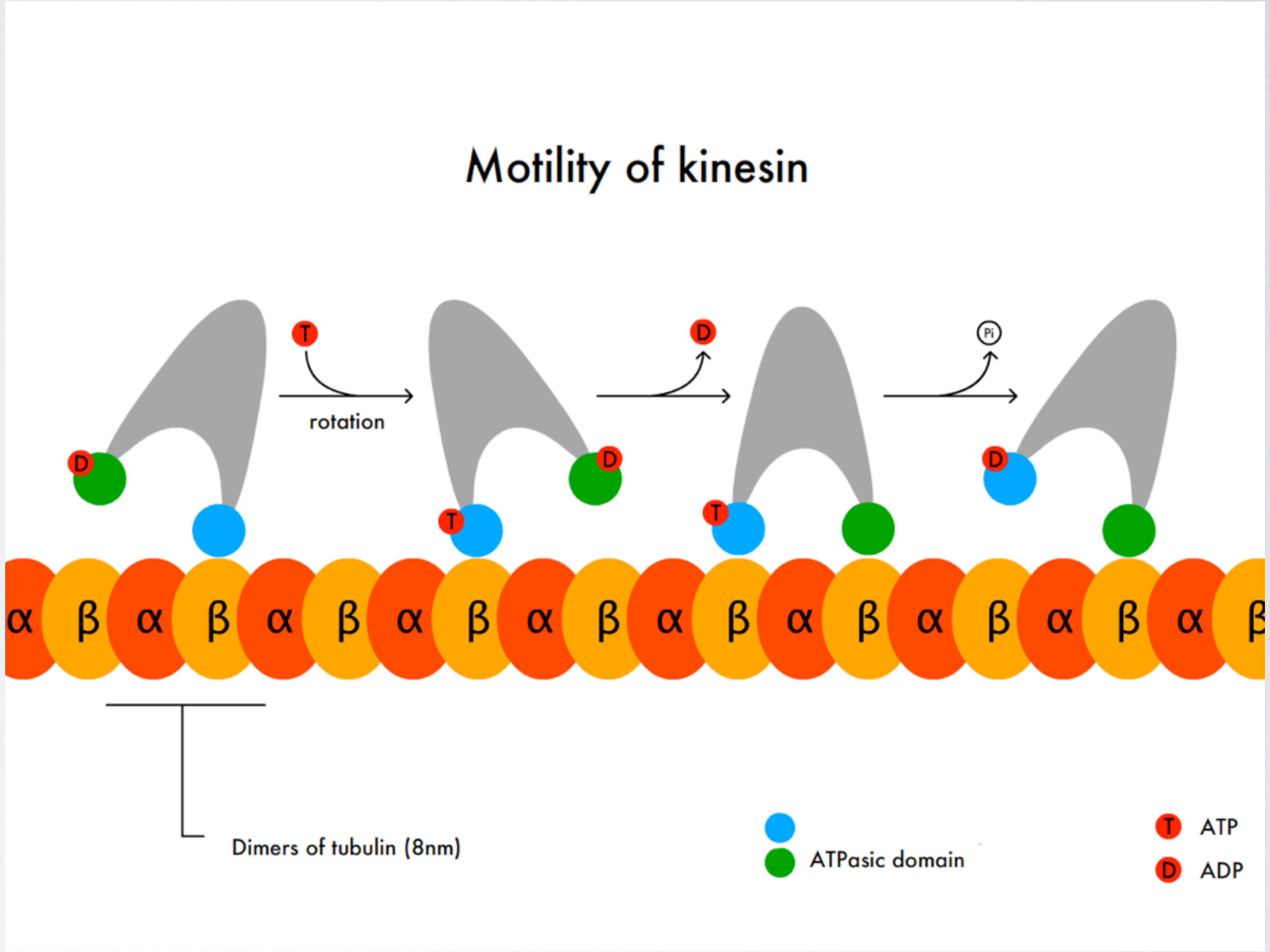
Overview

- Prediction
- Prediction and computation
- Computational universality
- Efficiency: sequential vs parallel computation
- Prediction

Prediction



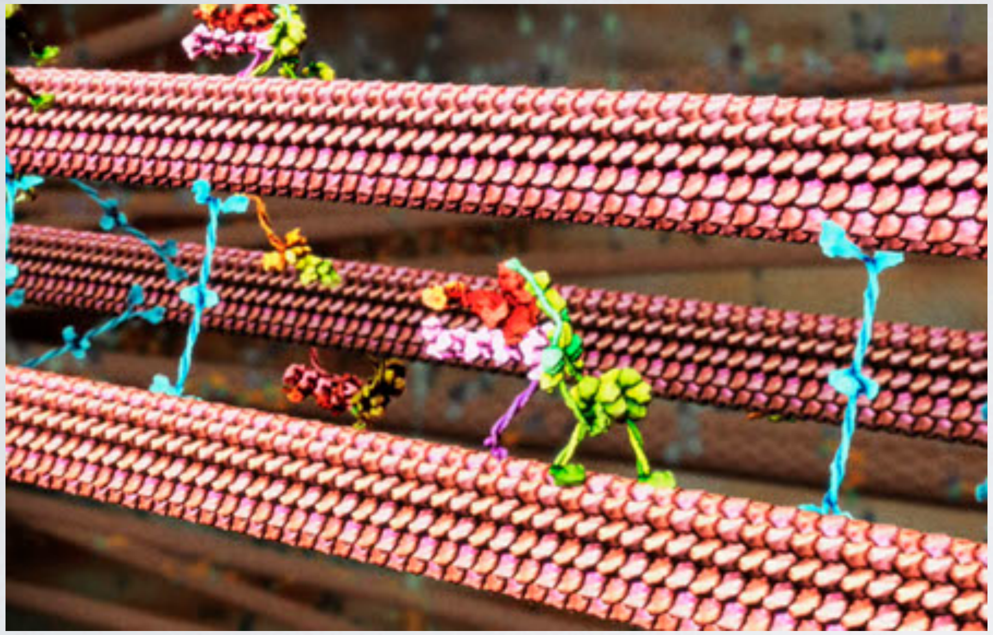
Drew Berry



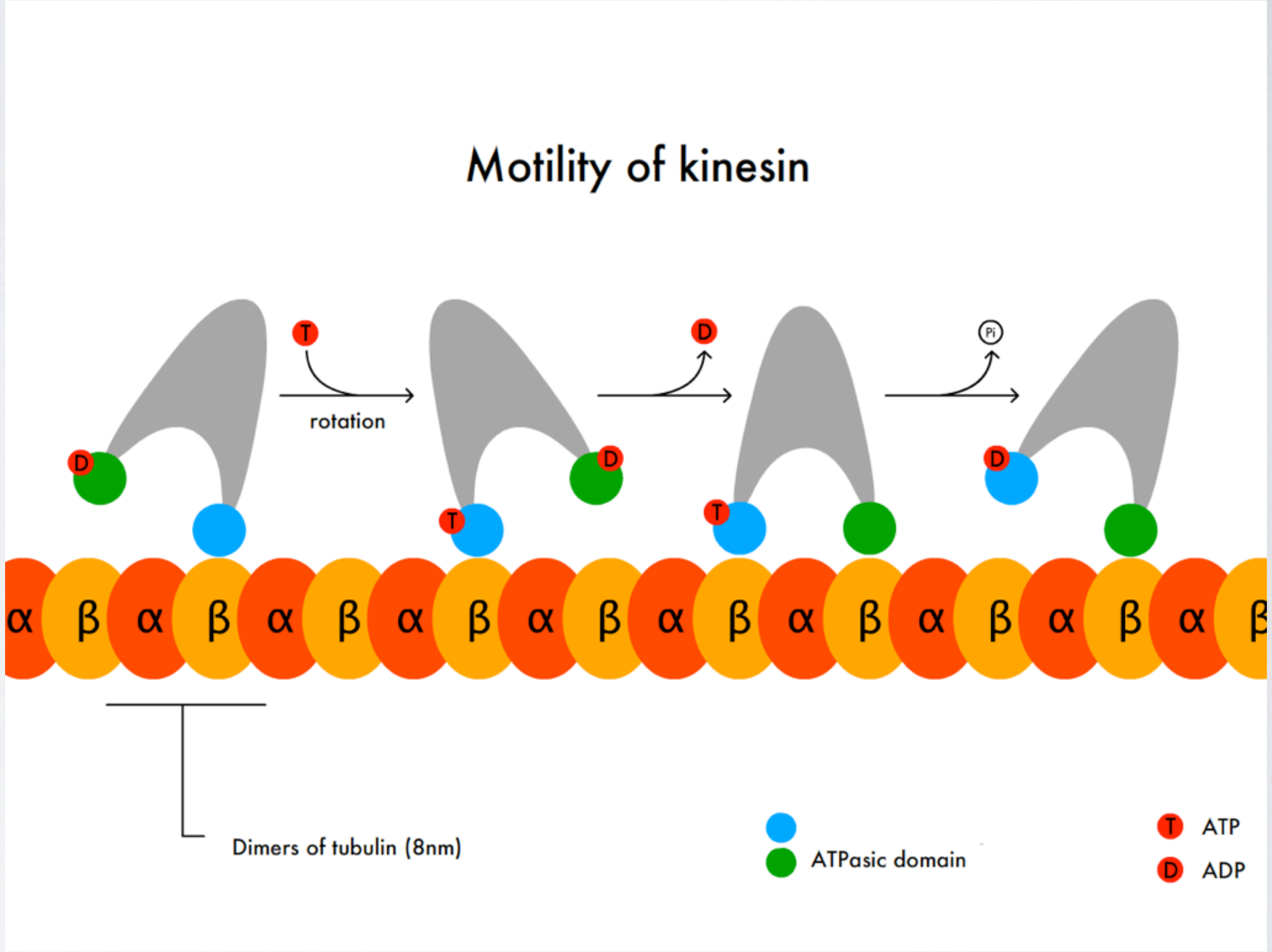
http://en.wikipedia.org/wiki/File:Motility_of_kinesin_en.png

- Kinesin: a molecular walker
- Step size of 8nm
- How long to walk a given distance?

Prediction



Drew Berry



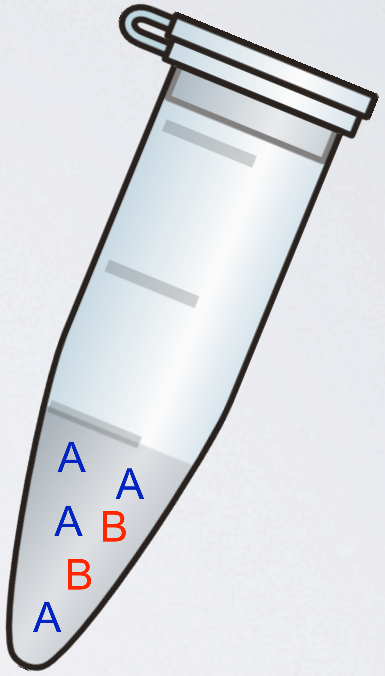
http://en.wikipedia.org/wiki/File:Motility_of_kinesin_en.png

- Kinesin: a molecular walker
- Step size of 8nm
- How long to walk a given distance?
- time = time_per_step x distance / step_size

Prediction

$$A \xrightarrow{1} B$$

- Exponential decay
- How many A's do we expect there to be at time t?

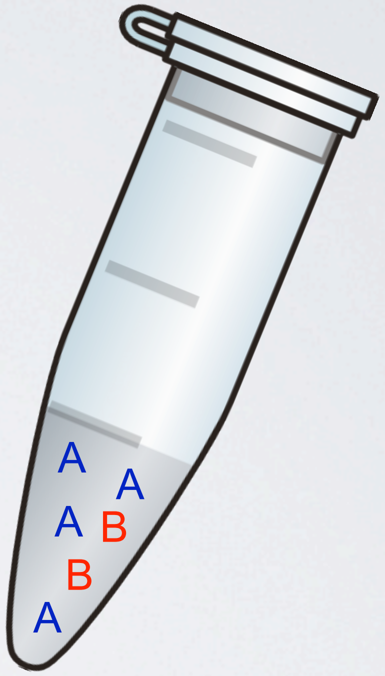


Prediction

$$A \xrightarrow{1} B$$

- Exponential decay
- How many A's do we expect there to be at time t?

$$\#A_t = \frac{\#A_{t-1}}{2}$$



Prediction

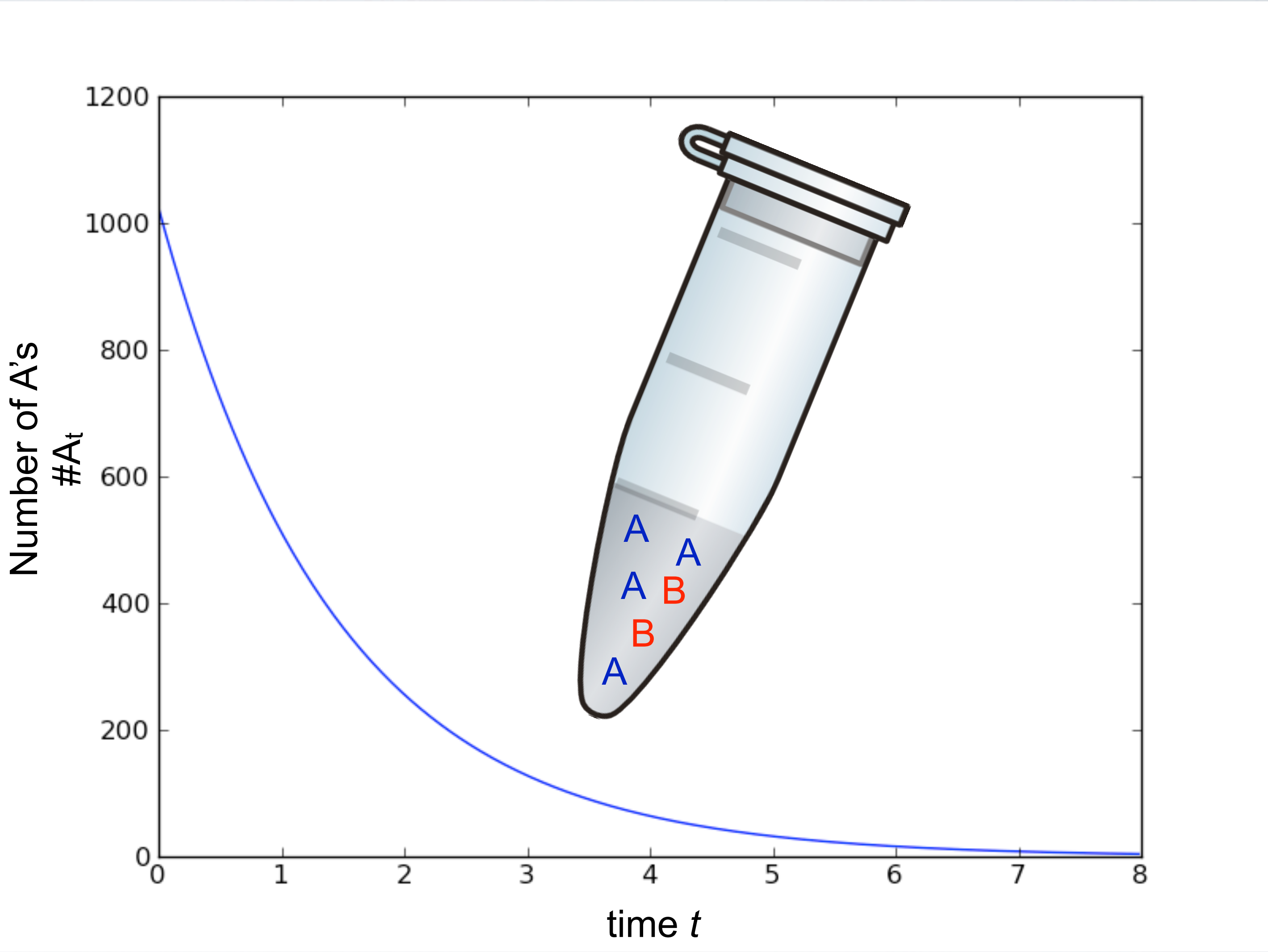
$$A \xrightarrow{1} B$$

- Exponential decay
- How many A's do we expect there to be at time t?

$$\#A_t = \frac{\#A_{t-1}}{2}$$

Or

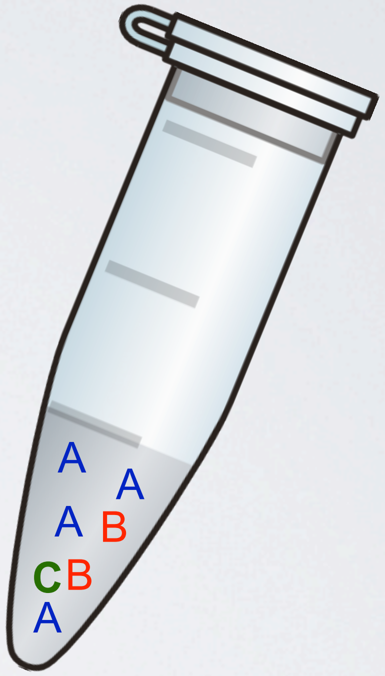
$$\#A_t = \#A_0 \frac{1}{2^t}$$



Prediction

$$A + C \xrightarrow{1} B + C$$

- Catalytic conversion of A's to B's
- One C, and many A's
- How many A's do we expect there to be at time t?



$$\#A_t = \#A_{t-1} - 1$$

Or

$$\#A_t = \#A_0 - t$$

Prediction

- A pair of linear maps

$$x_t = \begin{cases} x_{t-1}/2 & \text{if } x \text{ is even} \\ 3x_{t-1}+1 & \text{if } x \text{ is odd} \end{cases}$$

Prediction

- A pair of linear maps

$$x_t = \begin{cases} x_{t-1}/2 & \text{if } x \equiv 0 \mod 2 \\ 3x_{t-1}+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

$$g_{\mathrm{M}}(x) = \frac{a_i}{q}(x-i) + b_i \quad x \equiv i \mod q$$

?

Prediction

- A pair of linear maps

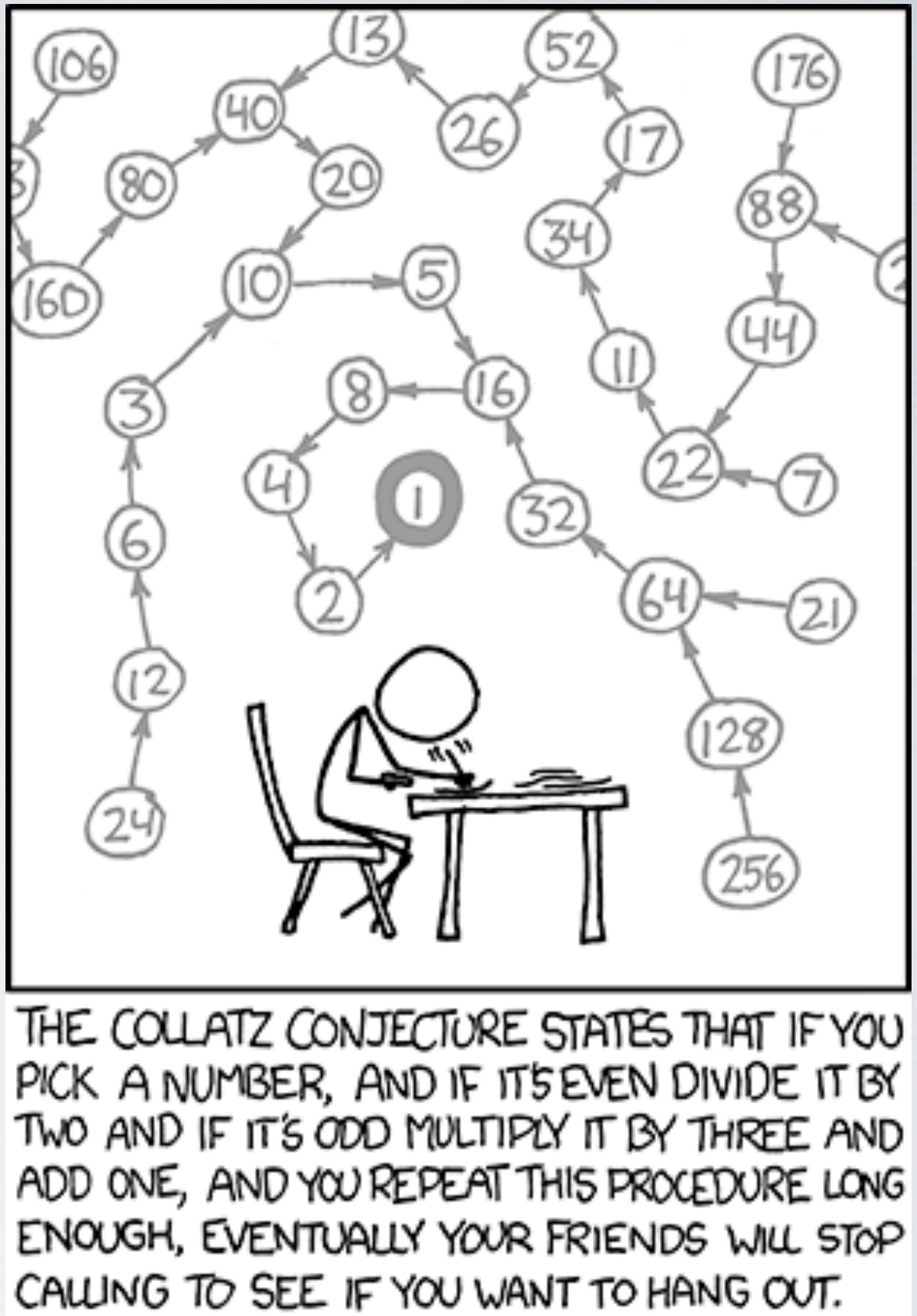
$$x_t = \begin{cases} x_{t-1}/2 & \text{if } x \equiv 0 \mod 2 \\ 3x_{t-1} + 1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

  - For all x_0, is there some t such that $x_t = 1$?

“Mathematics is not yet ready for such problems”

Paul Erdős

  - Maybe we’ll learn a lot by trying to solve it!



xkcd, #710

$$g_{\mathrm{M}}(x) = \frac{a_i}{q}(x - i) + b_i \quad x \equiv i \mod q$$

?

Predicting physical systems

- Even very simple-looking systems can carry out arbitrarily complicated computations

- $\Longrightarrow$ There are very simple-looking systems whose dynamics is so complicated that we **provably** have no simple formula to predict them

- For almost all of these systems we can not even hope to **simulate** any faster than full explicit (and slow!) simulation. The best we can do is just watch it evolve over time

- Even with a quantum computer, molecular computer, or any kind of highly parallel computer!

- The good news is that these systems are all **computers**

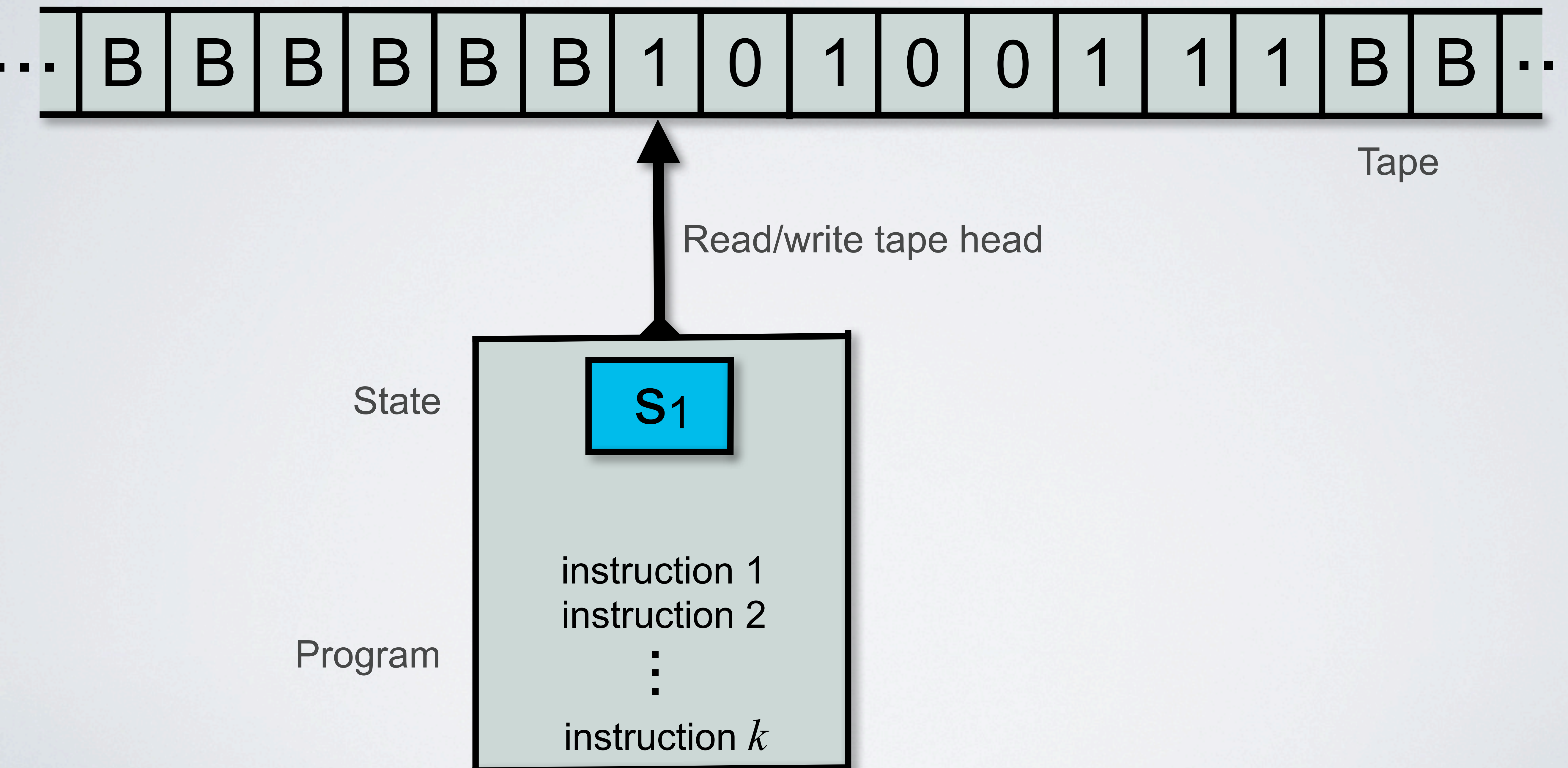
Computation

- Computation is all about dynamics
- In 1936, Turing wanted to define a general model of instruction-based dynamics

http://www.computerhistory.org/revolution/calculators/1/56/225

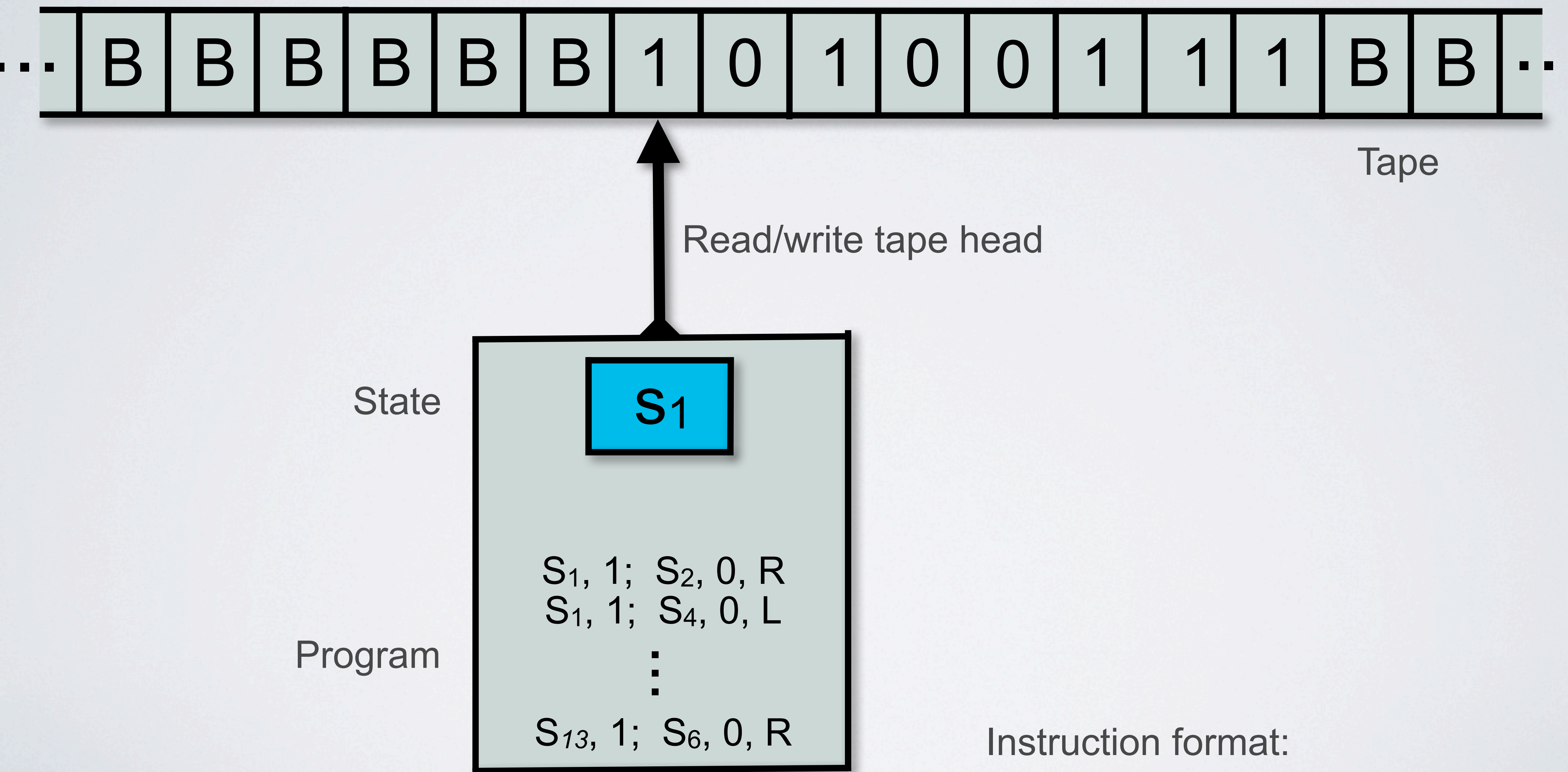
Turing machine

- A simple form of computer



Turing machine

- A simple form of computer

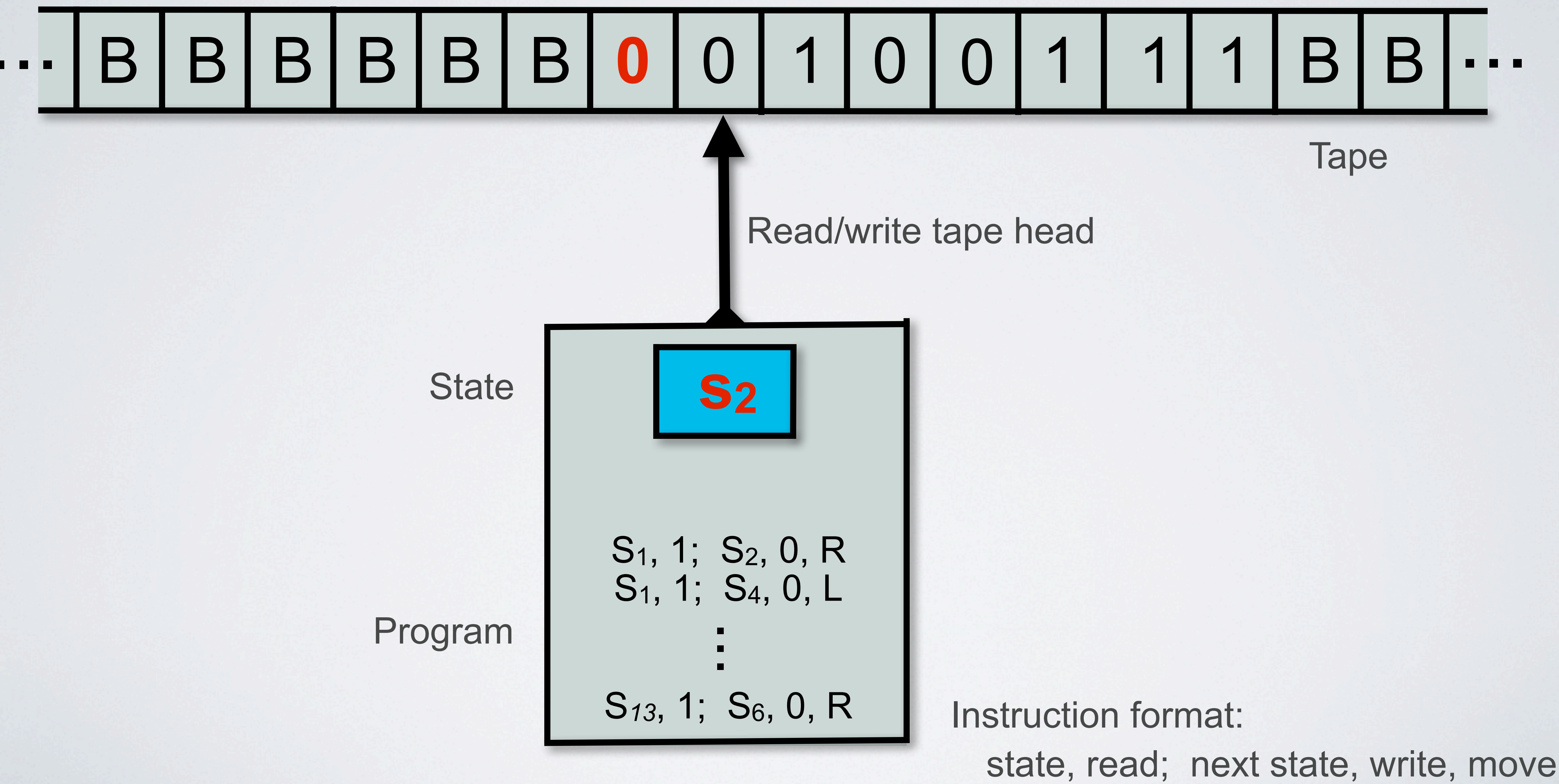


Instruction format:

state, read; next state, write, move

Turing machine

- A simple form of computer



Turing machine

- An implementation of a Turing machine

Turing machine

- An implementation of a Turing machine

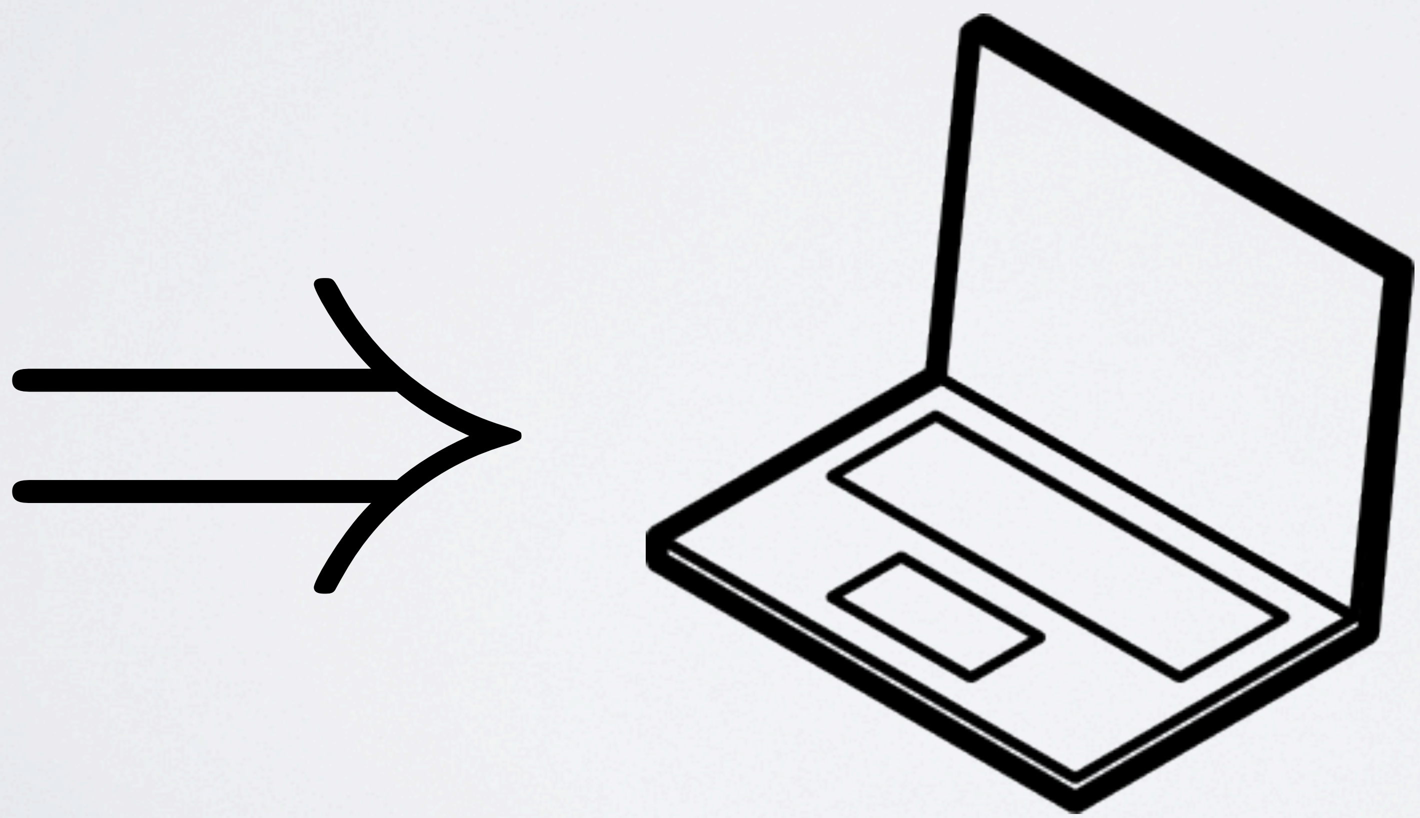
Turing machine

- An implementation of a Turing machine

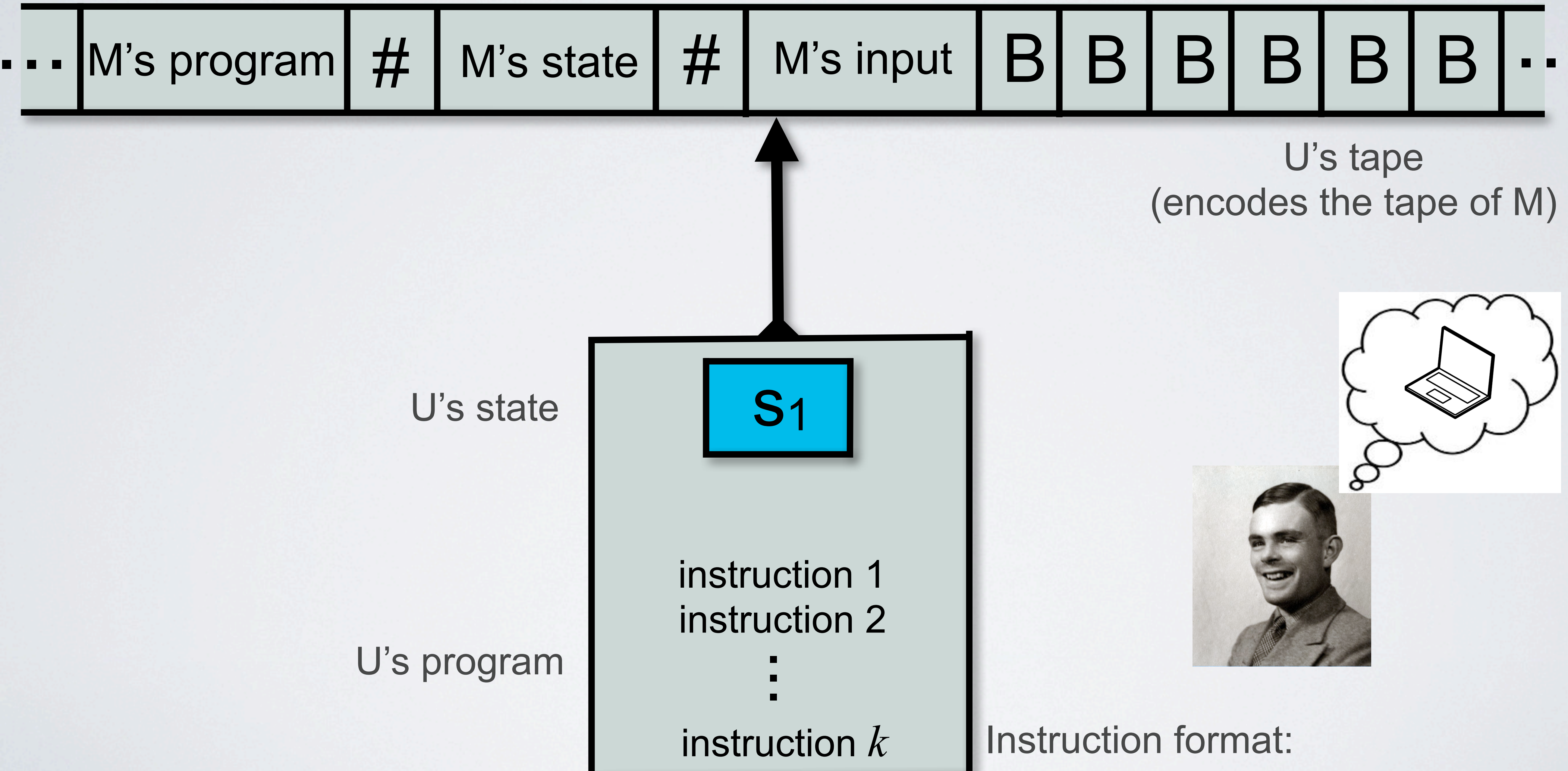
Anders Nissen, Martin Have, Mikkel Vester, Sean Geggie.
Computer Science, Aarhus University 2009.

Universal computation

- A mathematical idea that changed the world
- Turing showed that there is a Turing machine U that can simulate *any* other Turing machine



Universal Turing machine



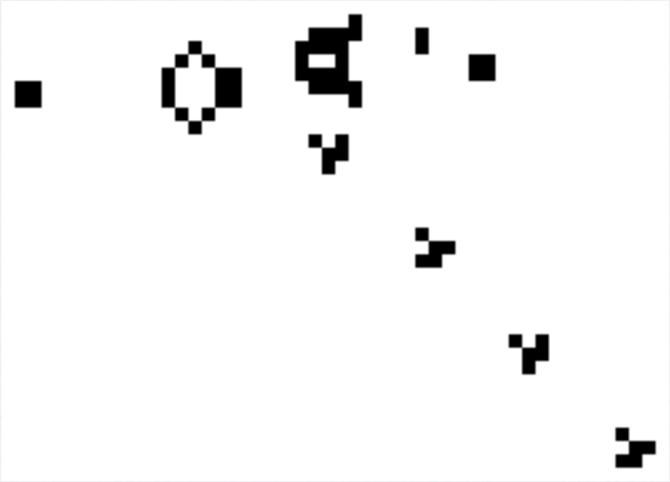
Instruction format:

state, read; next state, write, move

- Almost all questions about the long term dynamics of Turing machines are undecidable
- Universality uses the idea of simulation
- Lets use simulation to show that computation is ubiquitous
- There are *ridiculously* simple systems that are capable of universal computation!

Cellular automata

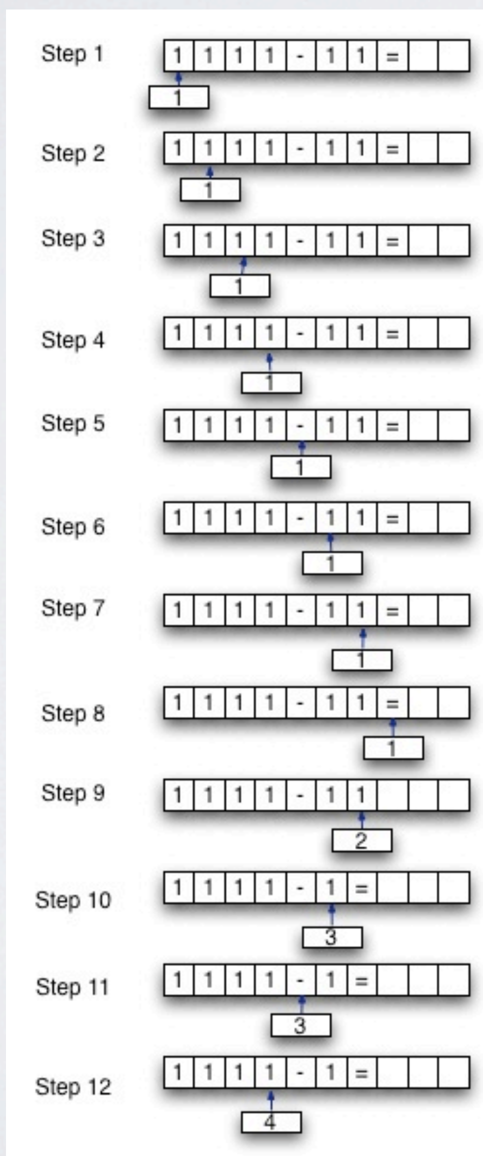
- Grid of cells, in 1 or more dimensions
- Each cell has one of a finite number of states
- Synchronous updates of states based on current state and that of neighbors



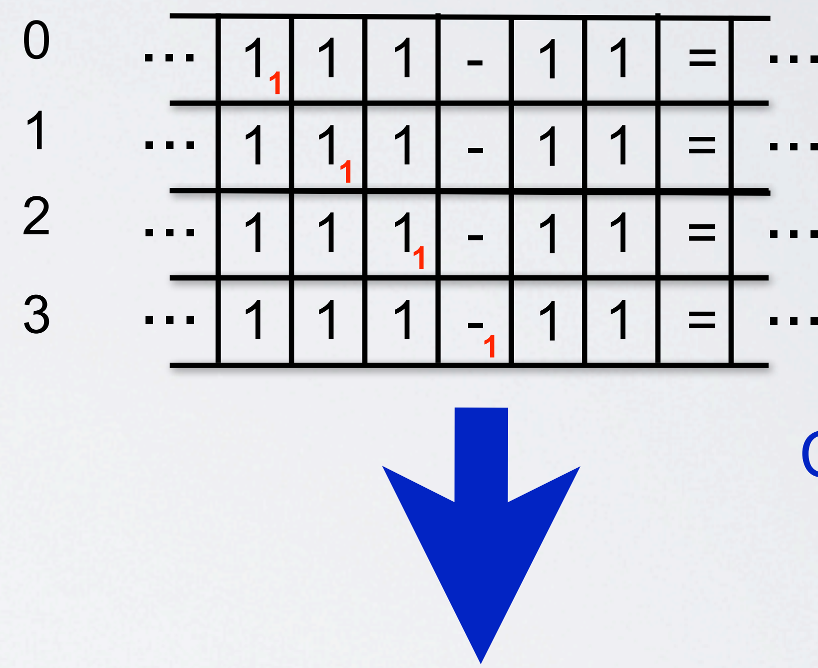
Cellular automata

- Direct simulation of Turing machines is easy (if we have enough states)

Turing machine
time/space history (tableau)



1D CA time evolution

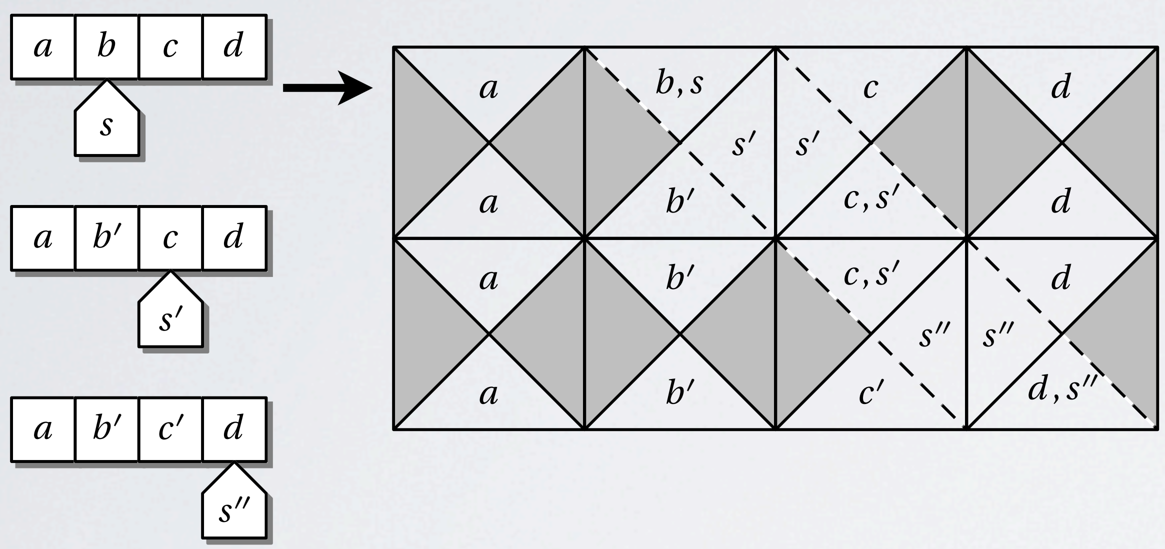


CA rules encode
TM instructions

Tiling

- Direct (tableau-style) simulations of Turing machines also show up in Wang tiling, DNA self-assembly, and Boolean circuits

TM step
0
1
2

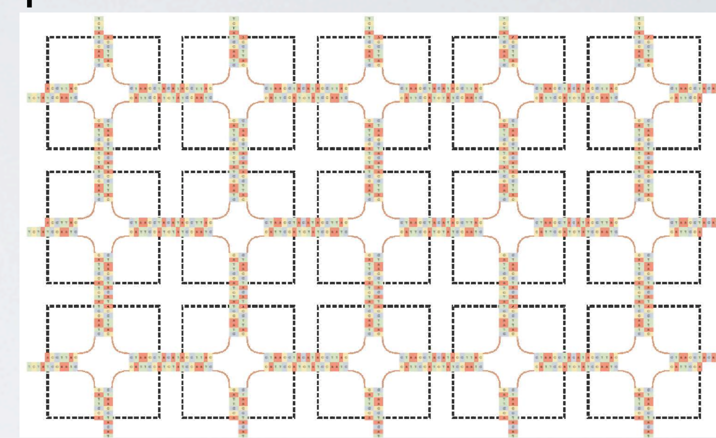


Turing machine

Wang tiles

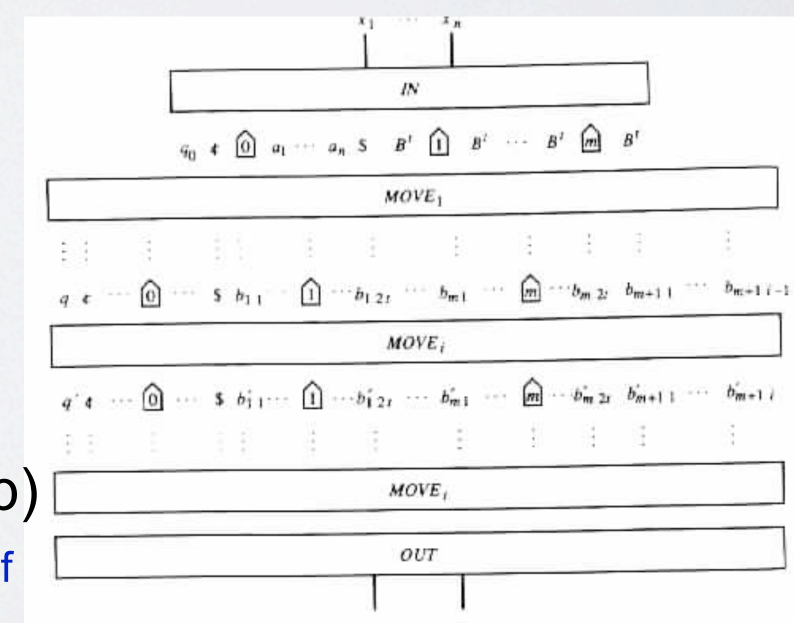
Pic credit: Moore & Mertens. The Nature of Computation. OUP 2012.

TM step
0
1
2



Self-assembled DNA tiles

Pic credit: Scott M Summers. Universality in algorithmic self-assembly. PhD thesis, ISU 2010



Boolean circuits (one layer per TM step)

Pic credit: E. Gurari. An Introduction to the Theory of Computation. Comp Sci Press. 1989

Cellular automata

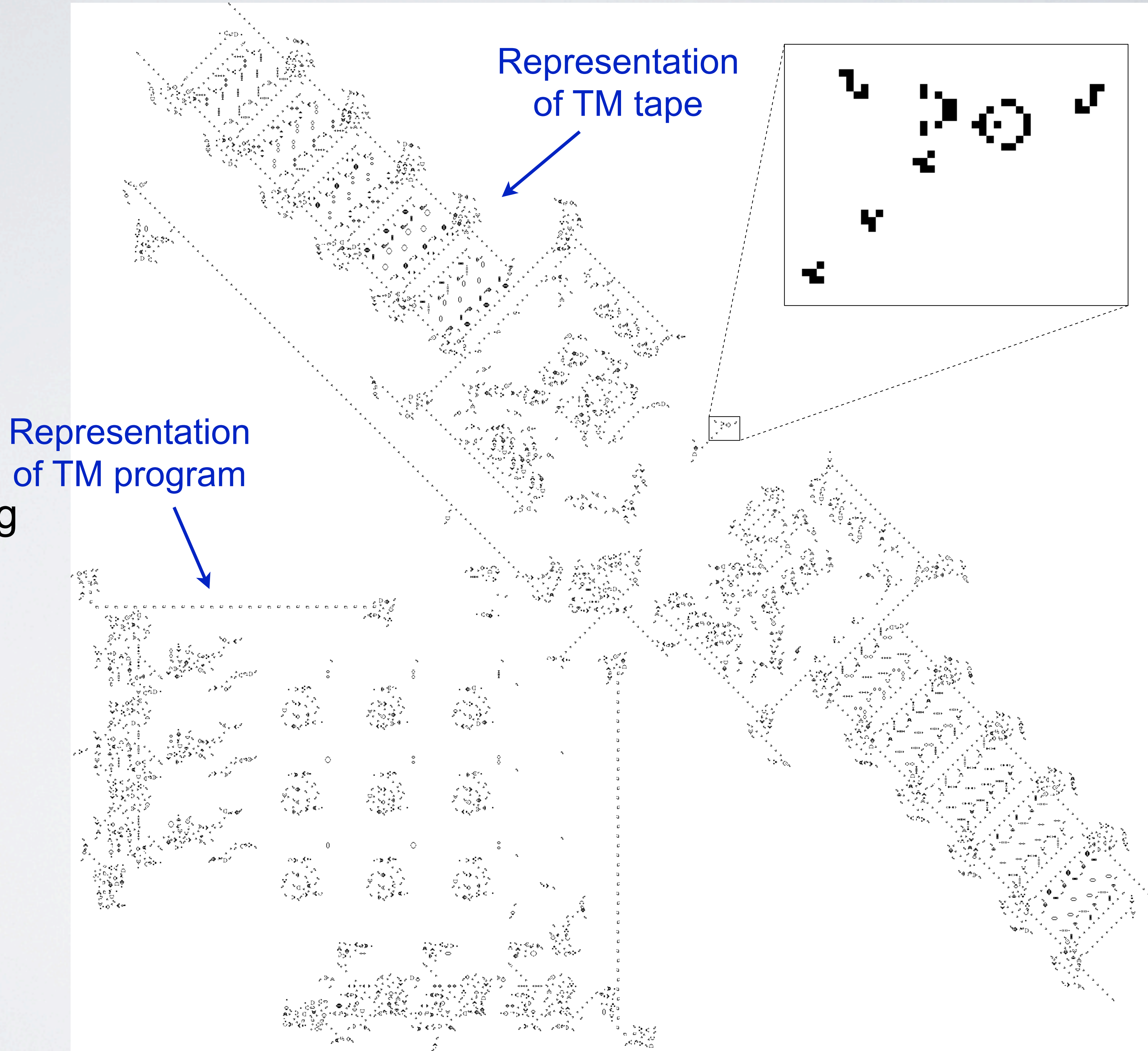
- Game of life
  1. Any live cell with fewer than two live neighbours dies.
  2. Any live cell with two or three live neighbours lives on to the next generation.
  3. Any live cell with more than three live neighbours dies
  4. Any dead cell with exactly three live neighbours becomes a live cell

  John Conway, 1970s

- Simulation of a Turing machine

- So 2D, 2 state CA, with a small neighborhood, are universal!

- What about in 1D?



Rule 110

- A 2-state 1D cellular automaton

$0 = \square$
$1 = \blacksquare$

time

Initial configuration

0

Rule 110

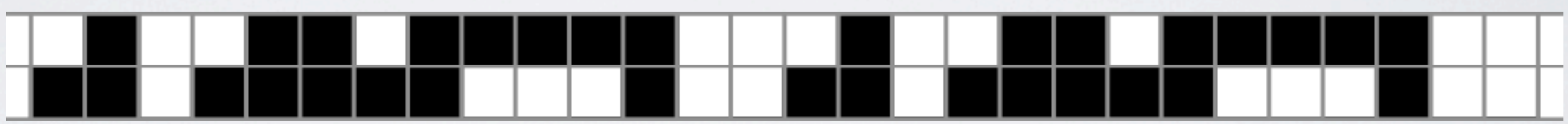
- A 2-state 1D cellular automaton

$0 = \square$

$1 = \blacksquare$

time

Initial configuration



0

1

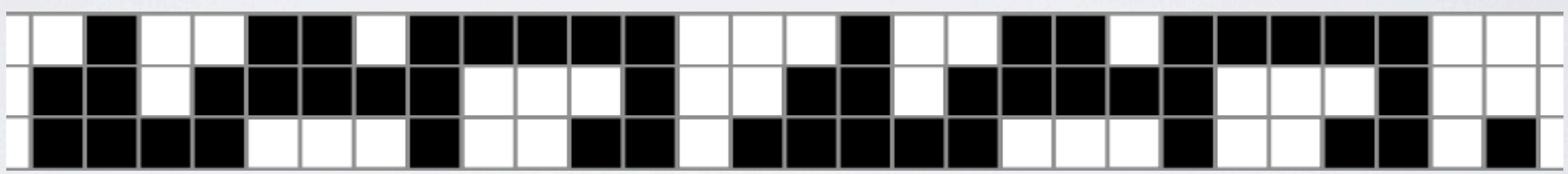
Rule 110

- A 2-state 1D cellular automaton

$0 = \square$
$1 = \blacksquare$

time

Initial configuration



0
1
2

Rule 110

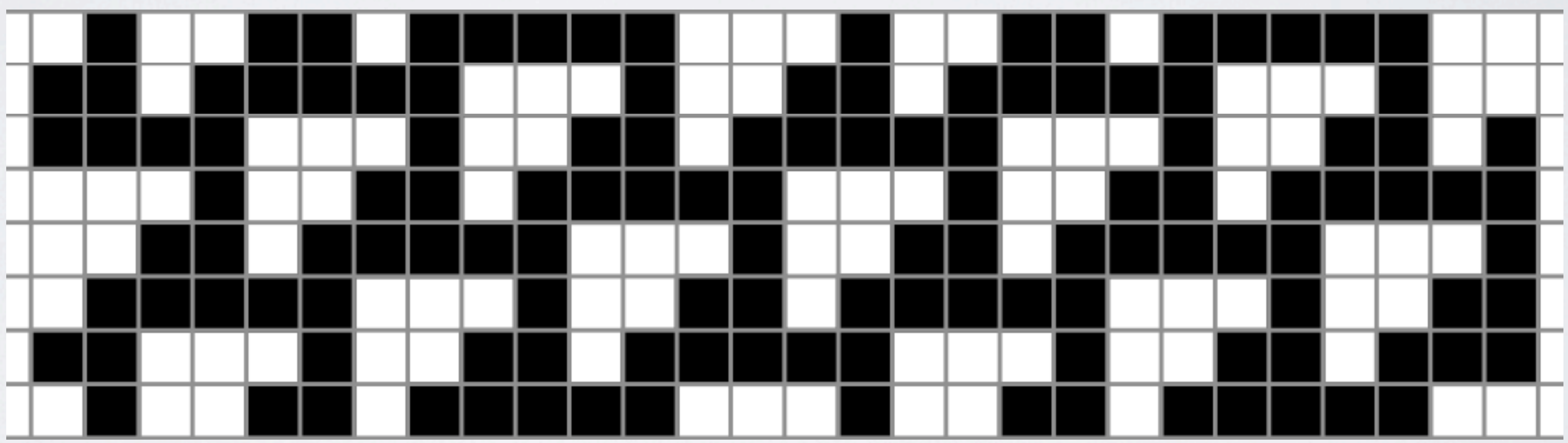
- A 2-state 1D cellular automaton

$0 = \square$

$1 = \blacksquare$

time

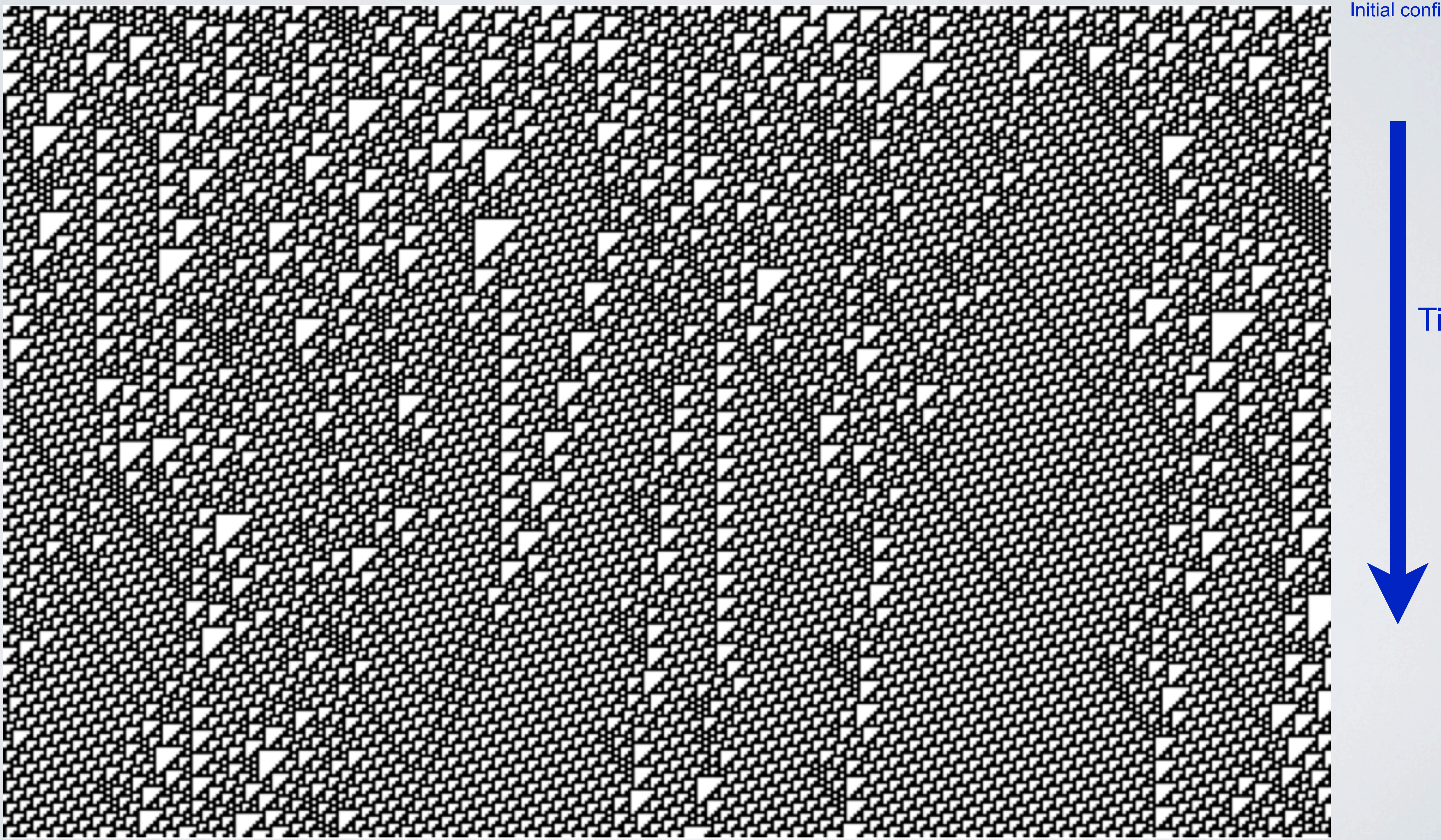
Initial configuration



0
1
2
3
4
5
6
7
⋮

Rule 110

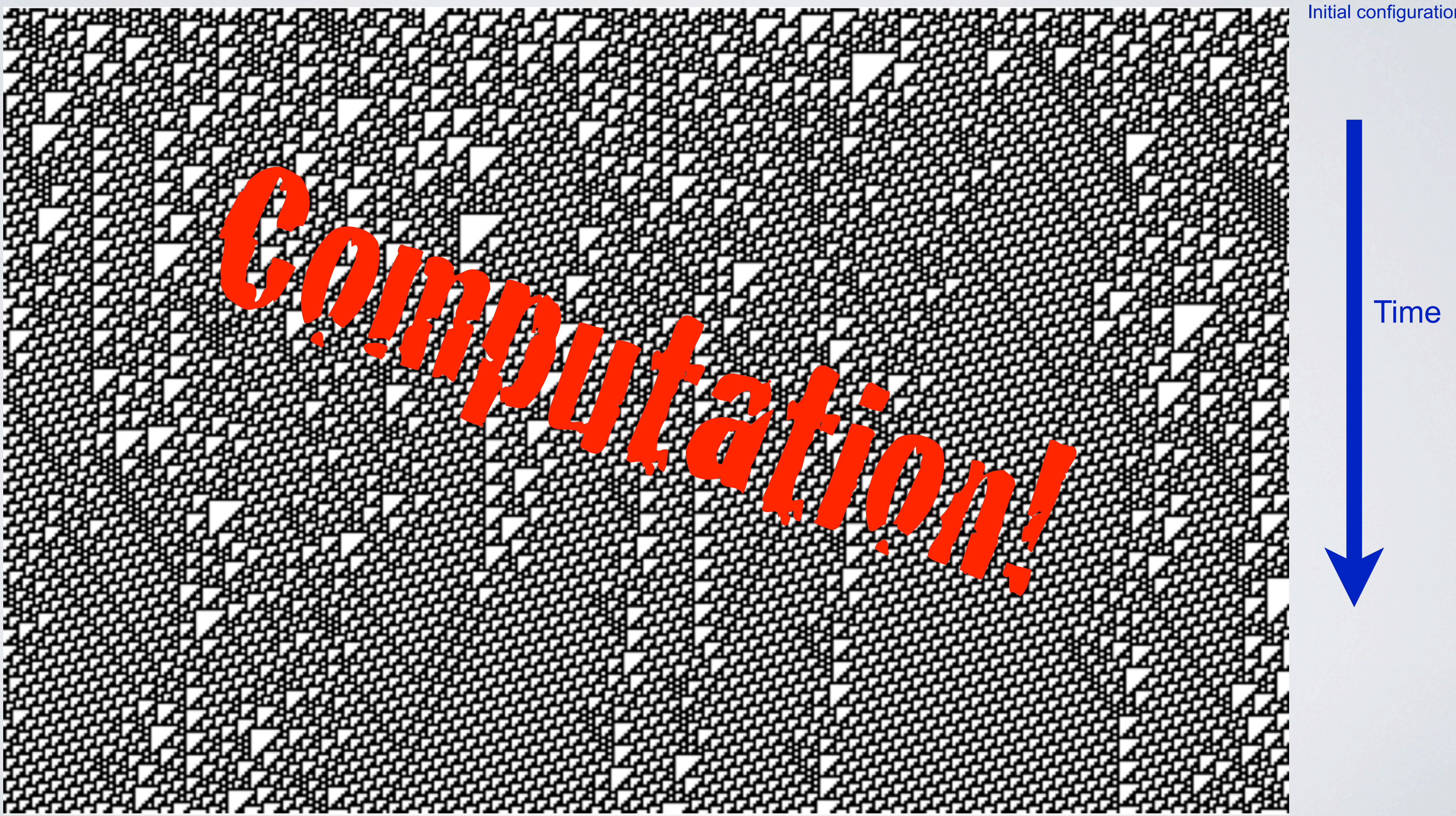
- What is going on here?



Cook, Matthew. Universality in Elementary Cellular Automata. Complex systems (2004) 15(1):1–40

Rule 110

- What is going on here?

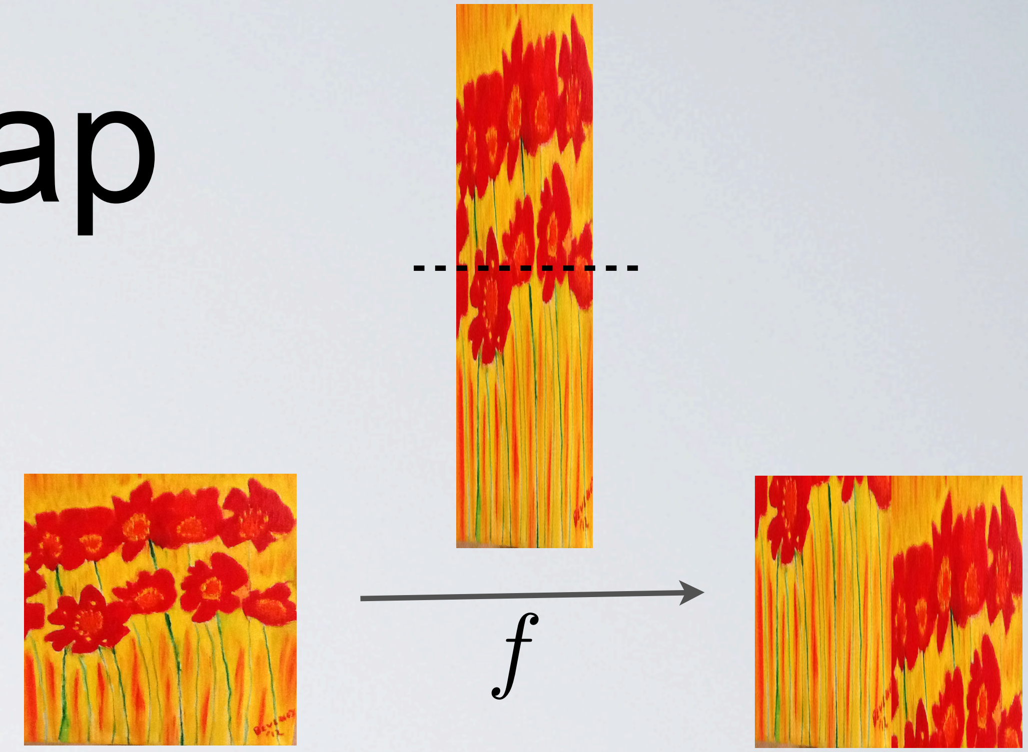


Cook, Matthew. Universality in Elementary Cellular Automata. Complex systems (2004) 15(1):1–40

Universality and simulation: by the numbers

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \geq 1/2 \end{cases}$$

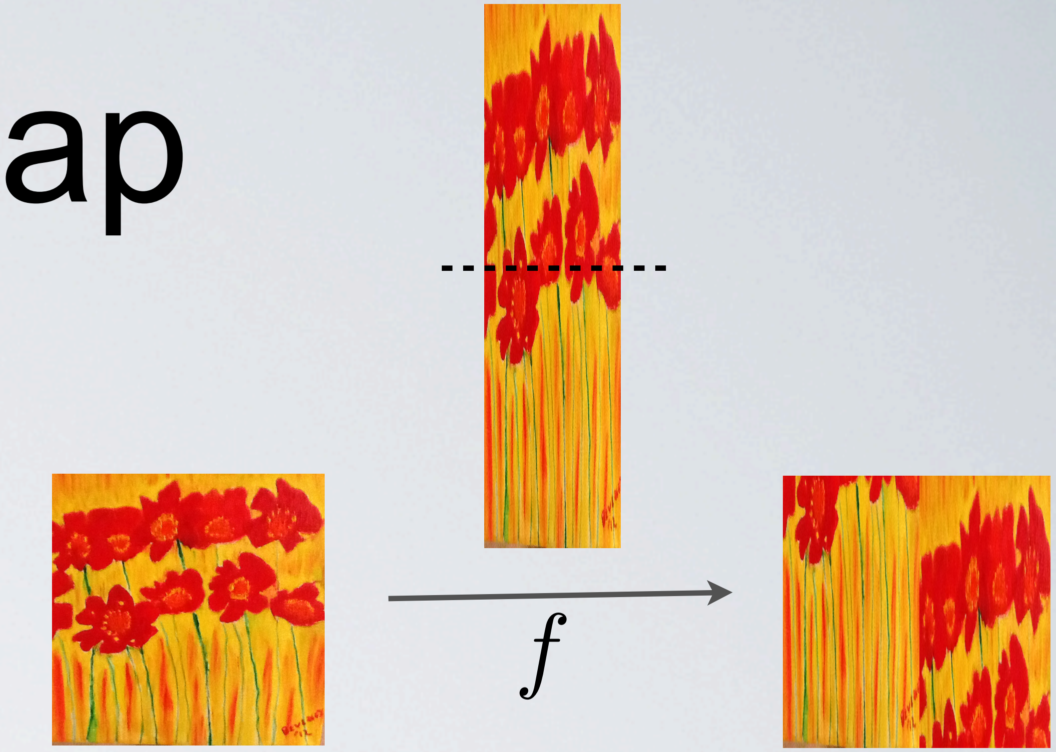


Pic: Beverley Henley

- A simple example of a chaotic dynamical system on the unit square

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \geq 1/2 \end{cases}$$

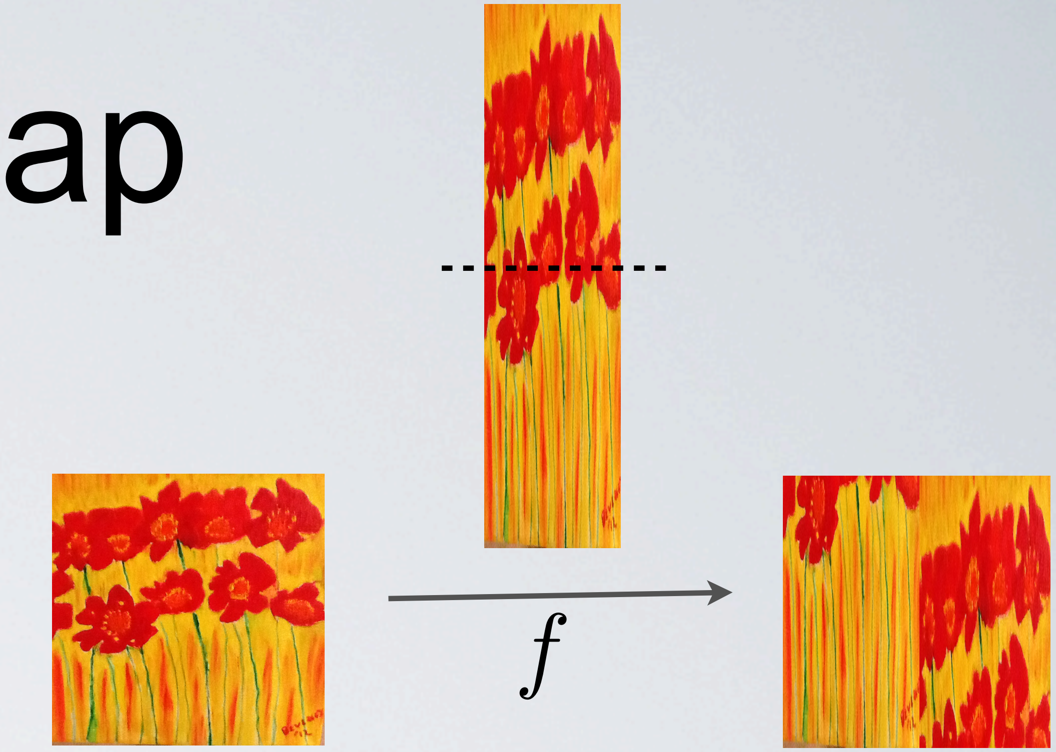


Pic: Beverley Henley

- A simple example of a chaotic dynamical system on the unit square

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \geq 1/2 \end{cases}$$

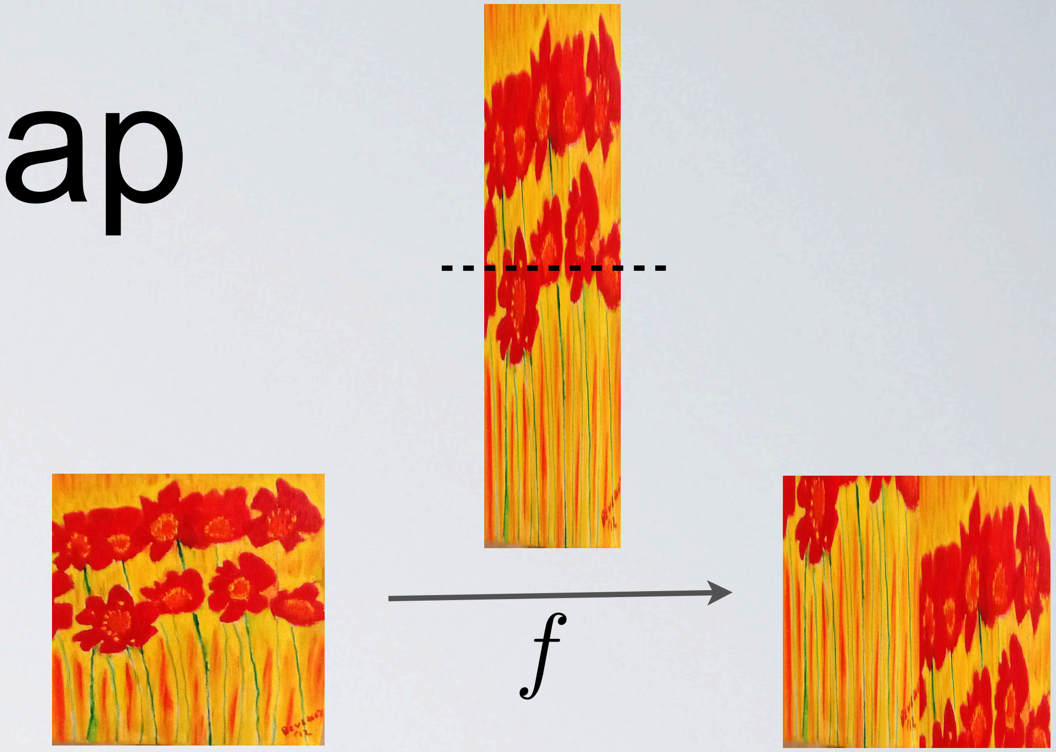


Pic: Beverley Henley

- A simple example of a chaotic dynamical system on the unit square
- Example values *x* and *y*: $x = 1/2 + 1/4 + 1/8$ $y = 1/4 + 1/16$

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y-1) & \text{if } y \geq 1/2 \end{cases}$$

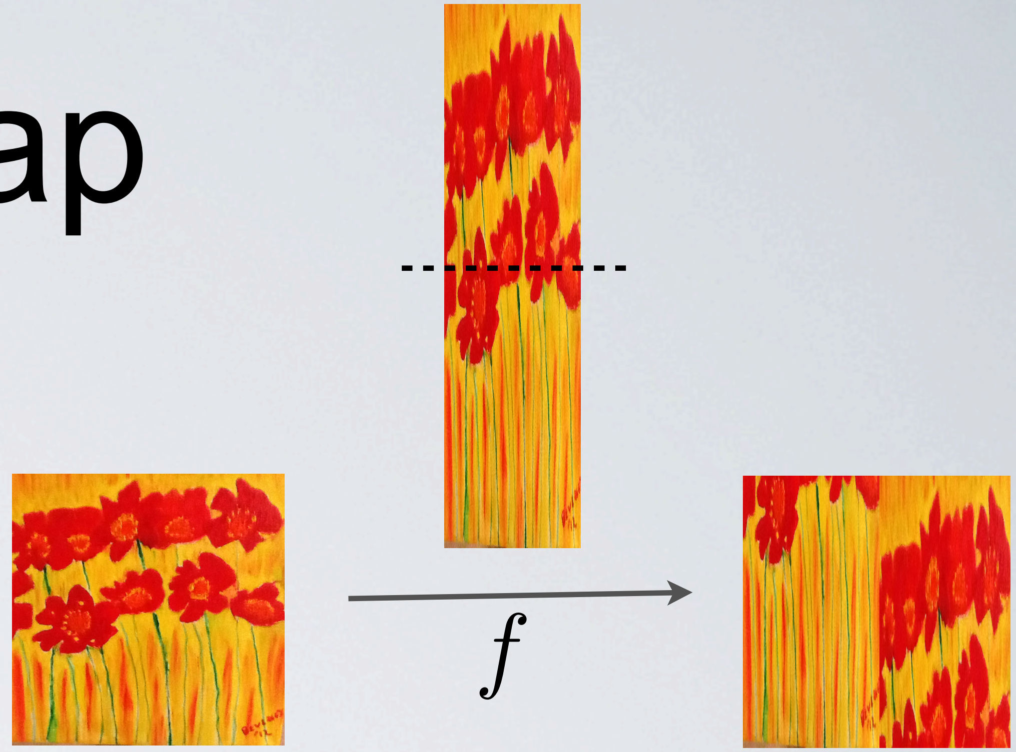


Pic: Beverley Henley

- A simple example of a chaotic dynamical system on the unit square
- Example values *x* and *y*: $x = 1/2 + 1/4 + 1/8$ $y = 1/4 + 1/16$
- Now write *x* and *y* in binary: $x = 0.111$ $y = 0.0101$

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \geq 1/2 \end{cases}$$

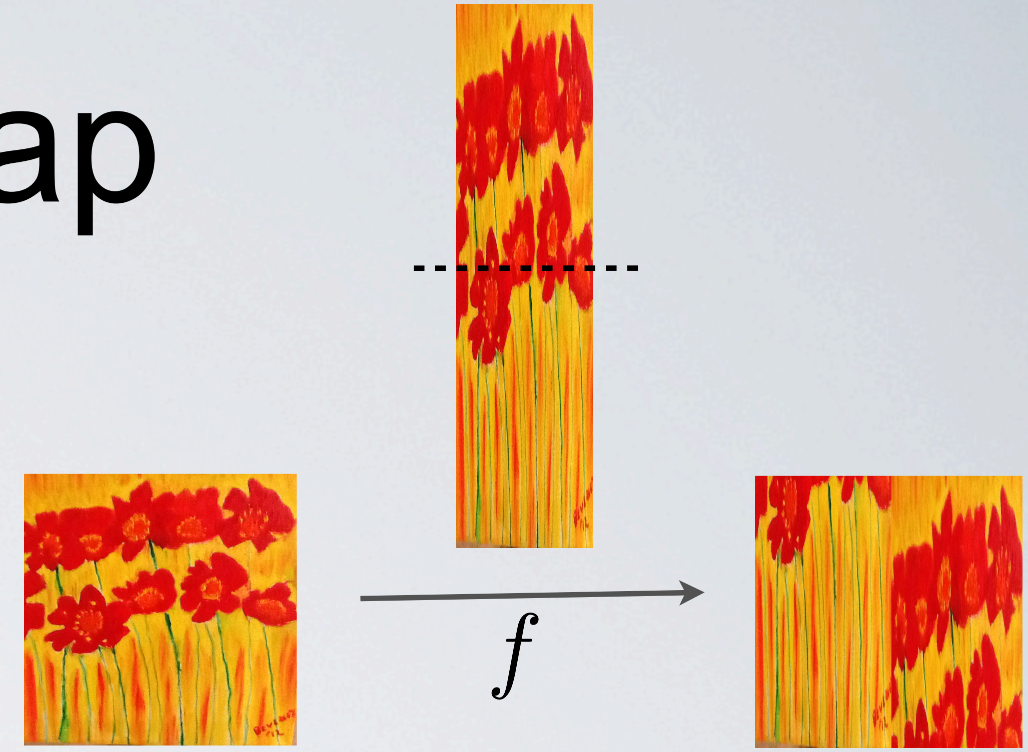


Pic: Beverley Henley

- A simple example of a chaotic dynamical system on the unit square
- Example values *x* and *y*: $x = 1/2 + 1/4 + 1/8$ $y = 1/4 + 1/16$
- Now write *x* and *y* in binary: $x = 0.111$ $y = 0.0101$
- Mirror *x:* $x = 111.0$ $y = 0.0101$

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \geq 1/2 \end{cases}$$



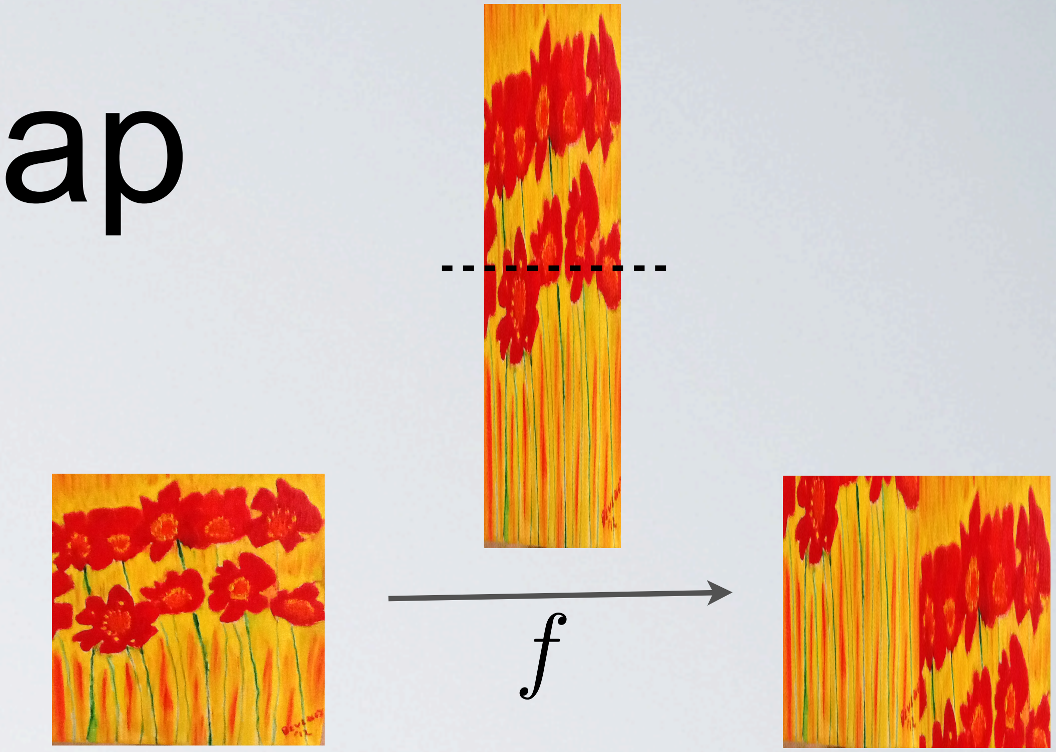
Pic: Beverley Henley

- A simple example of a chaotic dynamical system on the unit square
- Example values *x* and *y*: $x = 1/2 + 1/4 + 1/8$ $y = 1/4 + 1/16$
- Now write *x* and *y* in binary: $x = 0.111$ $y = 0.0101$
- Mirror *x:* $x = 111.0$ $y = 0.0101$
- Write as a bi-infinite sequence: ...000111.0101000...

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y-1) & \text{if } y \geq 1/2 \end{cases}$$

test reads MSB of y

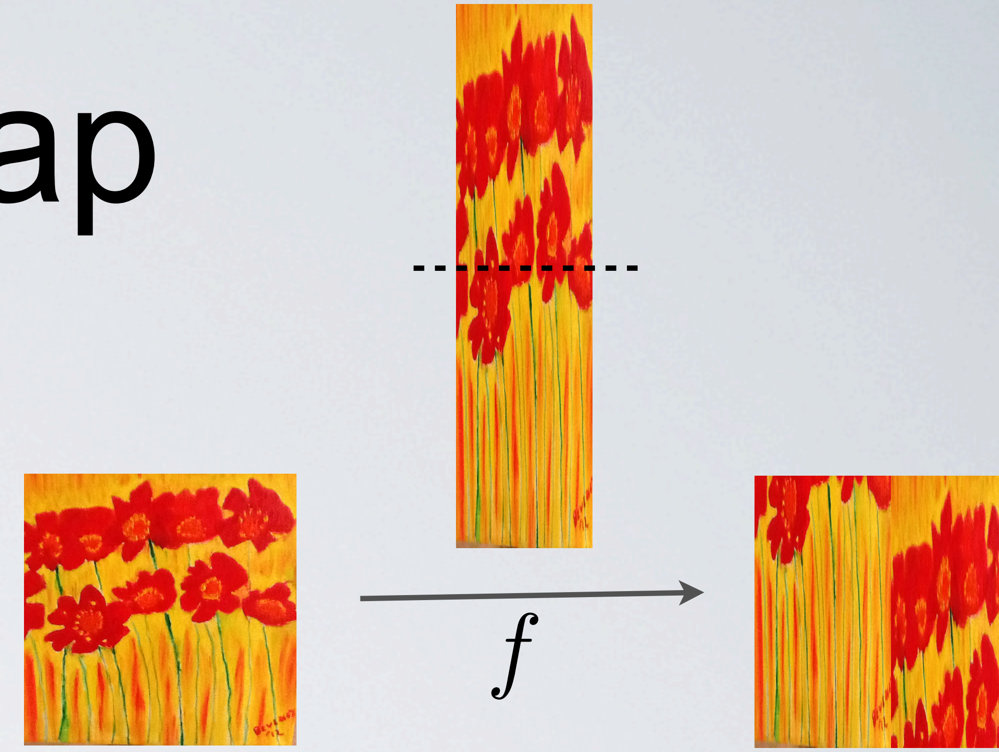


Pic: Beverley Henley

- A simple example of a chaotic dynamical system on the unit square
- Example values *x* and *y*: $x = 1/2 + 1/4 + 1/8$ $\quad y = 1/4 + 1/16$
- Now write *x* and *y* in binary: $x = 0.111$ $\quad y = 0.0101$
- Mirror *x:* $x = 111.0$ $\quad y = 0.0101$
- Write as a bi-infinite sequence: ...000111.0101000...
- Iterate *f* : ...0001110.101000...

Baker's map

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \geq 1/2 \end{cases}$$



Pic: Beverley Henley

test reads MSB of y

- A simple example of a chaotic dynamical system on the unit square
- Example values *x* and *y*: $x = 1/2 + 1/4 + 1/8$ $\quad y = 1/4 + 1/16$
- Now write *x* and *y* in binary: $x = 0.111$ $\quad y = 0.0101$
- Mirror *x:* $x = 111.0$ $\quad y = 0.0101$
- Write as a bi-infinite sequence: ...000111.0101000...
- Iterate *f* :

...0001110.101000...

...00011101.01000...

...000111010.1000...

...0001110101.000...

- Test on *y* reads most significant bit
- Moore saw that this map simulates right shift of a TM tape head

Moore. Unpredictability and undecidability in dynamical systems. PRL. 1990

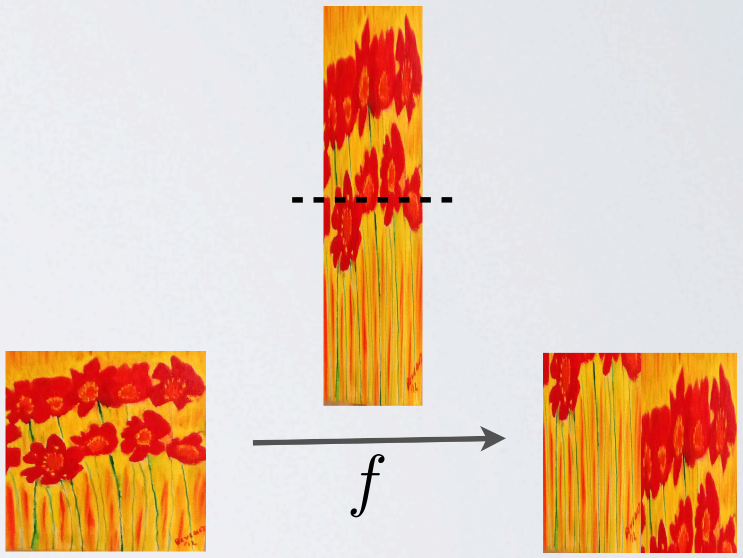
Generalized shift

deleting/writing a bit

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y-1) & \text{if } y \geq 1/2 \end{cases}$$

...000111.0101000...

test reads MSB of y



Generalized shift

deleting/writing many bits

$$f(x, y) = (a_i x + b_i, c_i y + d_i) \quad \text{if } e_i \le y < h_i$$

TM state TM head

...000111.110110101000...

TM tape

test read MSBs of y: i.e., TM state and read symbol

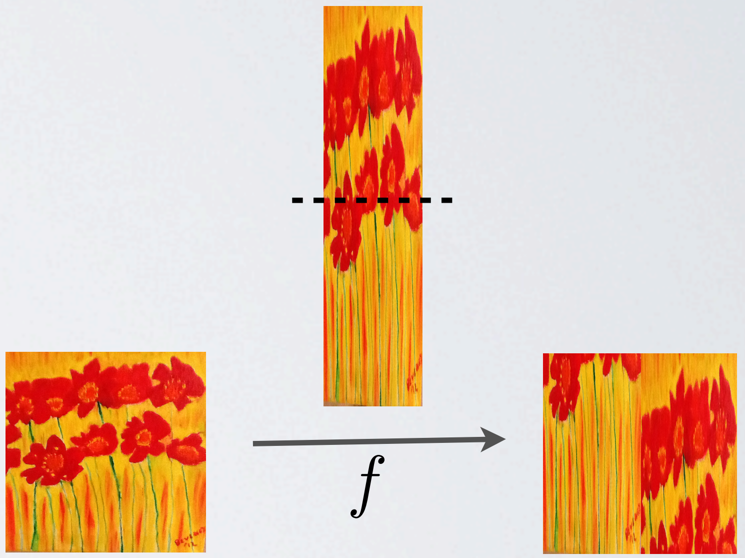
Moore. Unpredictability and undecidability in dynamical systems. PRL. 1990

deleting/writing a bit

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \ge 1/2 \end{cases}$$

...000111.0101000...

test reads MSB of y



Generalized shift

deleting/writing many bits

$$f(x, y) = (a_i x + b_i, c_i y + d_i) \quad \text{if } e_i \leq y < h_i$$

deleting/writing a bit

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y - 1) & \text{if } y \geq 1/2 \end{cases}$$

TM state TM head

...000111.110110101000...

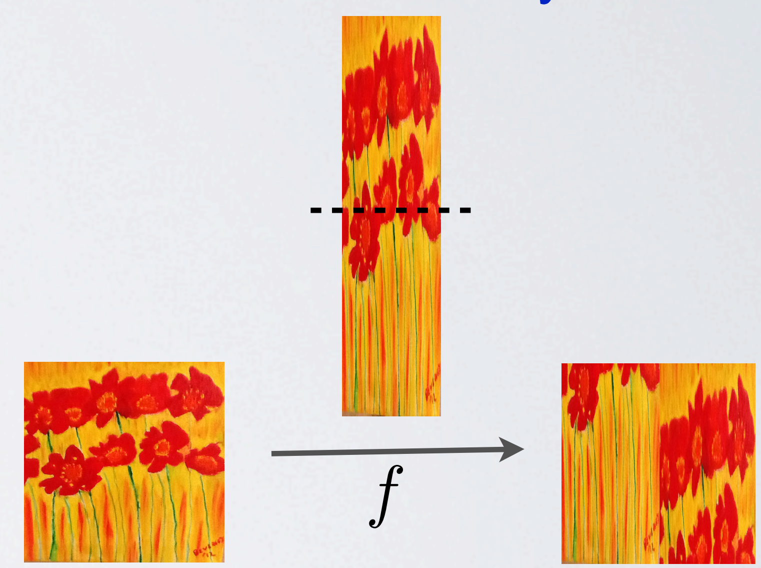
TM tape

test read MSBs of y: i.e., TM state and read symbol

...000111.0101000...

test reads MSB of y

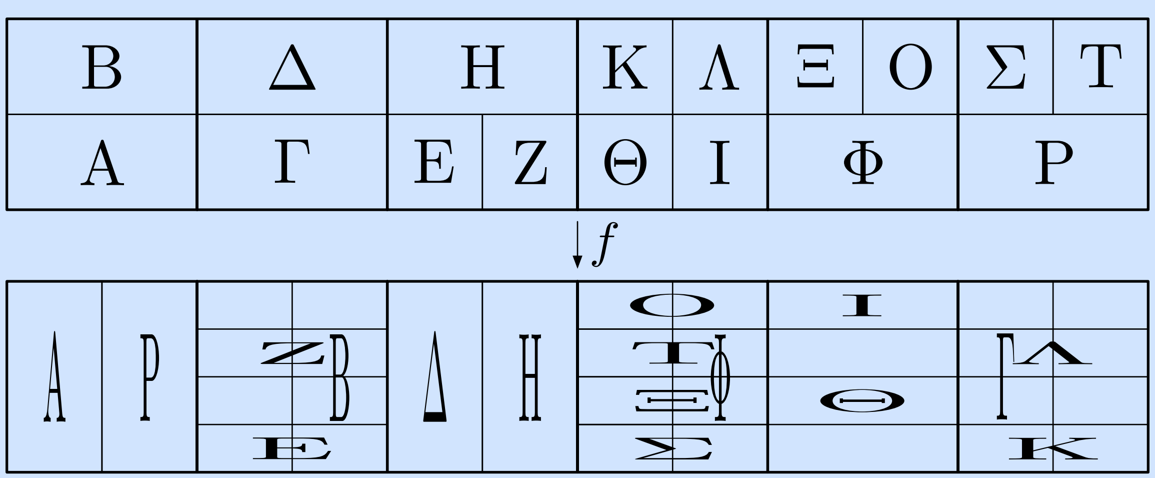
Moore. Unpredictability and undecidability in dynamical systems. PRL. 1990



F	s_1	s_2	s_3	s_4	s_5	s_6
0	$0, s_1, L$	$0, s_6, L$	$0, s_2, R$	$1, s_5, R$	$1, s_4, L$	$1, s_1, L$
1	$1, s_2, L$	$0, s_3, L$	$1, s_3, L$	$0, s_6, R$	$1, s_4, R$	$0, s_4, R$

Neary, Woods. Small weakly universal Turing machines. FCT 2009

A small universal Turing machine...



... represented as a piecewise affine map on [0,6] x [0,1]

Pic credit: Moore & Mertens. The Nature of Computation. OUP 2012.

Generalized shift

deleting/writing many bits

$$f(x,y) = (a_i x + b_i, c_i y + d_i) \quad \text{if } e_i \le y < h_i$$

deleting/writing a bit

$$f(x,y) = \begin{cases} (x/2, 2y) & \text{if } y < 1/2 \\ (x/2 + 1/2, 2y-1) & \text{if } y \ge 1/2 \end{cases}$$

TM state   TM head

...000111.110110101000...

TM tape

test read MSBs of y: i.e., TM state and read symbol

...000111.0101000...

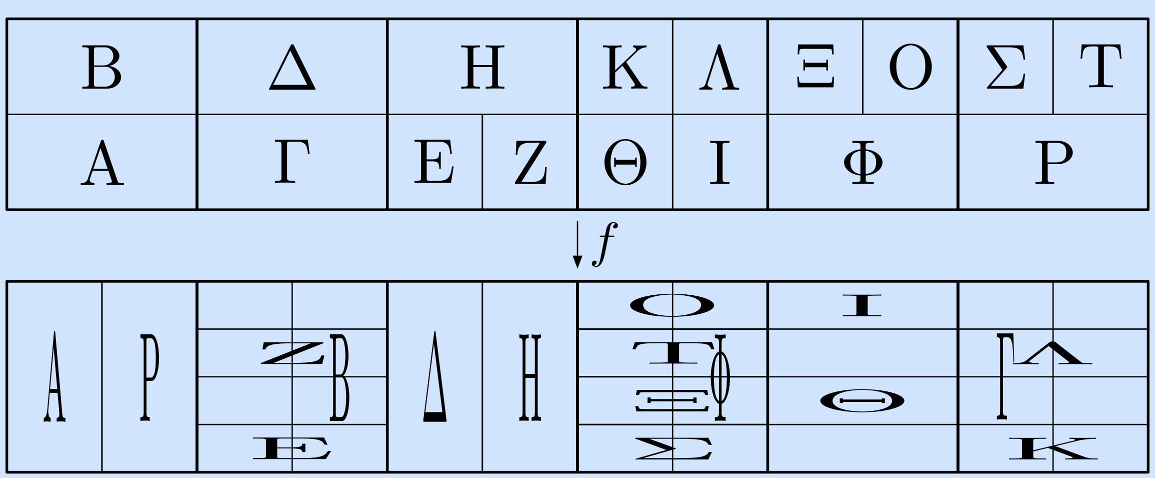
test reads MSB of y

Moore. Unpredictability and undecidability in dynamical systems. PRL. 1990

F	s_1	s_2	s_3	s_4	s_5	s_6
0	$0, s_1, L$	$0, s_6, L$	$0, s_2, R$	$1, s_5, R$	$1, s_4, L$	$1, s_1, L$
1	$1, s_2, L$	$0, s_3, L$	$1, s_3, L$	$0, s_6, R$	$1, s_4, R$	$0, s_4, R$

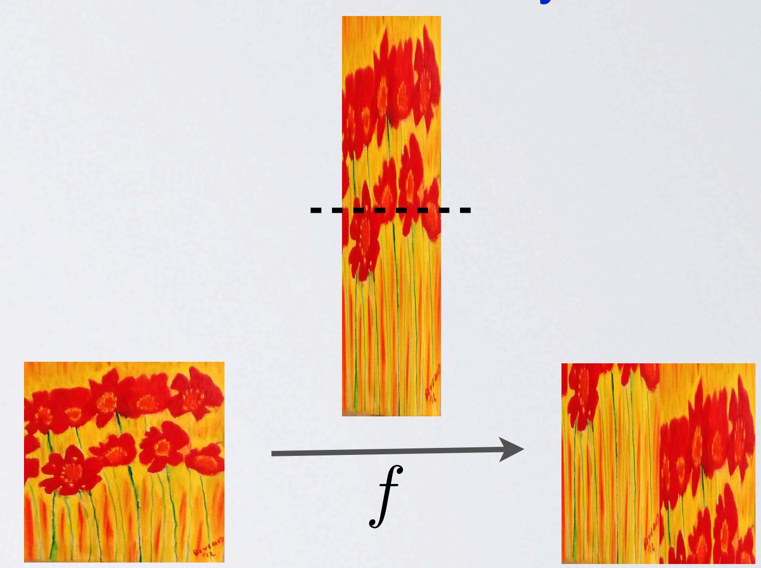
Neary, Woods. Small weakly universal Turing machines. FCT 2009

A small universal Turing machine...



... represented as a piecewise affine map on [0,6] x [0,1]

Pic credit: Moore & Mertens. The Nature of Computation. OUP 2012.



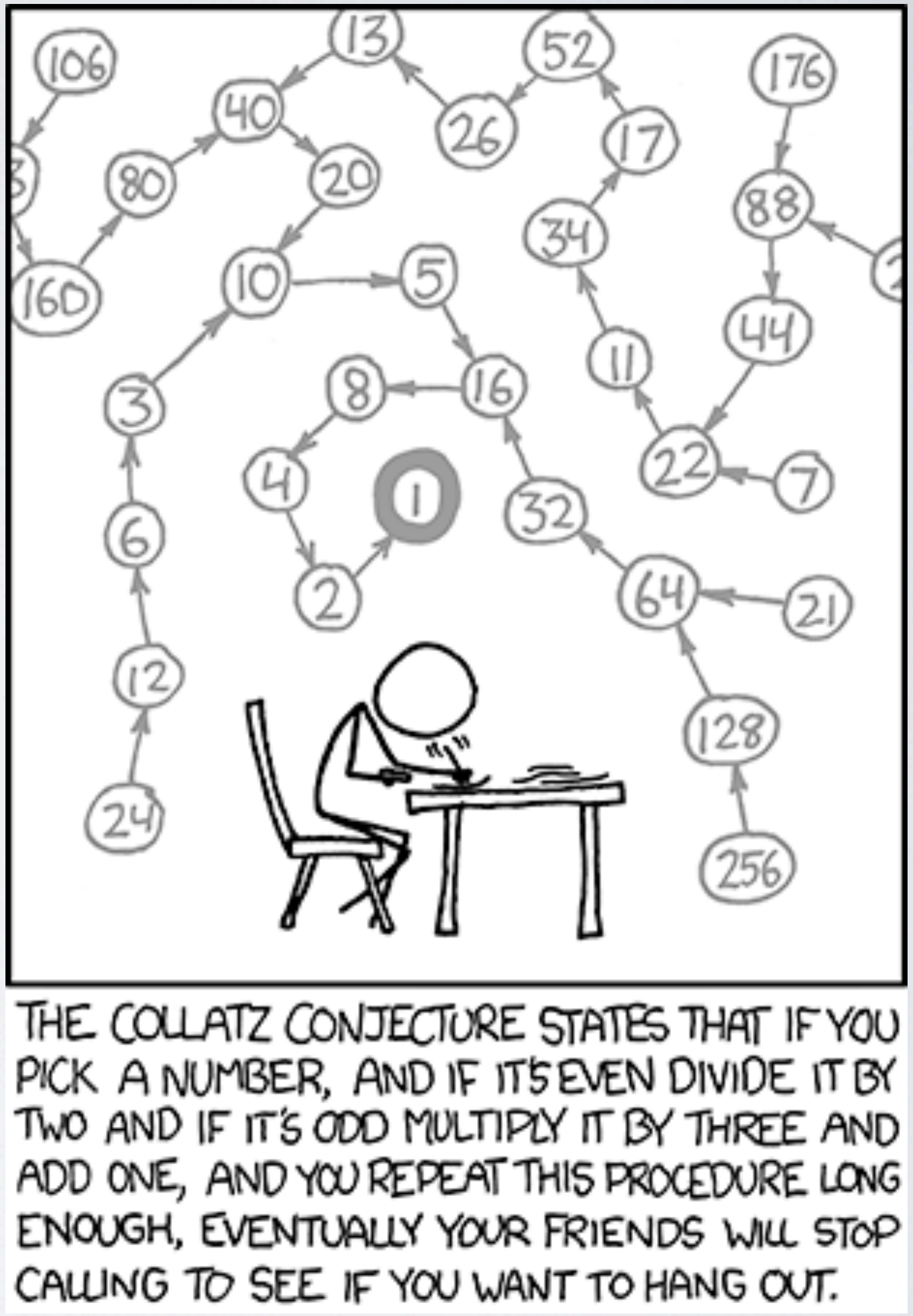
- These generalized shift maps are universal
- Prediction is impossible

Collatz function

- Recall the Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

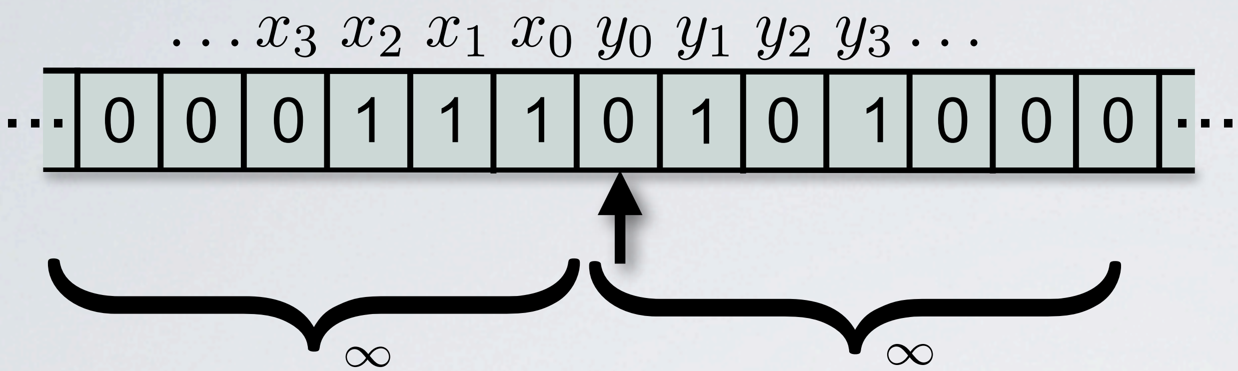
  - For all x, is there some t such that $g^t(x) = 1$?
  - That is, does $g(g(g(\ldots.g(x)\ldots))) = 1$?



http://xkcd.com/710/

- Lets look at some other *Collatz-like functions*

Generalized Collatz functions (2D)

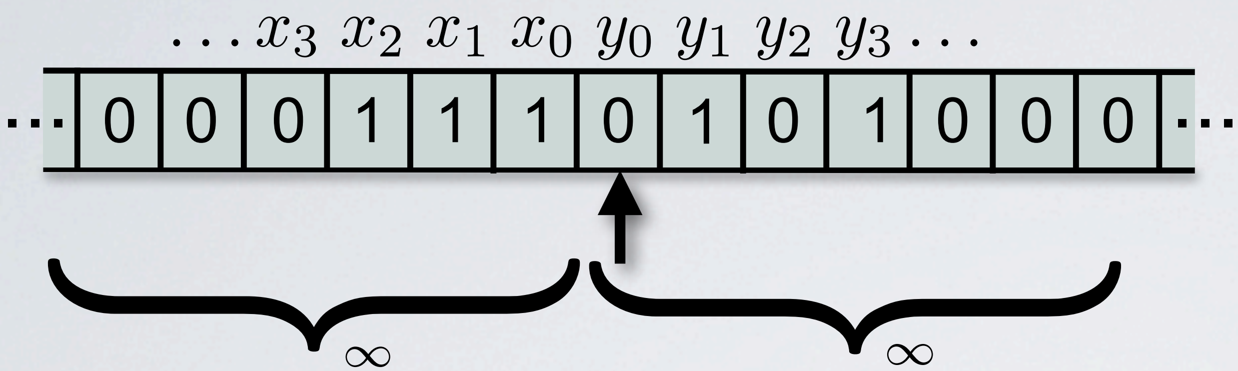


$$x = \sum_{i=0}^{\infty} 2^i x_i \qquad y = \sum_{i=0}^{\infty} 2^i y_i$$

The original Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Generalized Collatz functions (2D)



$$x = \sum_{i=0}^{\infty} 2^i x_i \qquad y = \sum_{i=0}^{\infty} 2^i y_i$$

$$g_{\text{right}}(x, y) = \begin{cases} (2x, y/2) & y \text{ even} \\ (2x+1, (y-1)/2) & y \text{ odd} \end{cases}$$

...000111.0101000...

...0001110.101000...

...00011101.01000...

...000111010.1000...

...0001110101.000...

The original Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Generalized Collatz functions (2D)

$$\ldots x_3\ x_2\ x_1\ x_0\ y_0\ y_1\ y_2\ y_3 \ldots$$

⋯	0	0	0	1	1	1	0	1	0	1	0	0	0	⋯

$$x = \sum_{i=0}^{\infty} 2^i x_i \qquad y = \sum_{i=0}^{\infty} 2^i y_i$$

$$g_{\text{right}}(x,y) = \begin{cases} (2x, y/2) & y \text{ even} \\ (2x+1, (y-1)/2) & y \text{ odd} \end{cases}$$

...000111.0101000...

...0001110.101000...

...00011101.01000...

...000111010.1000...

...0001110101.000...

TM state

TM head

...000111.110110101000...

TM tape

read LSBs of y: i.e., TM state and read symbol

deleting/writing to tape

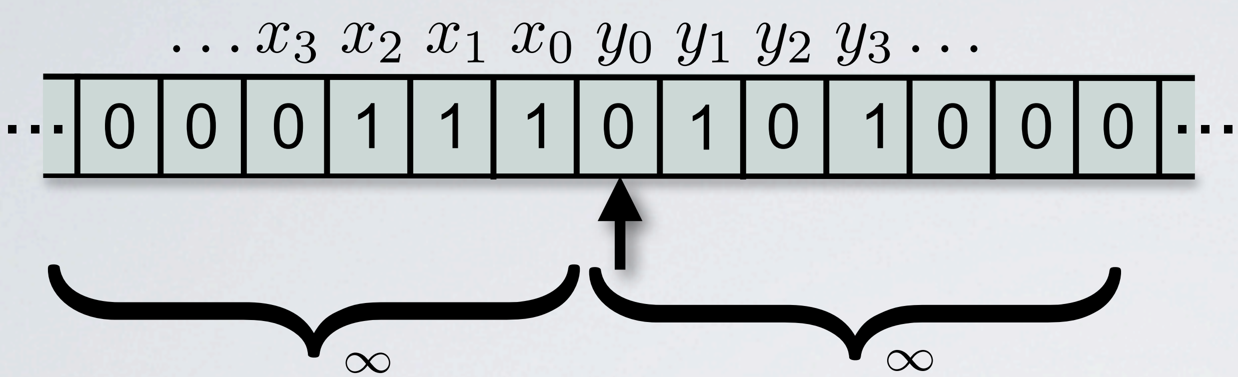
$$g_{\text{M}}(x,y) = (\frac{a_i}{q}(x-i)+b_i, \frac{c_i}{q}(y-i)+d_i) \quad y \equiv i \mod q$$

A generalized 2D Collatz function that simulates some Turing machine

The original Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Generalized Collatz functions (2D)



$$x = \sum_{i=0}^{\infty} 2^i x_i \qquad y = \sum_{i=0}^{\infty} 2^i y_i$$

$$g_{\text{right}}(x,y) = \begin{cases} (2x, y/2) & y \text{ even} \\ (2x+1, (y-1)/2) & y \text{ odd} \end{cases}$$

...000111.0101000...

...0001110.101000...

...00011101.01000...

...000111010.1000...

...0001110101.000...

TM state

TM head

...000111.110110101000...

TM tape

read LSBs of y: i.e., TM state and read symbol

deleting/writing to tape

$$g_{\text{M}}(x,y) = \left(\frac{a_i}{q}(x-i) + b_i, \frac{c_i}{q}(y-i) + d_i\right) \quad y \equiv i \mod q$$

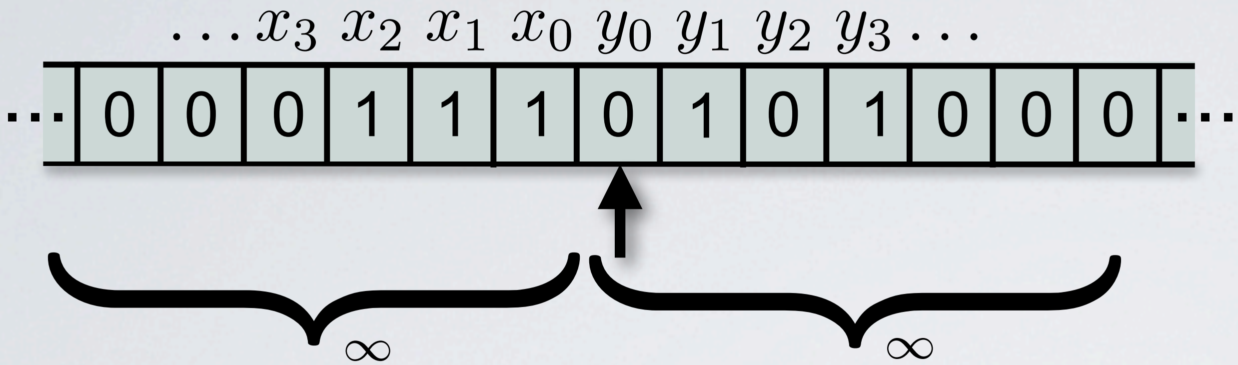
A generalized 2D Collatz function that simulates some Turing machine

- Generalized 2D Collatz functions are universal

The original Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Generalized Collatz functions (2D)



$$x = \sum_{i=0}^{\infty} 2^i x_i \qquad y = \sum_{i=0}^{\infty} 2^i y_i$$

$$g_{\text{right}}(x,y) = \begin{cases} (2x, y/2) & y \text{ even} \\ (2x+1, (y-1)/2) & y \text{ odd} \end{cases}$$

...000111.0101000...

...0001110.101000...

...00011101.01000...

...000111010.1000...

...0001110101.000...

TM state

TM head

...000111.110110101000...

TM tape

read LSBs of y: i.e., TM state and read symbol

deleting/writing to tape

$$g_{\mathrm{M}}(x,y) = \left(\frac{a_i}{q}(x-i) + b_i, \frac{c_i}{q}(y-i) + d_i\right) \quad y \equiv i \mod q$$

A generalized 2D Collatz function that simulates some Turing machine

- Generalized 2D Collatz functions are universal

The original Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Conway. Unpredictable iterations. 1972

Koiran, Moore. Closed form analytic maps in one and two dimensions can simulate Turing machines. 1996

Generalized Collatz functions (1D)

Lets simulate a 2D function $g(L, R)$ with a 1D function $g(x)$

Combine 2 variables into 1 using an exponential pairing function: $(L, R) \to 2^L 3^R = x$

We can easily increment *L* or *R:* $2x = 2^{L+1}3^R, \quad \text{or} \quad 3x = 2^L 3^{R+1}$
This can be used for addition and subtraction

Use another variable for temporary storage, which lets us do multiplication: $x = 2^L 3^R 5^T$

$$g_{\mathrm{M}}(x) = \frac{a_i}{q}(x - i) + b_i \quad x \equiv i \mod q$$

A generalized 1D Collatz function that simulates some Turing machine

The original Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x + 1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Generalized Collatz functions (1D)

Lets simulate a 2D function $g(L, R)$ with a 1D function $g(x)$

Combine 2 variables into 1 using an exponential pairing function: $(L, R) \to 2^L 3^R = x$

We can easily increment *L* or *R:* $2x = 2^{L+1}3^R, \quad \text{or} \quad 3x = 2^L 3^{R+1}$
This can be used for addition and subtraction

Use another variable for temporary storage, which lets us do multiplication: $x = 2^L 3^R 5^T$

$$g_{\mathrm{M}}(x) = \frac{a_i}{q}(x - i) + b_i \quad x \equiv i \mod q$$

A generalized 1D Collatz function that simulates some Turing machine

- Generalized 1D Collatz functions are universal (although slow)

The original Collatz function:

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x + 1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Generalized Collatz functions (1D)

Lets simulate a 2D function $g(L,R)$ with a 1D function $g(x)$

Combine 2 variables into 1 using an exponential pairing function: $(L,R) \to 2^L 3^R = x$

We can easily increment *L* or *R:* $2x = 2^{L+1}3^R, \quad \text{or} \quad 3x = 2^L 3^{R+1}$
This can be used for addition and subtraction

Use another variable for temporary storage, which lets us do multiplication: $x = 2^L 3^R 5^T$

$$g_{\mathrm{M}}(x) = \frac{a_i}{q}(x-i) + b_i \quad x \equiv i \mod q$$

A generalized 1D Collatz function that simulates some Turing machine

- Generalized 1D Collatz functions are universal (although slow)

The original Collatz function:

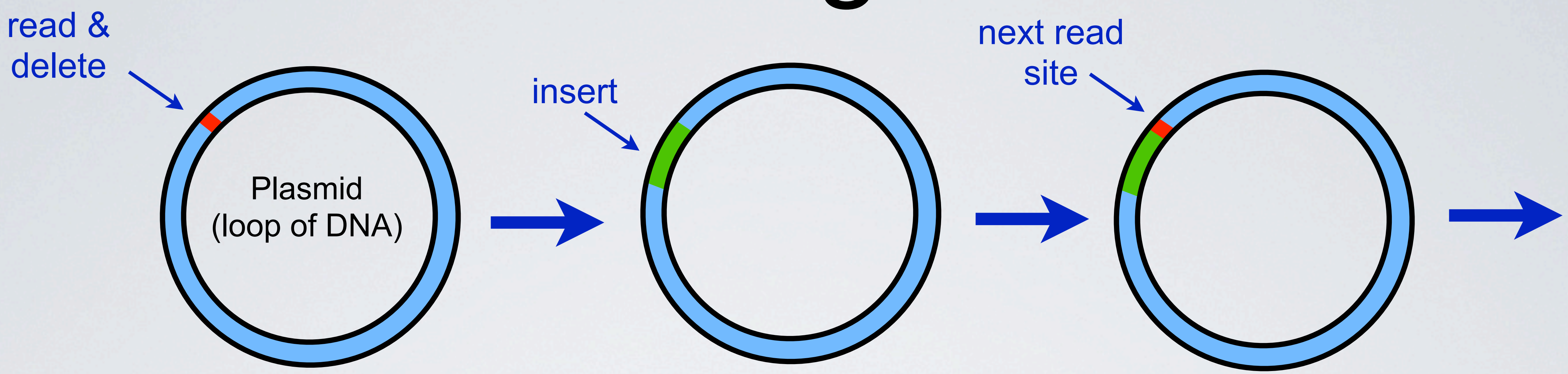
$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Conway. Unpredictable iterations. 1972
Koiran, Moore. Closed form analytic maps in one and two dimensions can simulate Turing machines. 1996

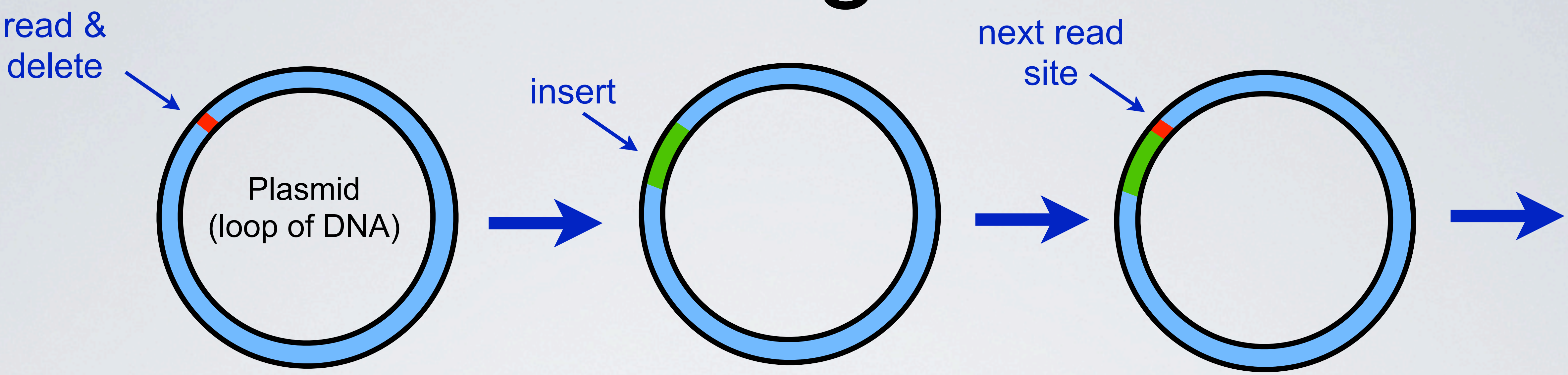

CATUNAMBÚ
Predilect
Espresso

Too many numbers?

Something else ...

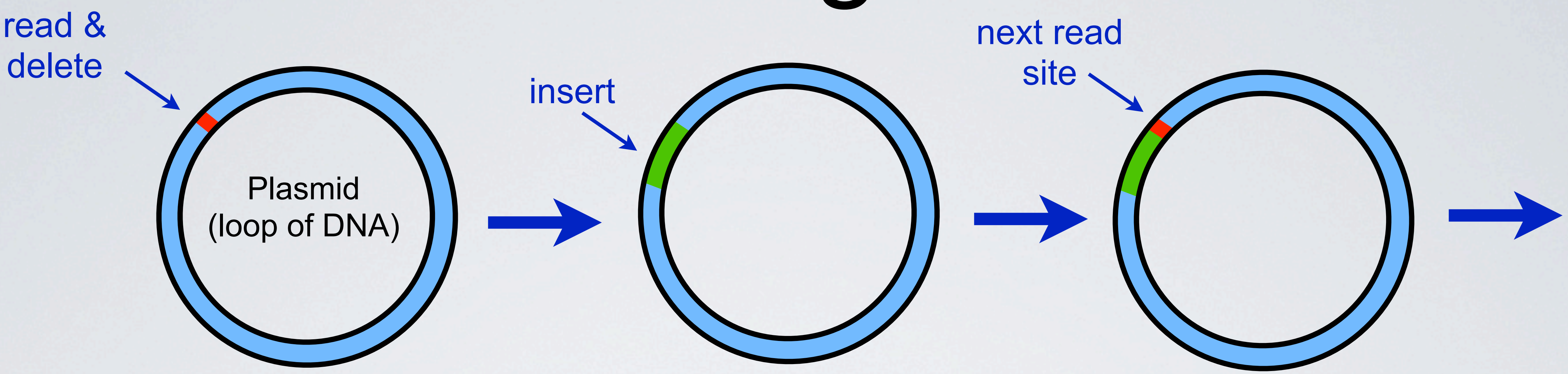


Something else ...



- This is universal!

Something else ...



- 2-tag systems:
  - Read on the left, append on the right
  - Delete 2 symbols
  - Example:

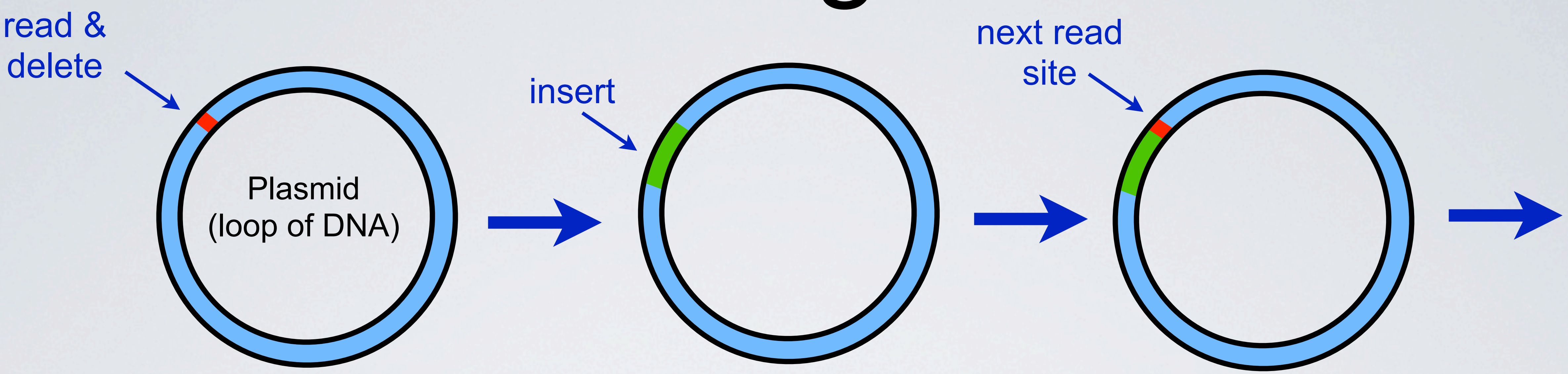
$$a \to bc \qquad b \to a \qquad c \to aaa$$

$$\begin{array}{l} aaa \\ \quad abc \\ \qquad cbc \\ \qquad\quad caaa \\ \qquad\qquad aaaaa \end{array}$$

De Mol. Tag systems and Collatz-like functions. TCS 2007

- This is universal!

Something else ...



- 2-tag systems:
  - Read on the left, append on the right
  - Delete 2 symbols
  - Example:
- This is universal!

$$a \to bc$$
$$b \to a$$
$$c \to aaa$$

aaa
abc
cbc
$caaa$
$aaaaa$

Simulates the Collatz function!

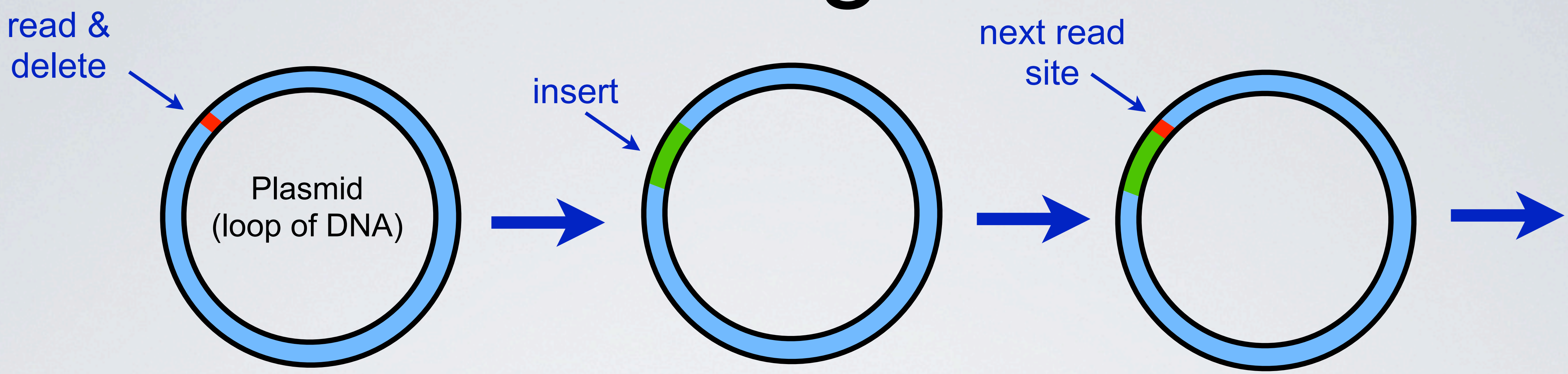
$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ (3x+1)/2 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

De Mol. Tag systems and Collatz-like functions. TCS 2007

The original Collatz

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

Something else ...



- 2-tag systems:
  - Read on the left, append on the right
  - Delete 2 symbols
  - Example:

- This is universal!

$$a \to bc$$
$$b \to a$$
$$c \to aaa$$

aaa
abc
cbc
$caaa$
$aaaaa$

De Mol. Tag systems and Collatz-like functions. TCS 2007

Simulates the Collatz function!

$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ (3x+1)/2 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

The original Collatz

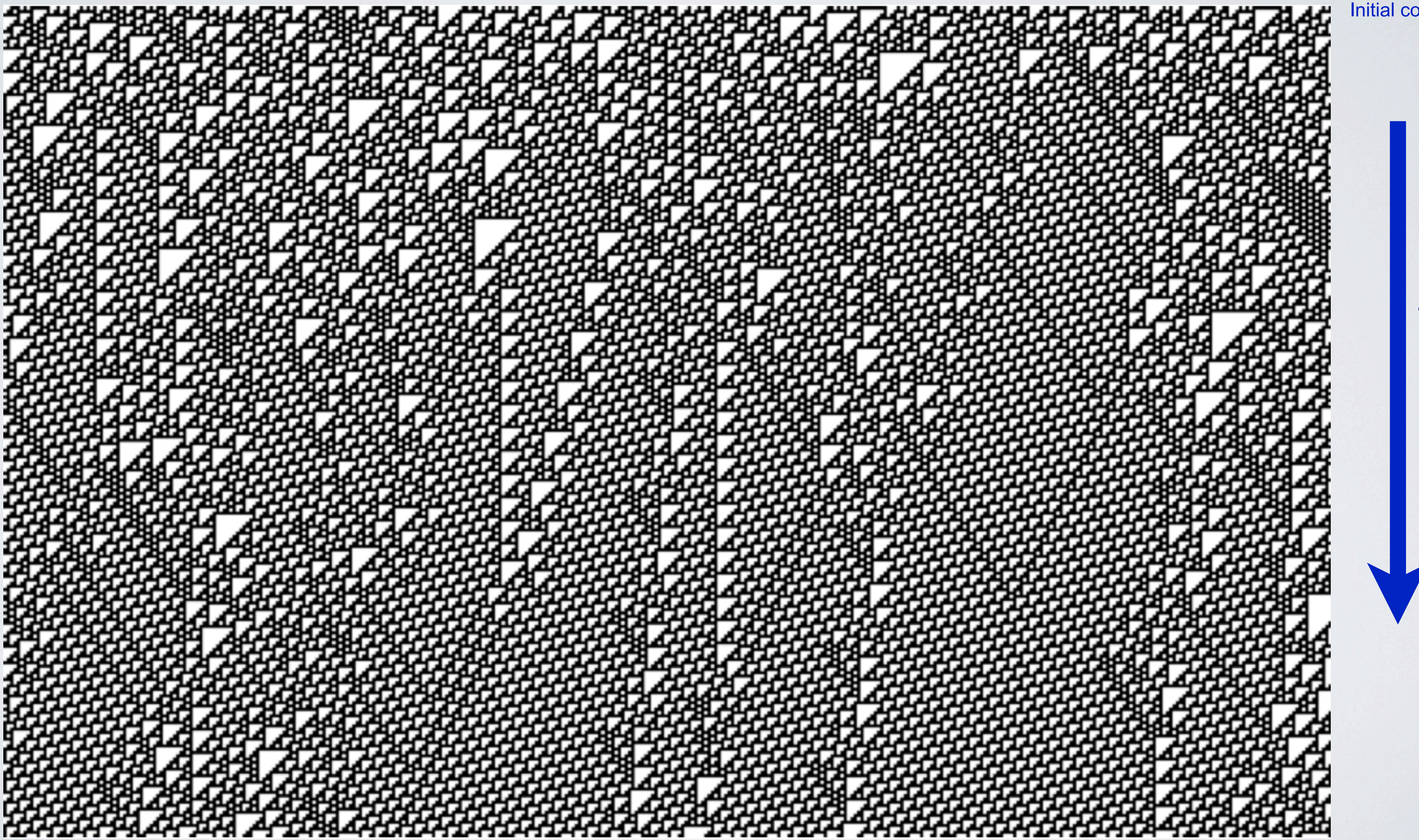
$$g(x) = \begin{cases} x/2 & \text{if } x \equiv 0 \mod 2 \\ 3x+1 & \text{if } x \equiv 1 \mod 2 \end{cases}$$

- 2-tag systems simulate Generalized 1D Collatz functions
- 2-tag systems are universal
  Cocke, Minsky. Universality of tag systems with P = 2. JACM 1964
- 2-tag systems are not so slow!
  Woods, Neary. On the time complexity of 2-tag systems and small universal Turing machines. FOCS 2006

Rule 110

TMs → 2-tag systems → cyclic-tag systems → Rule 110

- Rule 110 simulates tag systems

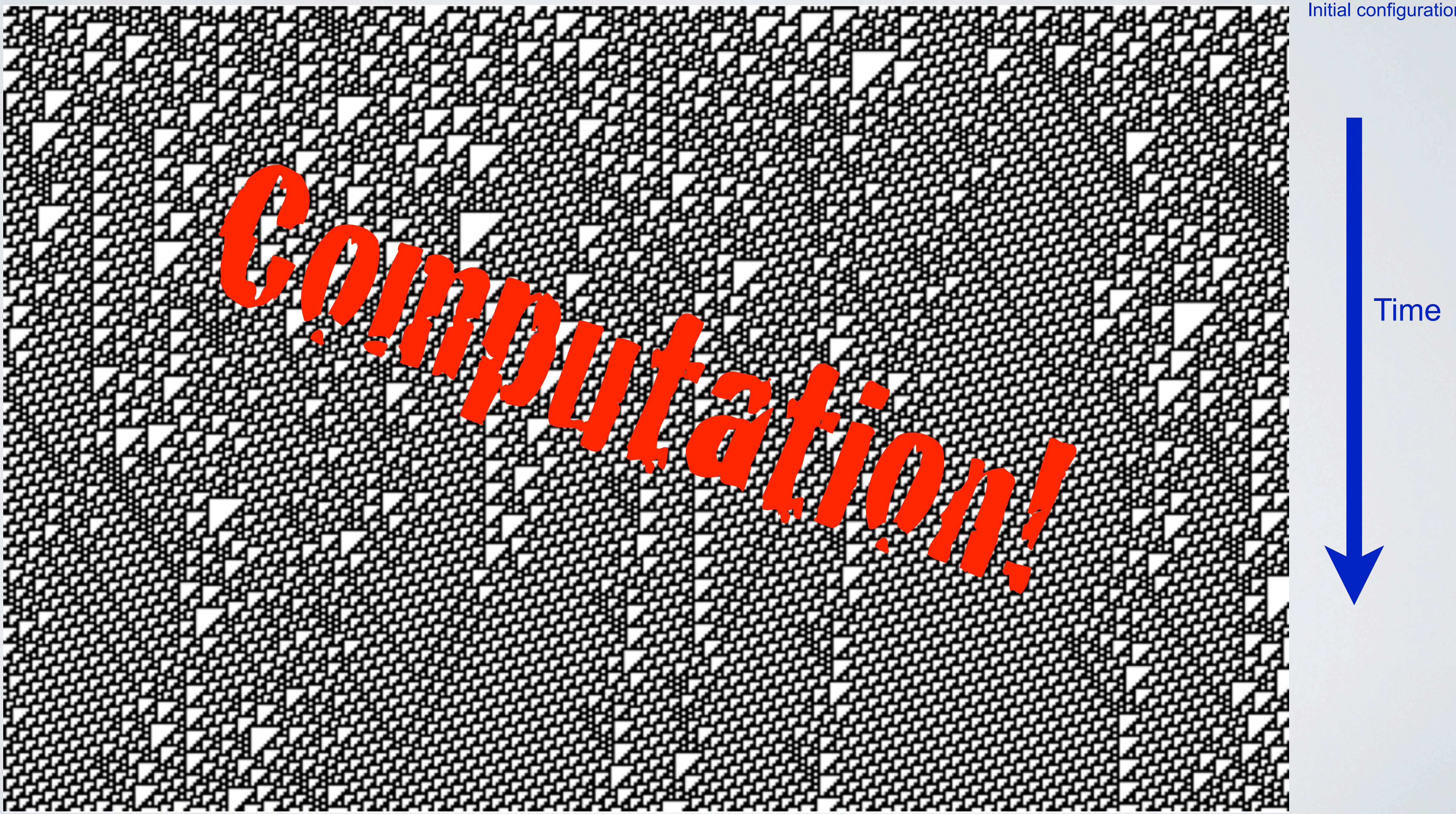


Cook, Matthew. Universality in Elementary Cellular Automata. Complex systems (2004) 15(1):1–40

Rule 110

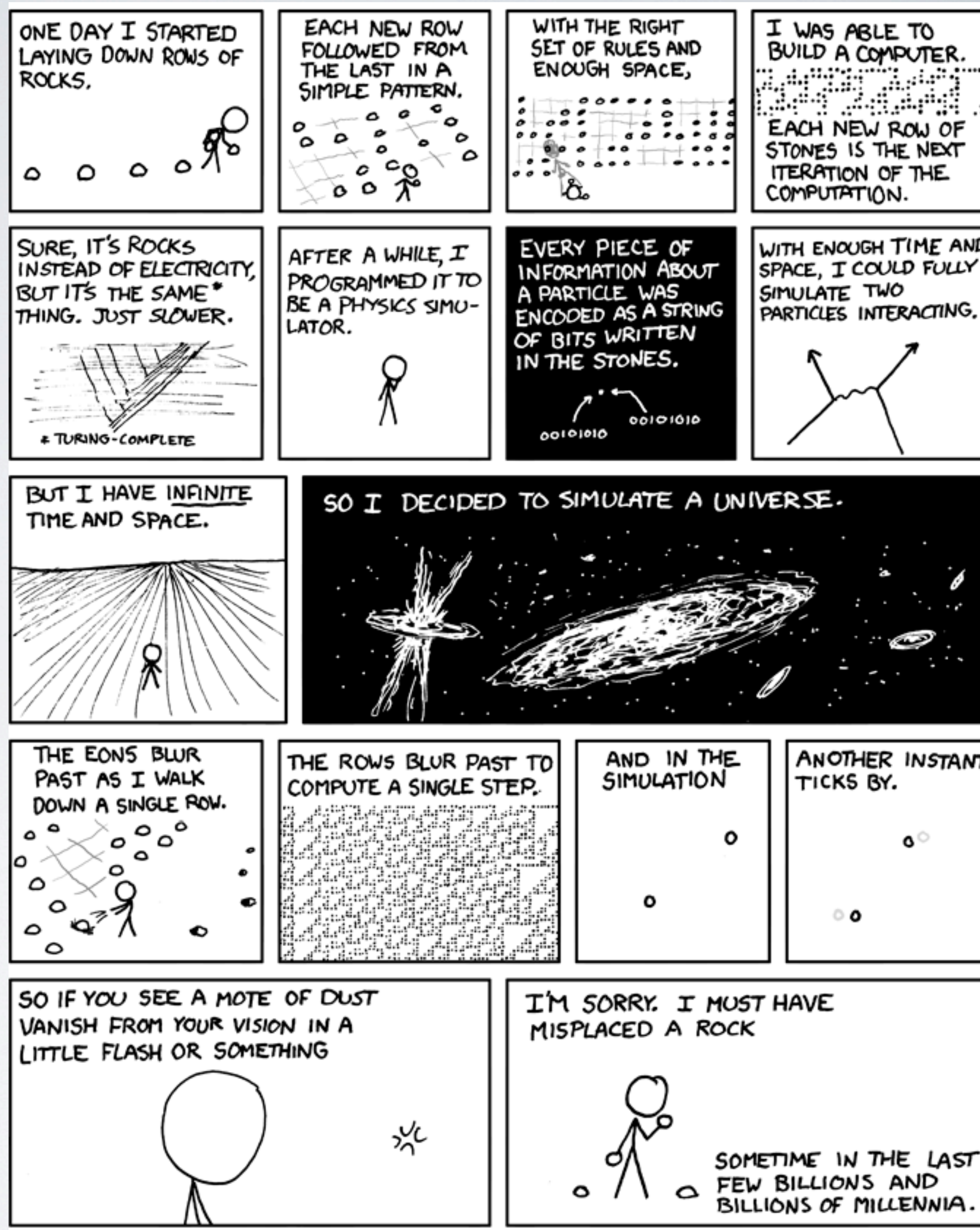
TMs → 2-tag systems → cyclic-tag systems → Rule 110

- Rule 110 simulates tag systems



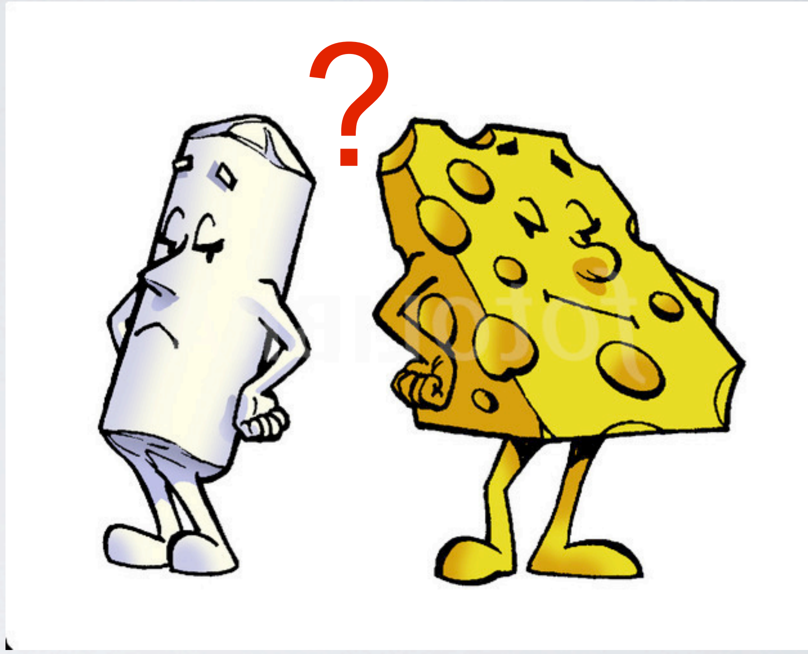
Cook, Matthew. Universality in Elementary Cellular Automata. Complex systems (2004) 15(1):1–40

- Rule 110 is a pop star!

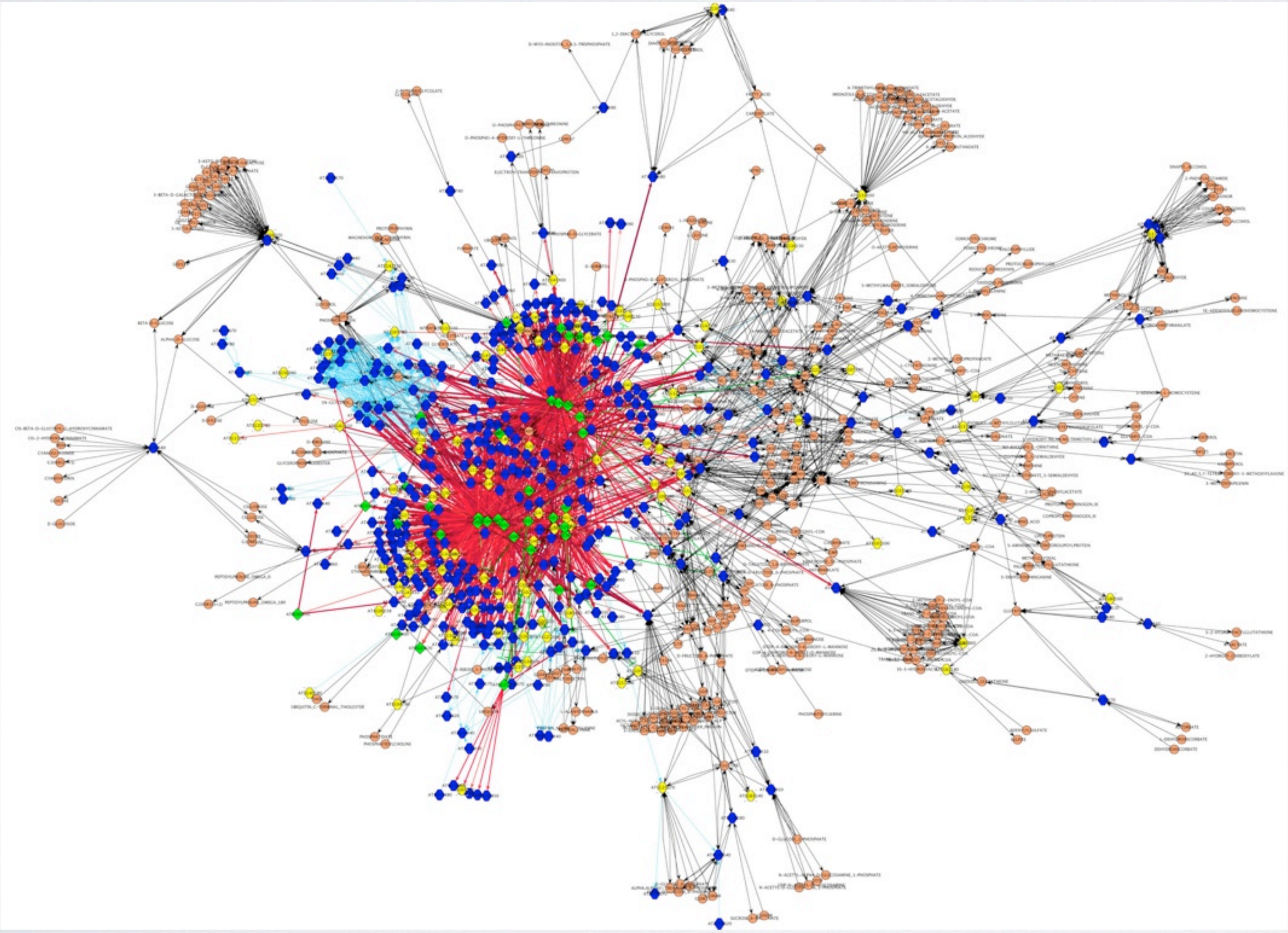


xkcd, #505: A bunch of rocks

- How is all of this related to molecules?



Miles Kelly. Fotolibre



Sumazin et al. Cell 147(2). 2011

Universal molecules

- Molecular systems capable of universal computation:
  - Chemical reactions networks

    Soloveichik, Cook, Winfree, Bruck. Computation with Finite Stochastic Chemical Reaction Networks. Natural Computing 2008
  - DNA strand displacement systems

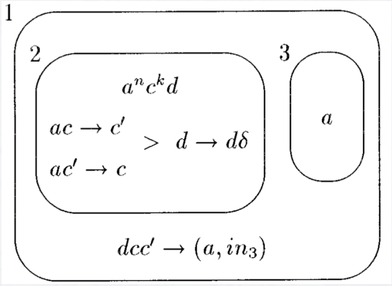
    Soloveichik, Seelig, Winfree. DNA as a Universal Substrate for Chemical Kinetics. PNAS 2010
  - DNA tile self-assembly systems

    Winfree. On the Computational Power of DNA Annealing and Ligation. DNA2. 1996
  - DNA polymer + restriction enzymes

    Rothemund. A DNA and restriction enzyme implementation of Turing Machines. DNA2. 1996
  - DNA polymer + hypothetical Enzymes

    Bennett. Thermodynamics of computation - A review. IJTP 1982
  - Membrane systems

    Păun. Computing with Membranes. JCSS. 2000
  - ....



Prediction

- We saw that with very simple devices we get a kind of “maximal complexity”
- These systems are universal: they can run any algorithm
- Any (molecular) system that embeds/simulates even these simple systems is impossible to predict in the long term. E.g. does the system ever reach a given configuration? Produce the right answer? Halt?
- But these questions are about behavior in the limit

- What about short term prediction? That is, time-bounded prediction?

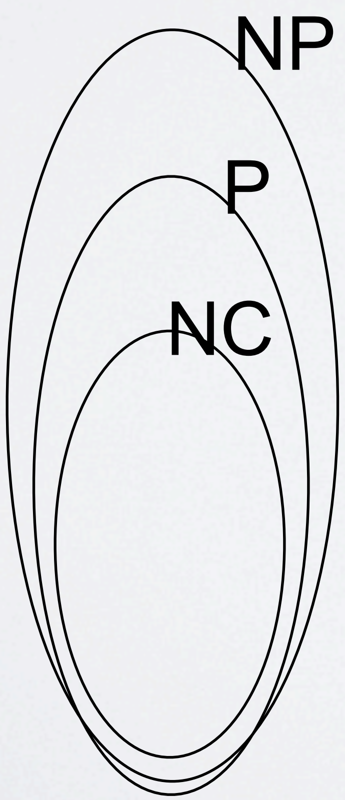
- Can systems that carry out computations be predicted using explicit simulations that run significantly faster than the systems themselves?

- What about short term prediction? That is, time-bounded prediction?

- For example, for a system that runs in time t, can we simulate it in time $O(\log t)$? $O(\log t)^k$?

Computational complexity

- The complexity of problems can be measured by the amount of *resources* needed to solve them
- P is the class of problems solved by Turing machines that run in time polynomial of their input length
- Problems outside of P are said to be intractable
- NP is the class of problems that are solvable in polynomial time on nondeterministic Turing machines
- P is contained in NP



P

Oh so famous!

$$\mathrm{P} = \bigcup_{k\in\mathbb{N}} \text{Turing machine time } n^k$$

$$\mathrm{NC} = \bigcup_{k\in\mathbb{N}} \text{parallel time } O(\log n)^k$$

(and polynomial processors)

Not so famous!

Computational complexity

- P is the class of problems solved by Turing machines that run in time polynomial of their input length

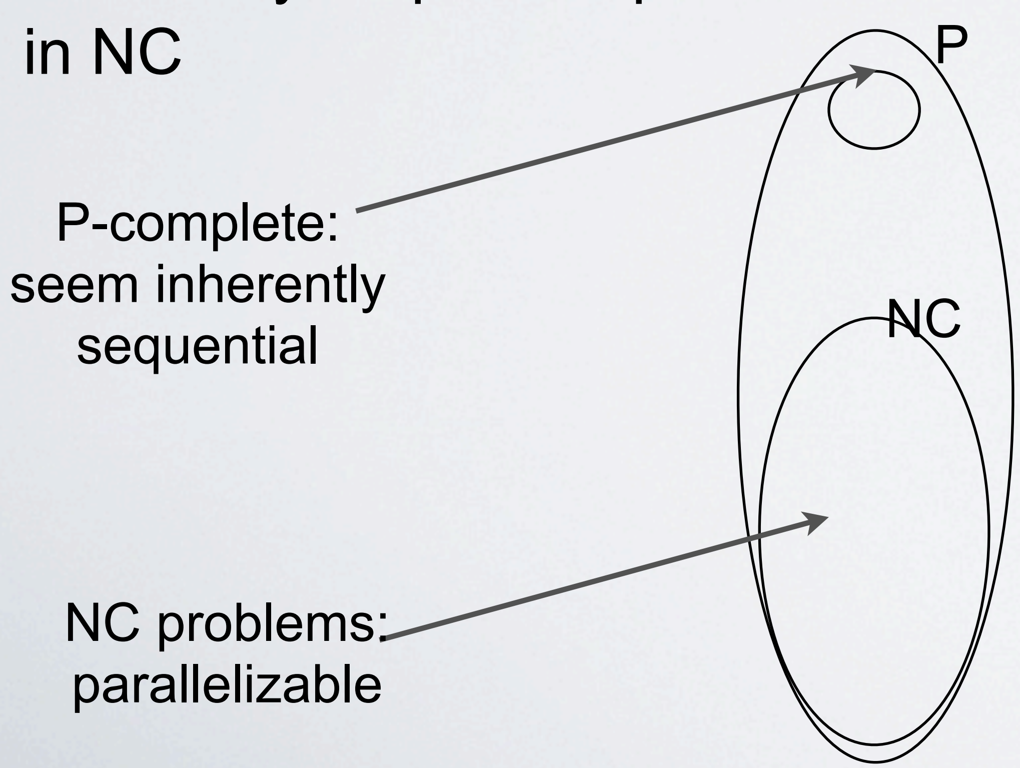
P

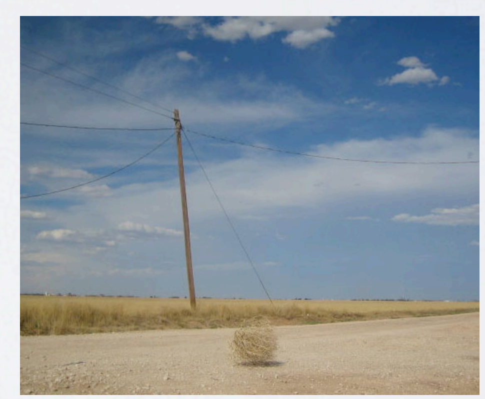
- NC ("Nick's class") is the class of problems that are solvable in polylogarithmic time on massively parallel computers (massively parallel = polynomial number of processors)

$$\mathrm{P} = \bigcup_{k \in \mathbb{N}} \text{Turing machine time } n^k$$

- NC is contained in P.

- Inherently sequential problems in P, believed not to lie in NC





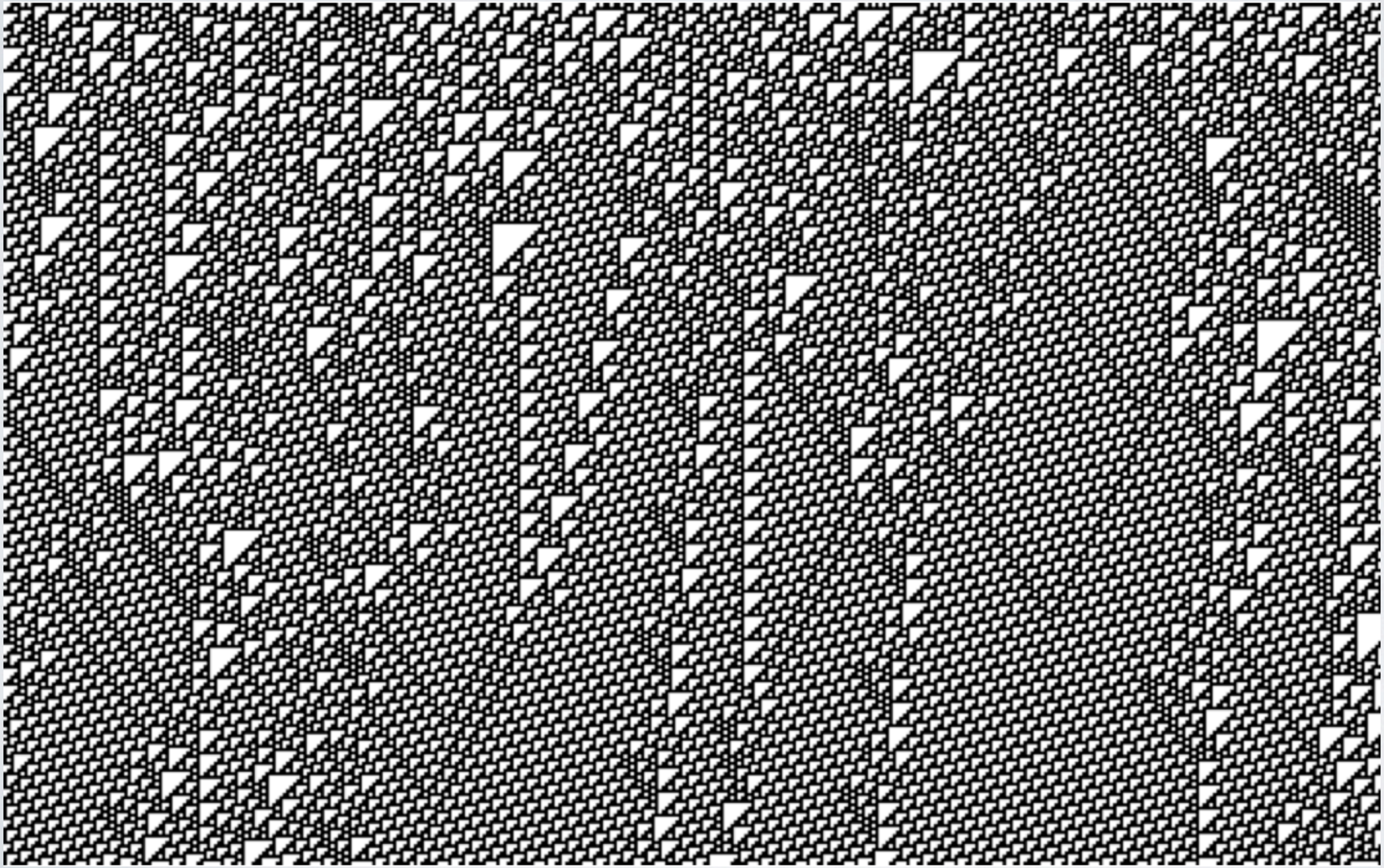
NC

$$\mathrm{NC} = \bigcup_{k \in \mathbb{N}} \text{parallel time } O(\log n)^k$$

$$\text{(and polynomial processors)}$$

Rule 110

- What is going on here?

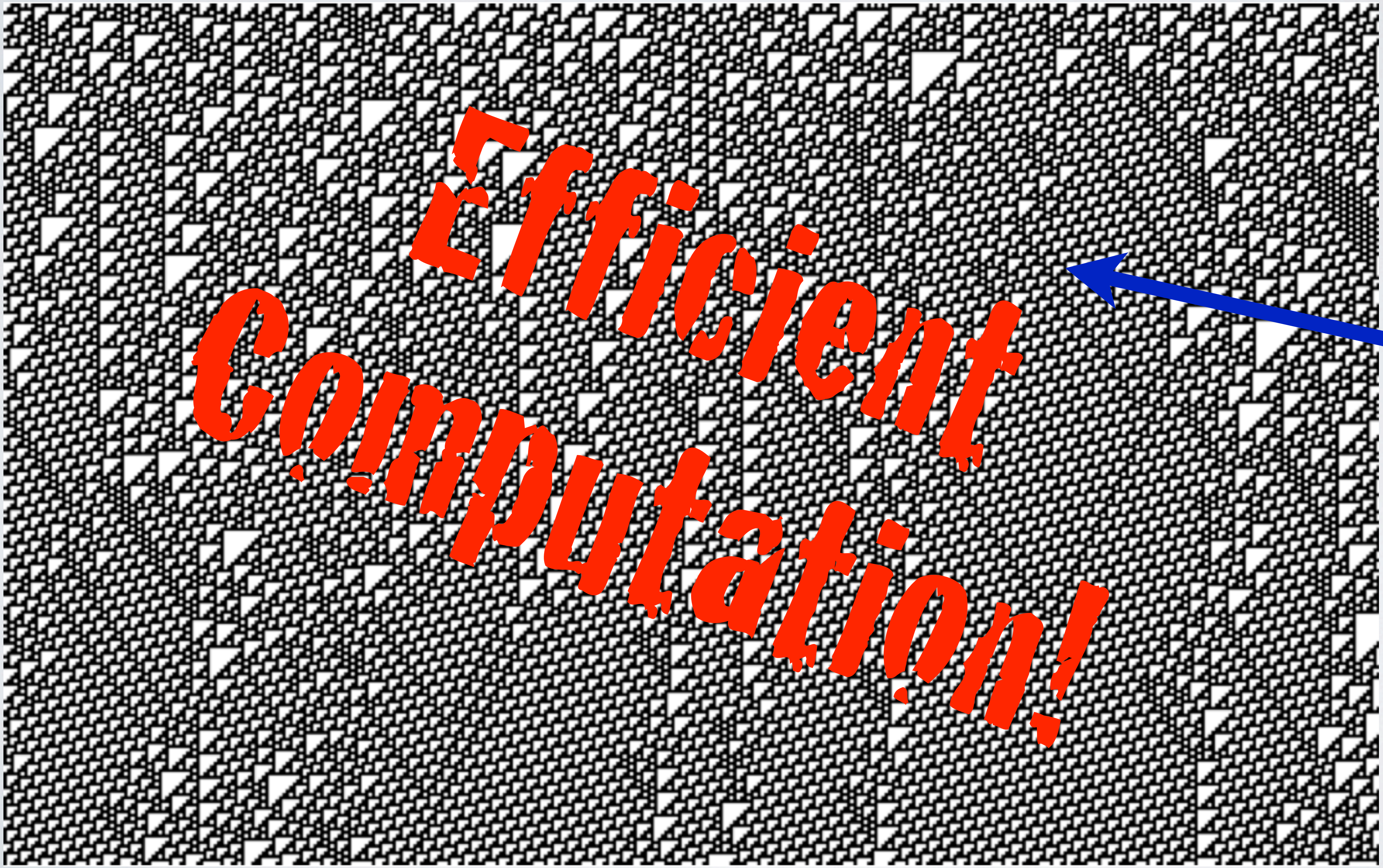


Cook, Matthew. Universality in Elementary Cellular Automata. Complex systems (2004) 15(1):1–40

Rule 110

- What is going on here?

Neary, Woods. P-completeness of cellular automaton Rule 110. ICALP 2006



I.e. not a lot of junk!

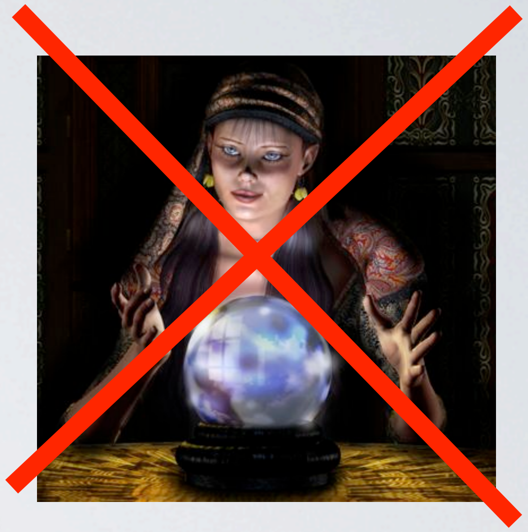
Cook, Matthew. Universality in Elementary Cellular Automata. Complex systems (2004) 15(1):1–40

Conclusion I

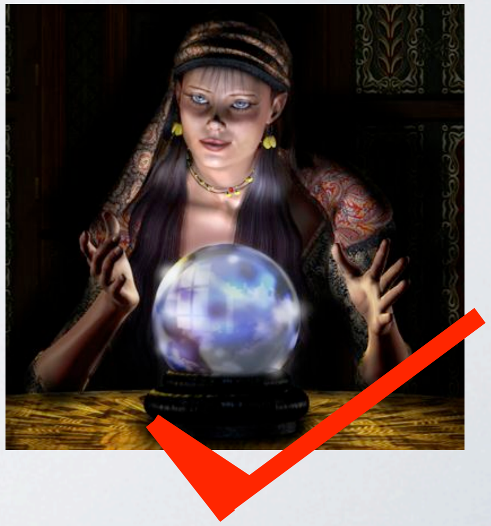
- Almost all questions about the long term dynamics of universal models of computation are undecidable
- We saw various types of simulation that lead to computational universality
- There are *ridiculously* simple systems that are capable of universal computation!
- Universality => Long term prediction is impossible
- Efficient universality => Short term prediction (i.e. faster-than-explicit simulation) is also impossible
- We can use simulation to determine “how much computation” a system is doing

Conclusion II

- Any system that efficiently simulates the following is efficiently universal, therefore they can be predicted no better than by explicit simulation (assuming P =/= NC):
  - Turing machines, cellular automata, Rule 110, 2-tag systems, 2D generalzed Collatz functions, 2D generalized shifts, ...



- Systems that carry out little or no computation (walker example, exponential decay reaction), or that are provably inefficient* at certain tasks, can be predicted much faster than by explicit simulation



Recommended reading

- Recommended Reading
  - Moore, Mertens. “The Nature of Computation” Oxford University Press, 2012

    “Indeed, if the physics of our universe could not support computation, it’s doubtful that it could support life”

  - Greenlaw, Hoover, Ruzzo. “Limits to parallel computation: P-completeness theory” Oxford University Press, 1995

- Thanks: DNA18 organizers & steering committee, Beverley Henley, Cris Moore, Niall Murphy, Moya Chen. Members of Erik Winfree’s and Shuki Bruck’s research groups. NSF for funding.

fin