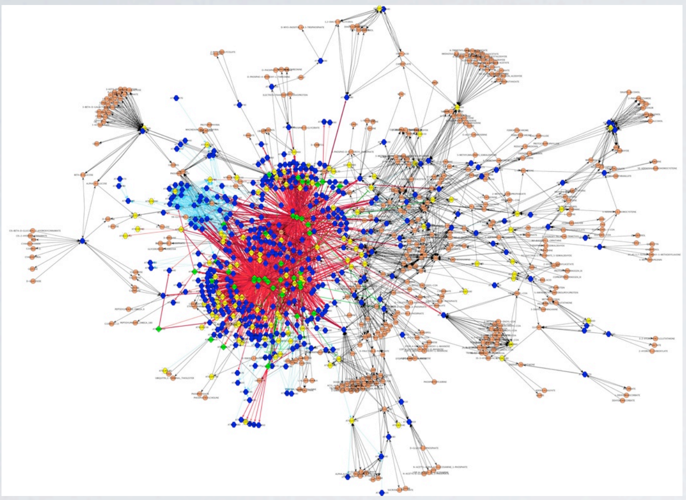
Active Self-Assembly of Algorithmic Shapes and Patterns in Polylogarithmic Time

Damien Woods[1], Ho-Lin Chen[2], Scott Goodfriend[3],
Nadine L. Dabby[1], Erik Winfree[1], Peng Yin[4]

[1]Caltech, [2]National Taiwan University, [3]UC Berkeley, [4]Harvard

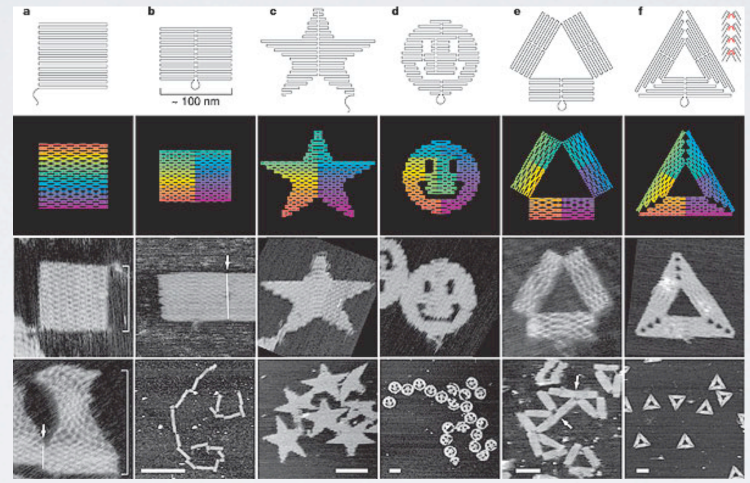
Motivation

- Nature computes

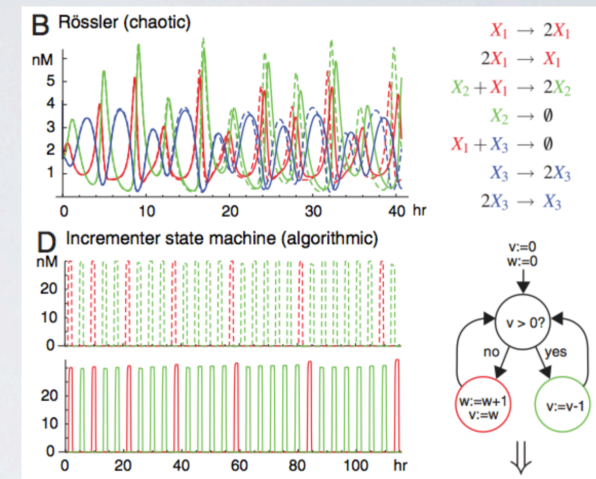


Sumazin et al. Cell 147(2). 2011

- Engineers are building nanoscale molecular (chemical) computers

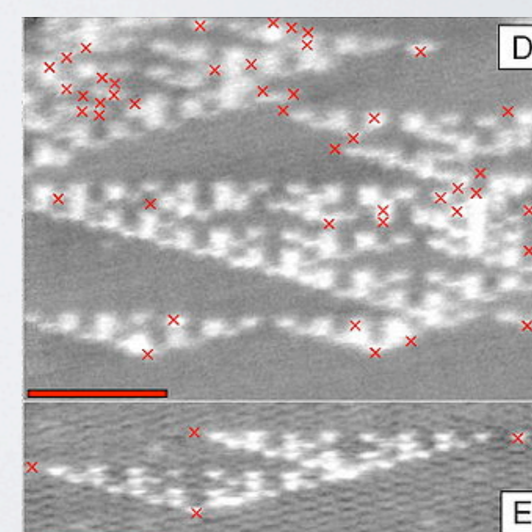


Rothemund, PWK. Folding DNA to create nanoscale shapes and patterns. Nature 2006



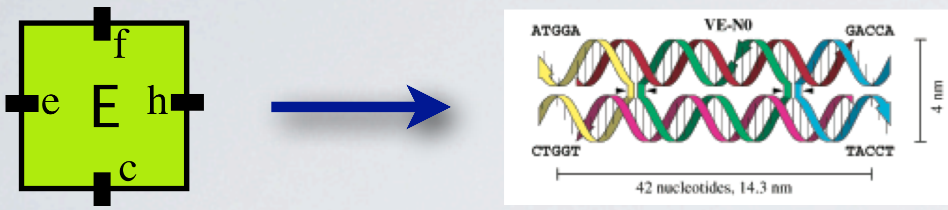
Soloveichik, Seelig, Winfree. DNA as a Universal Substrate for Chemical Kinetics. PNAS 2010

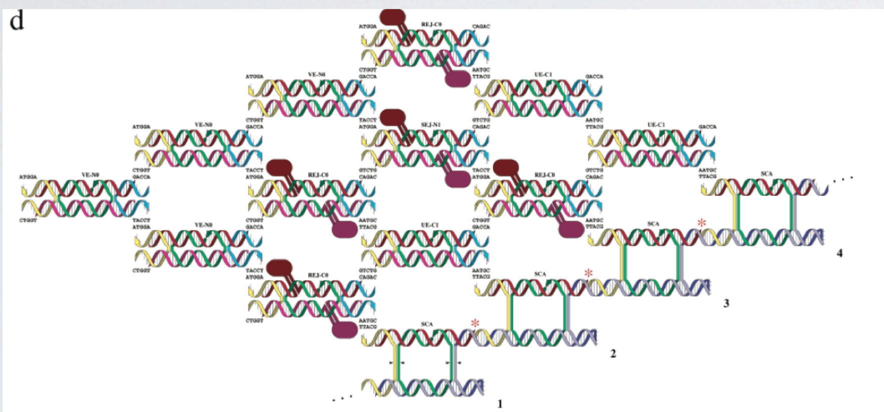
- We need a computational theory of self assembly and molecular interactions
- Perhaps more importantly (for us), we can find interesting theoretical problems: computation, geometry, asynchronousity, kinetics, thermodynamics



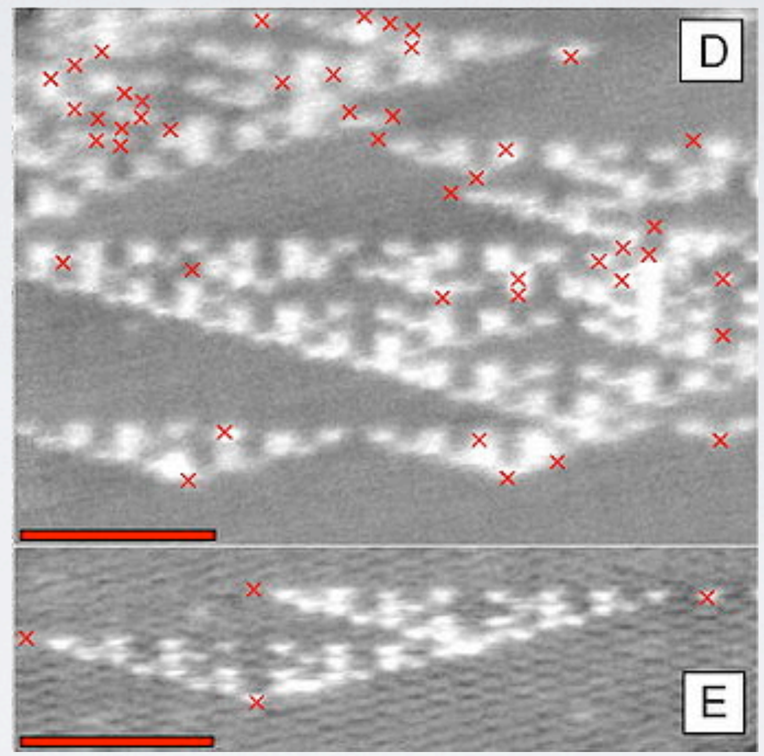
Rothemund, Papadakis, Winfree 2004

Algorithmic self-assembly with tiles: computation and geometry

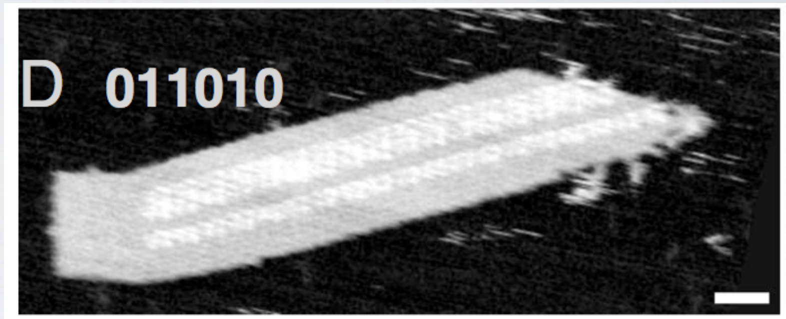




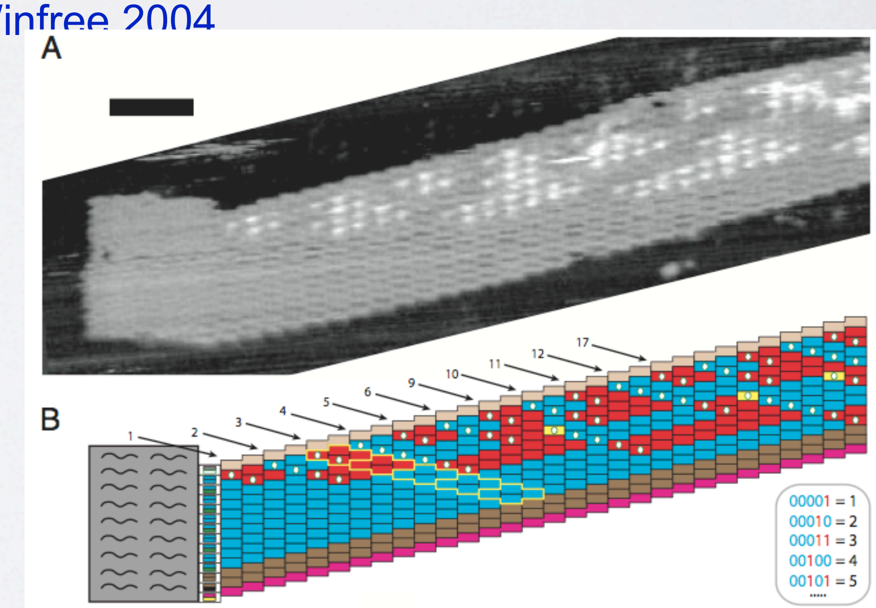
Barish, Rothemund, Winfree 2005



Rothemund, Papadakis, Winfree 2004



Barish, Schulman Rothemund, Winfree 2009



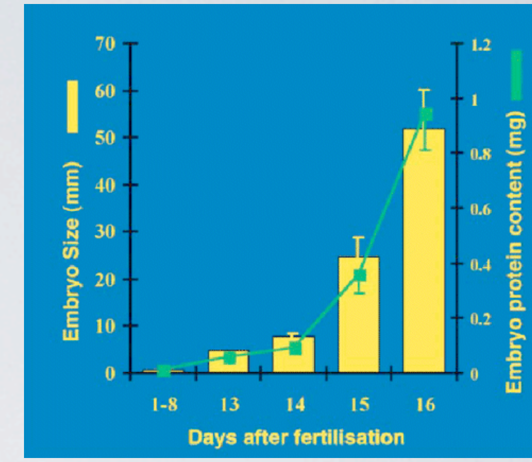
Barish, Schulman Rothemund, Winfree 2009

- Seeman built tiles out of DNA in the laboratory
- Winfree showed that DNA tiles can run algorithms (much like cellular automata, Wang tiles)
- A variety of algorithmic tile assembly systems have been built from DNA

INSPIRATION: DEVELOPMENTAL BIOLOGY

Newly hatched zebra fish, 3 days after zygote

Zebrafish. http://exploratorium.edu/imaging_station/

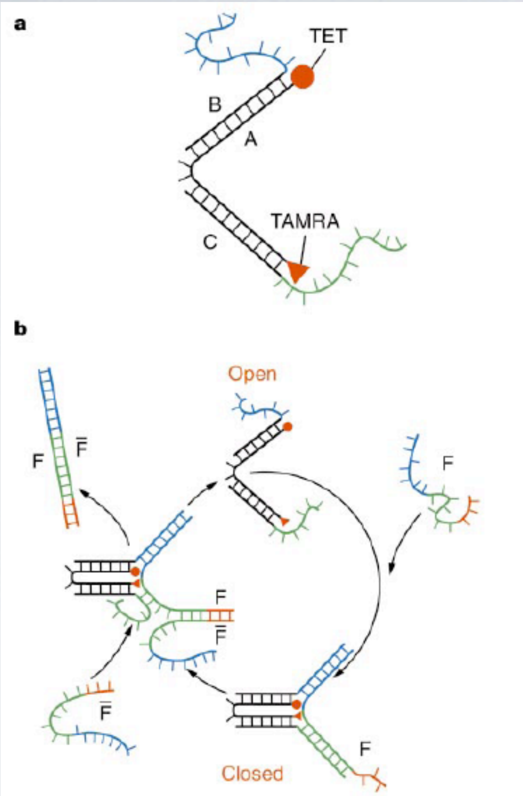


Morris *et al*. Teagasc report. June 2001. Cattle embryos: exponential increase in size and protein content E1-E16.

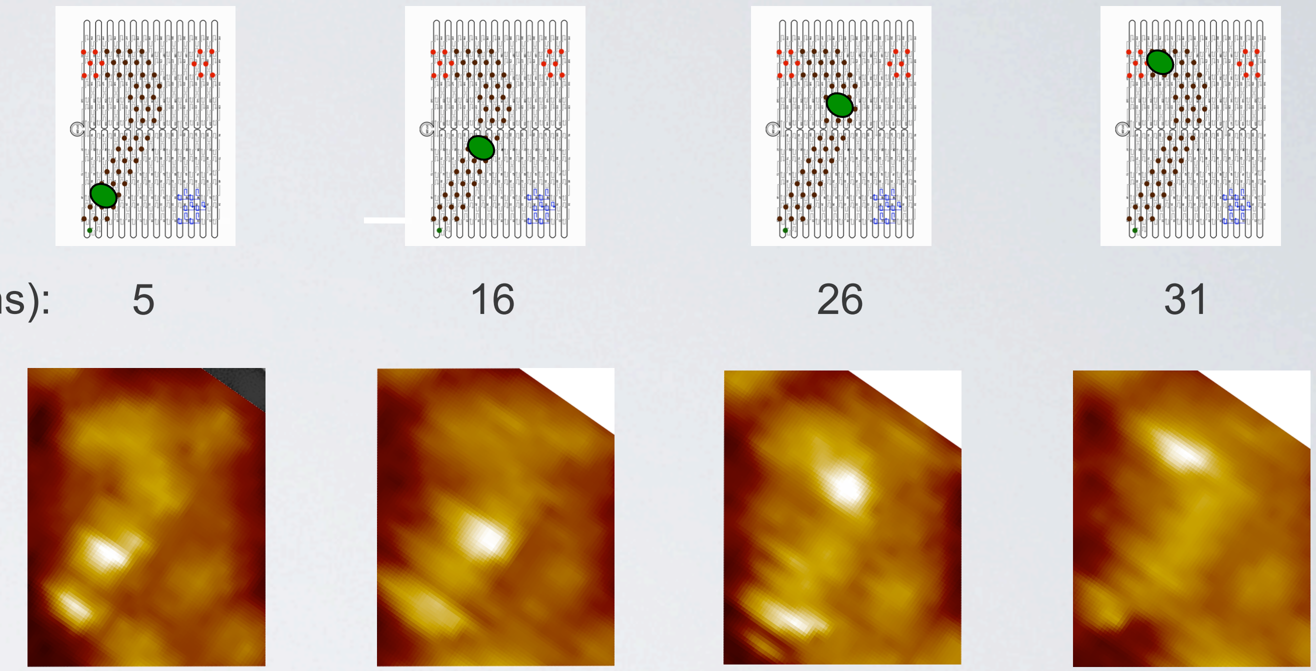
- Can we describe this growth process in abstract terms?
  - Conversion of amorphous fuel into a complicated structure (with a function!), movement of cells with respect to each other, robust to temperature and chemical changes in the environment, development works perfectly while cells are being “pushed around”, cellular differentiation is happening in a distributed manner. Fast. Autonomous.

- Universality of biology
- Can we engineer something of this complexity and size, but built on the molecular scale?

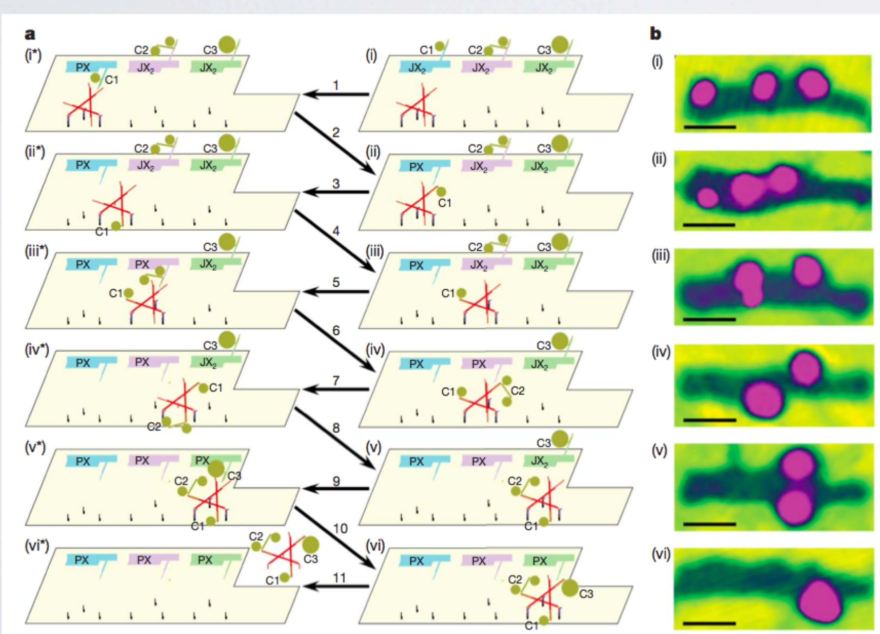
MOLECULAR MOTORS



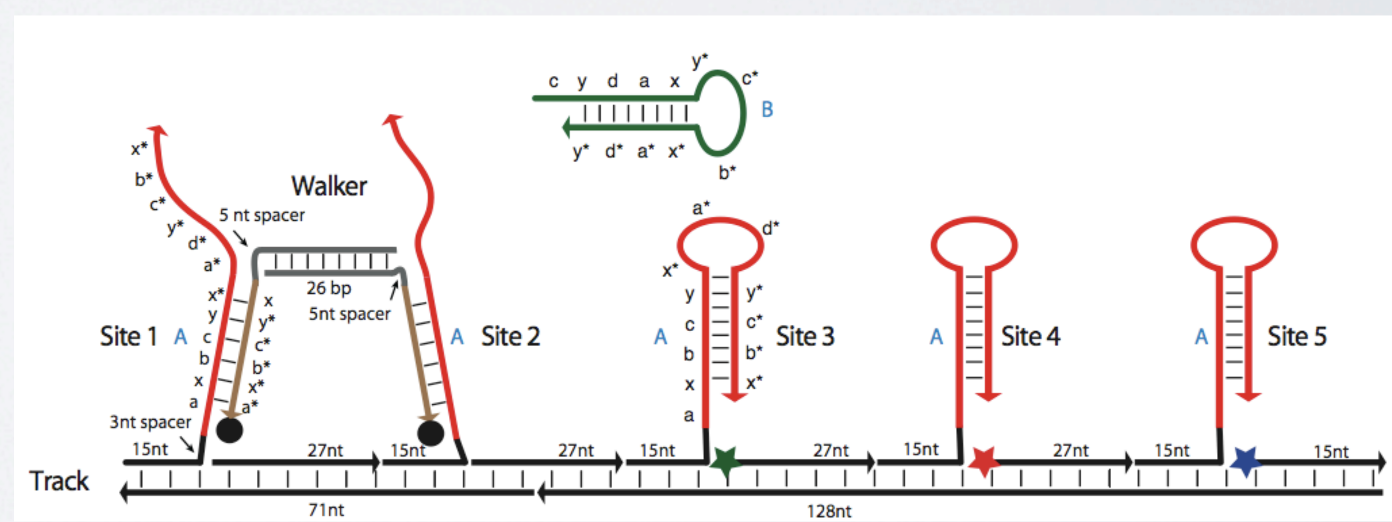
Yurke, Turberfield, Mills, Simmel, Neumann. Nature. 406:605-608. 2000



Lund, Manzo, Dabby, Michelotti, Johnson-Buck, Nangreave, Taylor, Pei, Stojanovic, Walter, Winfree, Yan. Molecular Robots Guided by Prescriptive Landscapes. Nature, 2010.

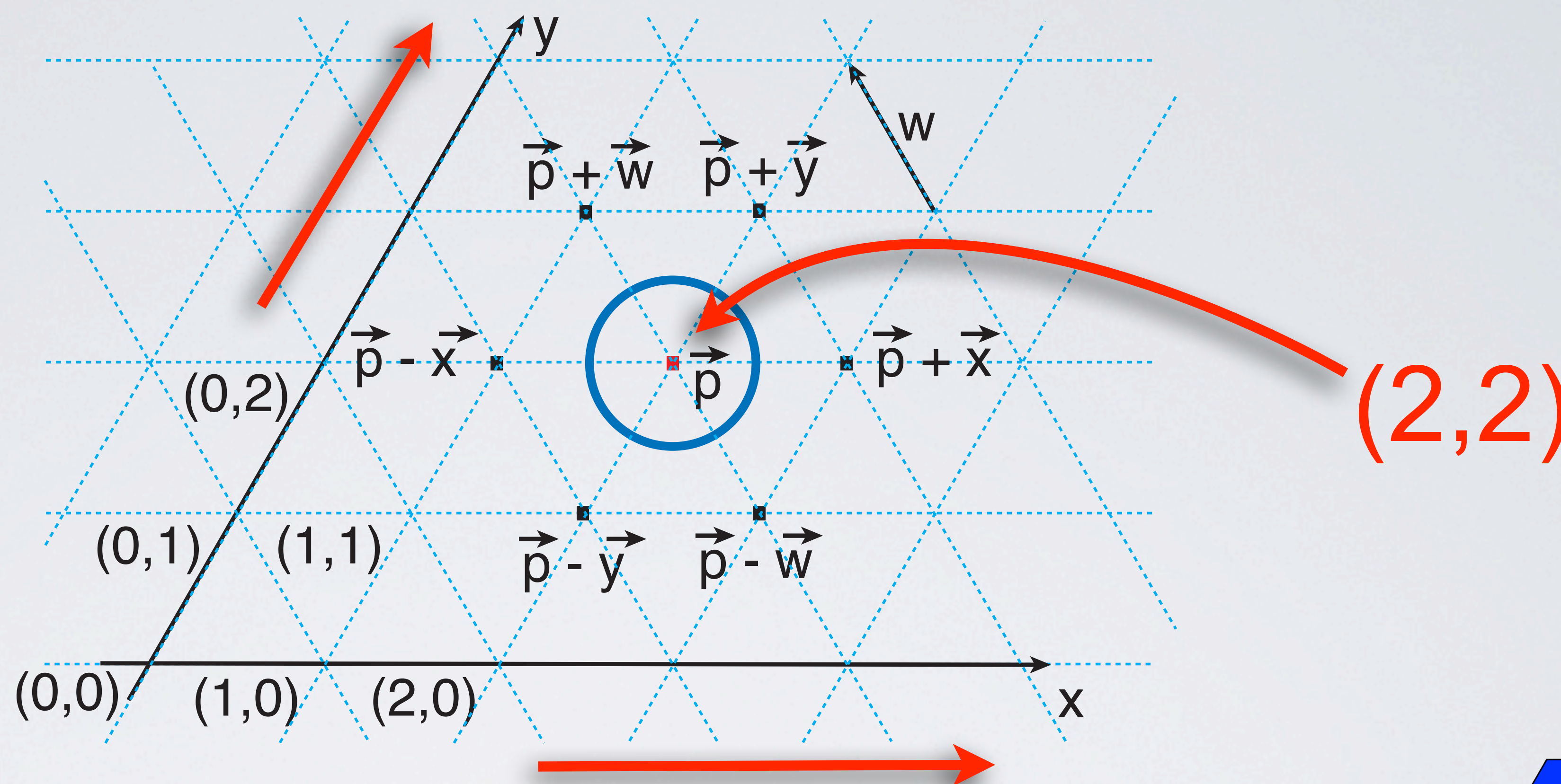


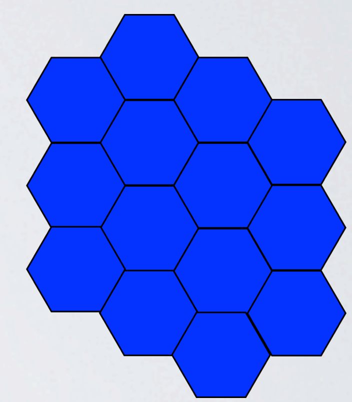
Gu, Chao, Xiao, Seeman. Nature. 2010



Yin, Choi, Calvert, Pierce, Nature. 451:318-322. 2008

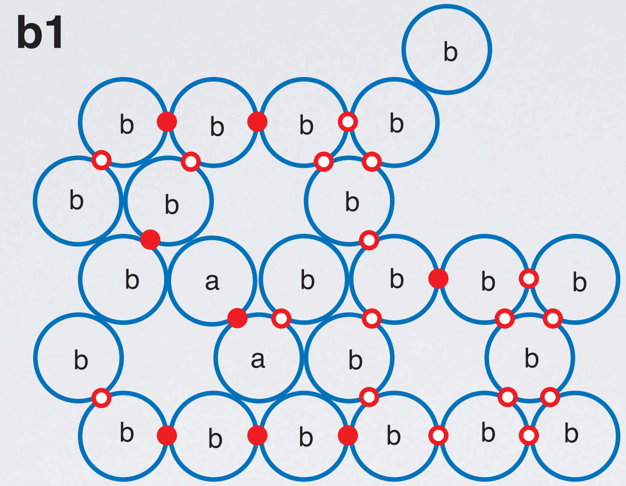
An active self-assembly model: nubots

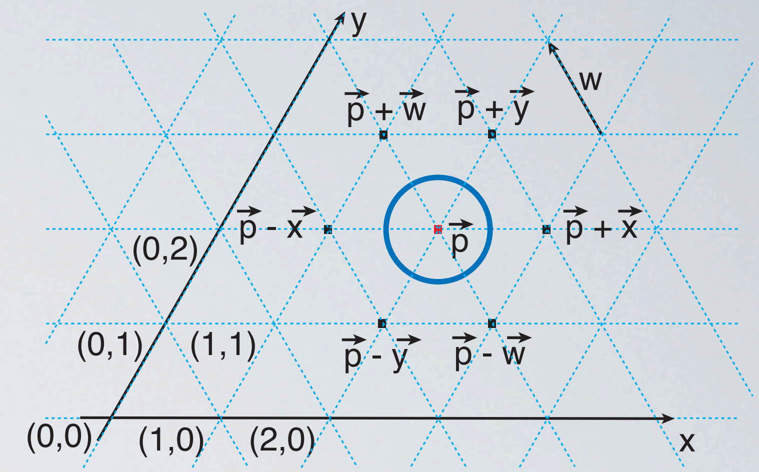




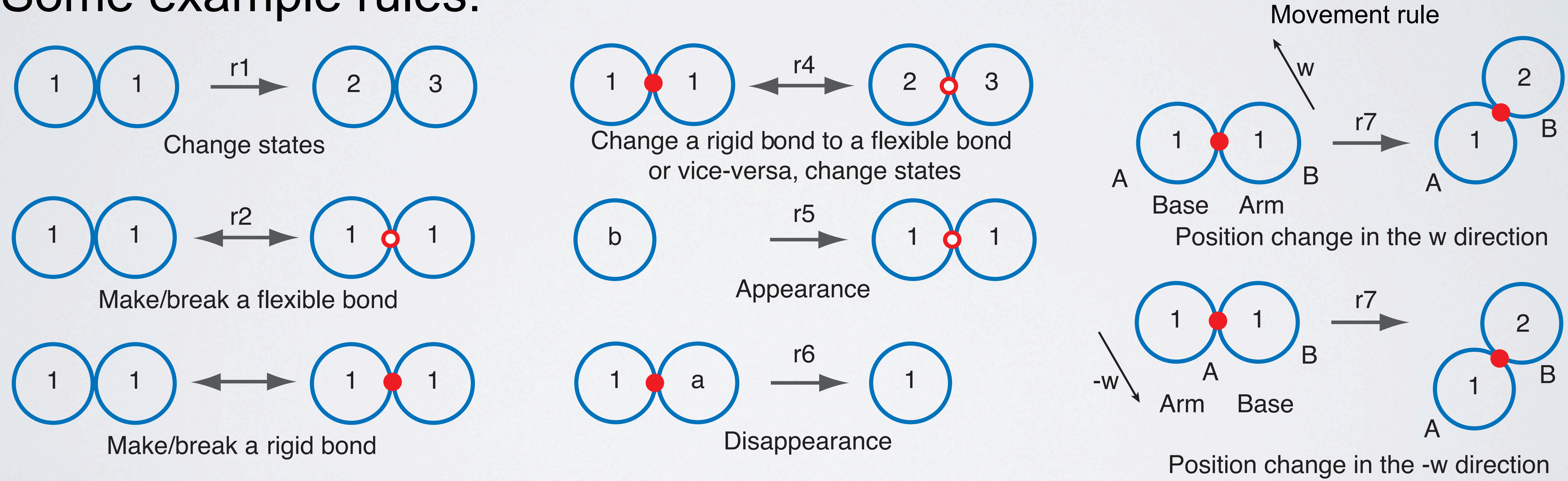
- 2D **triangular grid**
- **Monomers** have a **state** and a **position** on the grid
- The set of all possible states is finite (“monomers are simple”). Positions are pairs of integers.
- **Configuration**: finite set of monomers (i.e. monomer states and positions at some time instant)
- Configurations change over time using a finite set of **rules** (next slide)

An active self-assembly model





Some example rules:

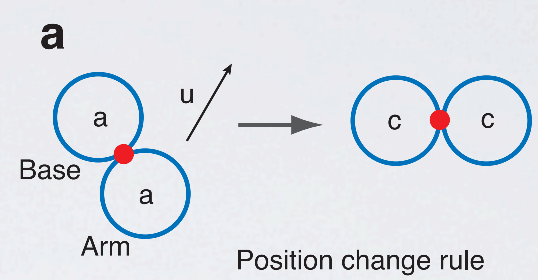


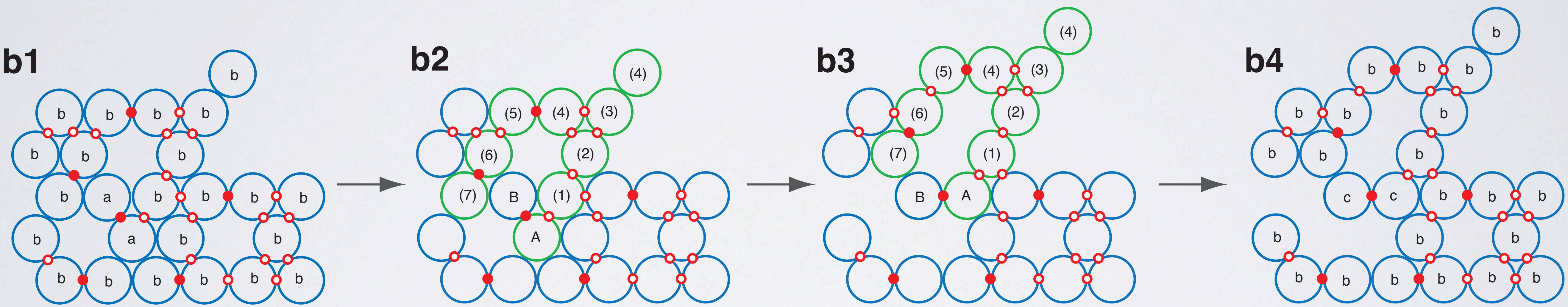
An active self-assembly model

Golden movement rule

Movement is **not** permitted to break **bonds**: A **rigid** bond between 2 monomers breaks if their relative position changes. A **flexible** bond between 2 monomers breaks if one of them moves so that they are no longer adjacent.

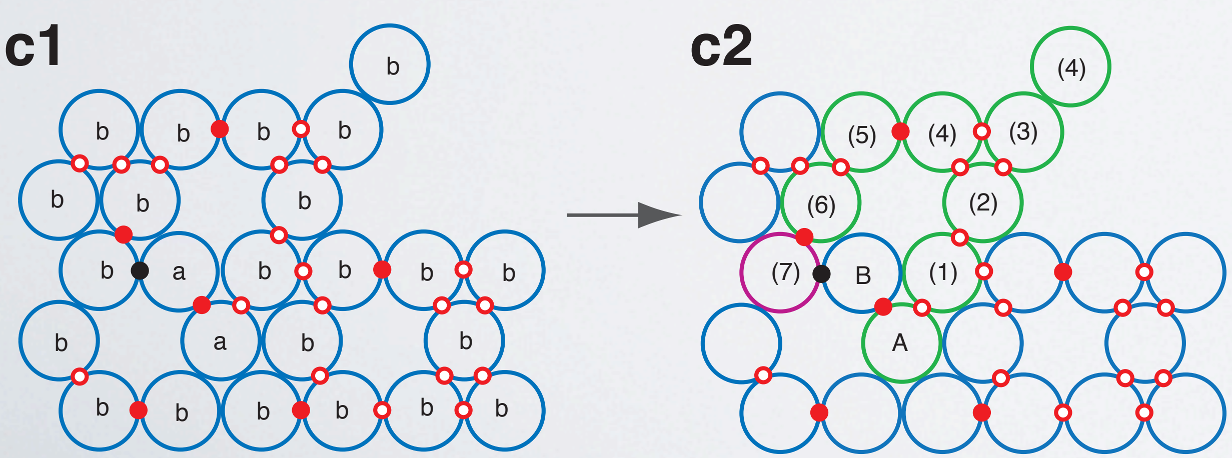
Movement rule example:





Current configuration

Successful application of movement rule



Current configuration

Movable set empty

Failure to apply movement rule (movement would break flexible/rigid bonds)

An active self-assembly model

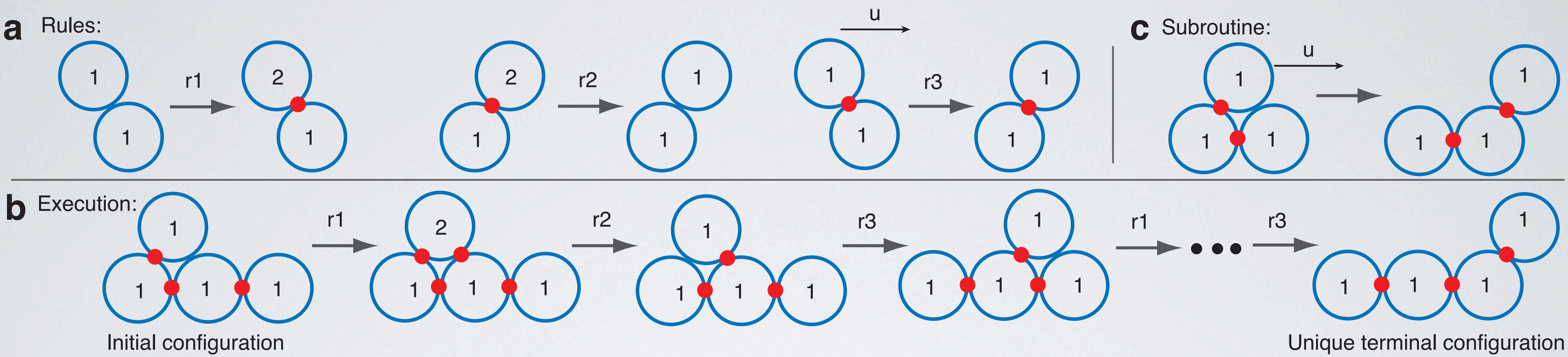
Rules are applied over time in an *asynchronous* fashion

System evolves as a continuous time Markov process

For a given monomer, the time, in seconds, until an applicable rule is applied (or “fires”) is an exponentially distributed random variable with mean 1

- For a given monomer, if a rule is applicable it fires in expected time 1
- If there are n monomers with applicable rules, then the expected time until *some* rule fires is $1/n$
- If there are n monomers with applicable rules, that can be applied independently, then the expected time for *all* rules to fire is $O(\log n)$

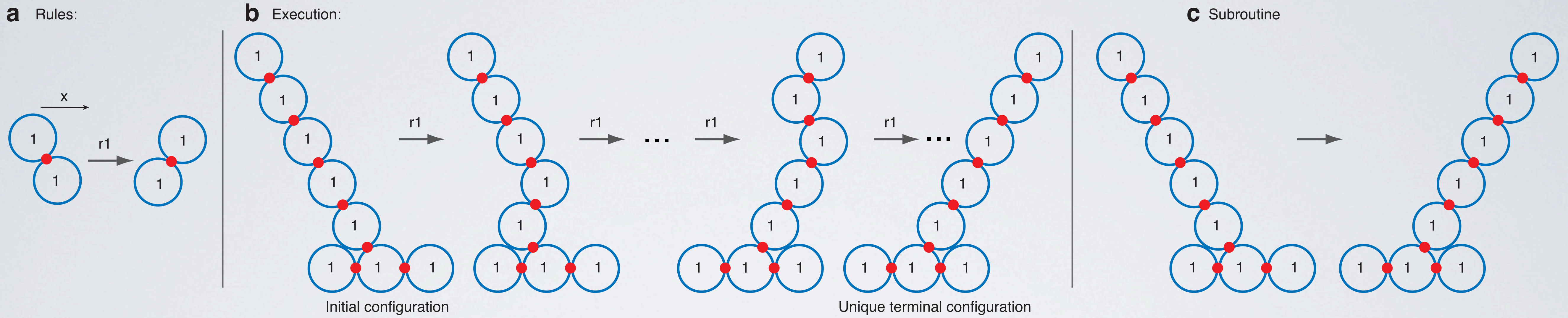
EXAMPLE: WALKER



Rules: $r1 = (1, 1, \text{null}, -\vec{w}) \rightarrow (2, 1, \text{rigid}, -\vec{w})$, $r2 = (1, 2, \text{rigid}, \vec{y}) \rightarrow (1, 1, \text{null}, \vec{y})$, $r3 = (1, 1, \text{rigid}, \vec{w}) \rightarrow (1, 1, \text{rigid}, \vec{y})$.

Expected time = 3(track length)

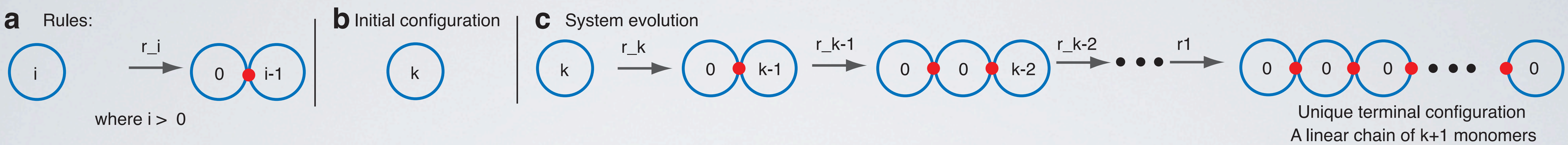
EXAMPLE: ROTATING ARM



$$\text{Rule: } r1 = (1, 1, \text{rigid}, \vec{w}) \rightarrow (1, 1, \text{rigid}, \vec{y}).$$

Expected time = O(log (arm length))

EXAMPLE: GROW A LINE



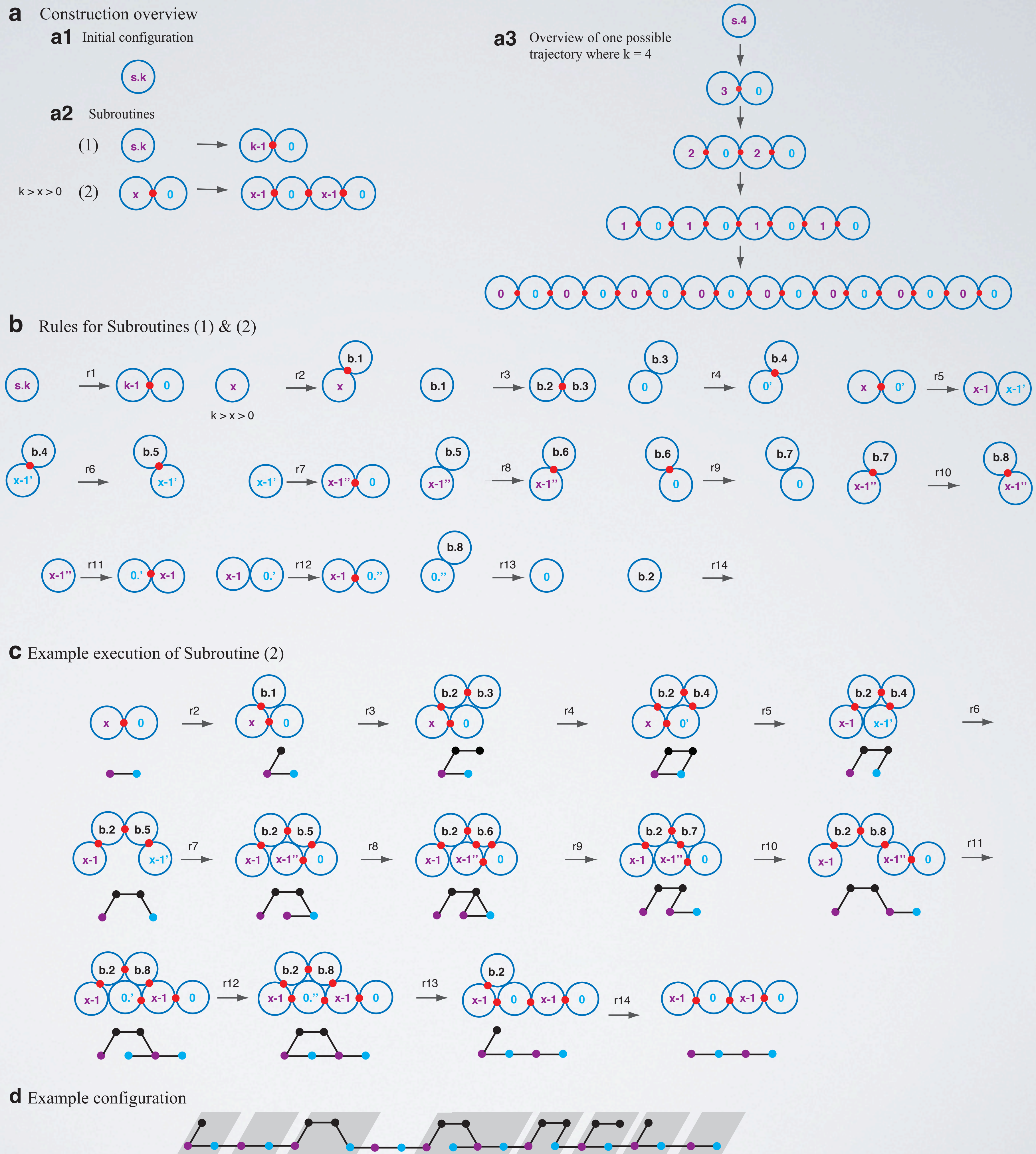
Rule set for length $k+1$ line: $\{r_i \mid r_i = (i, \text{empty}, \text{null}, \vec{x}) \rightarrow (0, i-1, \text{rigid}, \vec{x}),\ \text{where } k \geq i > 0\}$.

How much time?

Expected time = total line length - 1 = k

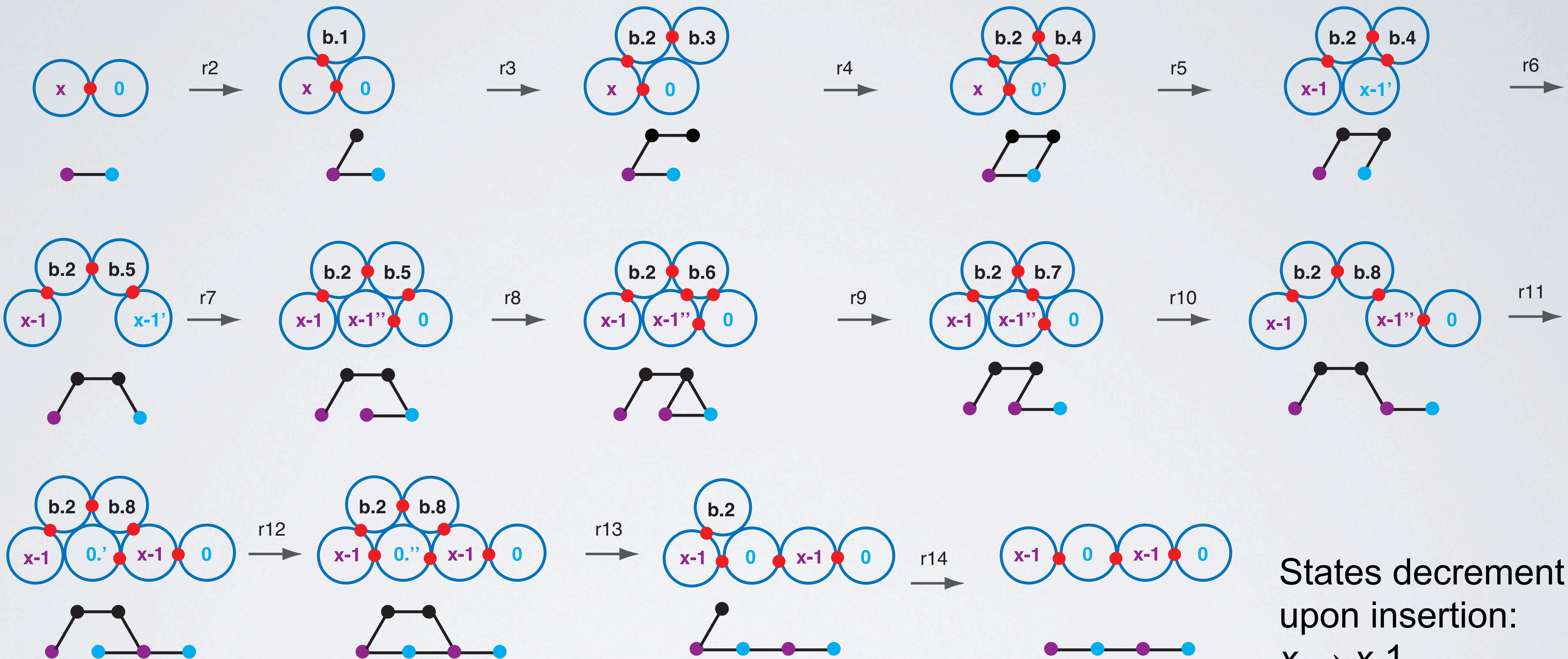
How to do better?

FAST GROWTH OF A LINE



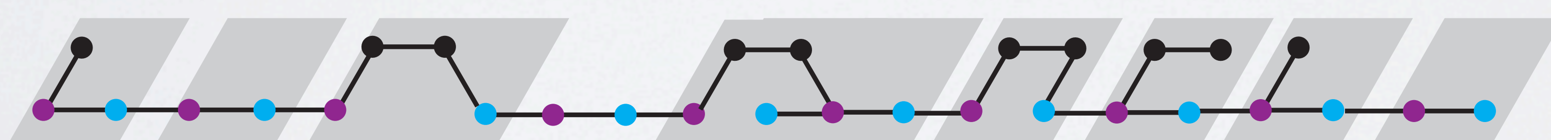
FAST GROWTH OF A LINE

c Example execution of Subroutine (2)



States decrement upon insertion:
$x \rightarrow x$-1,
$x \in \{k, k$-1,...,2, 1$\}$

d Example configuration

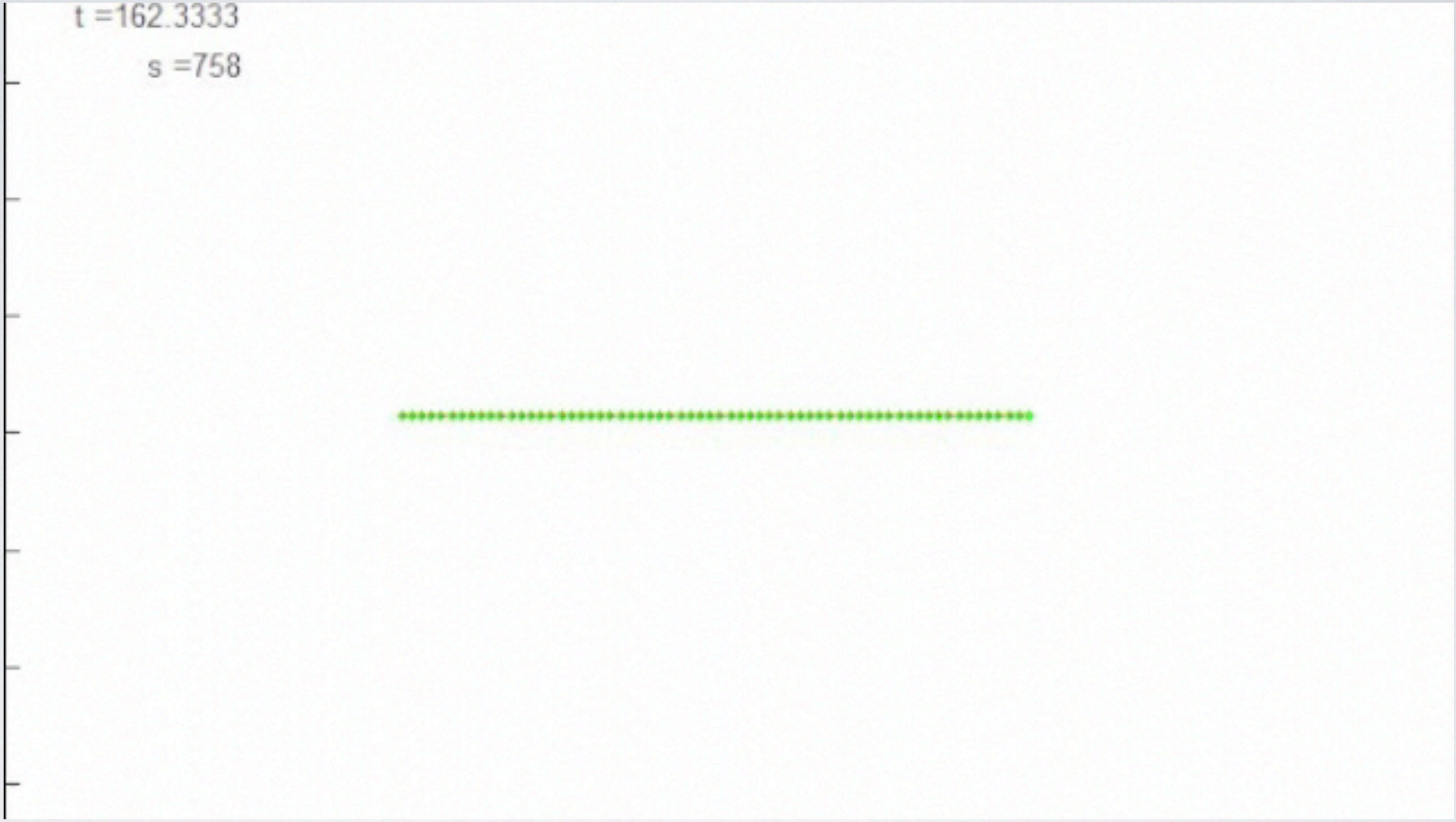


Highly parallel! Expected time = $O(\log n)$ to make a line of length $n = 2^k$

FAST GROWTH OF A LINE

t = time

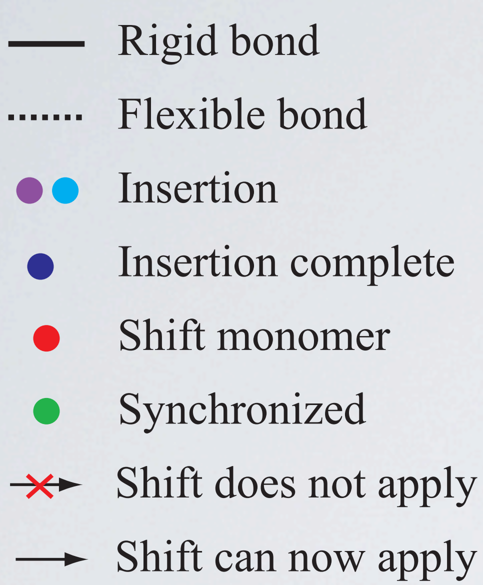
s = rule applications (steps)

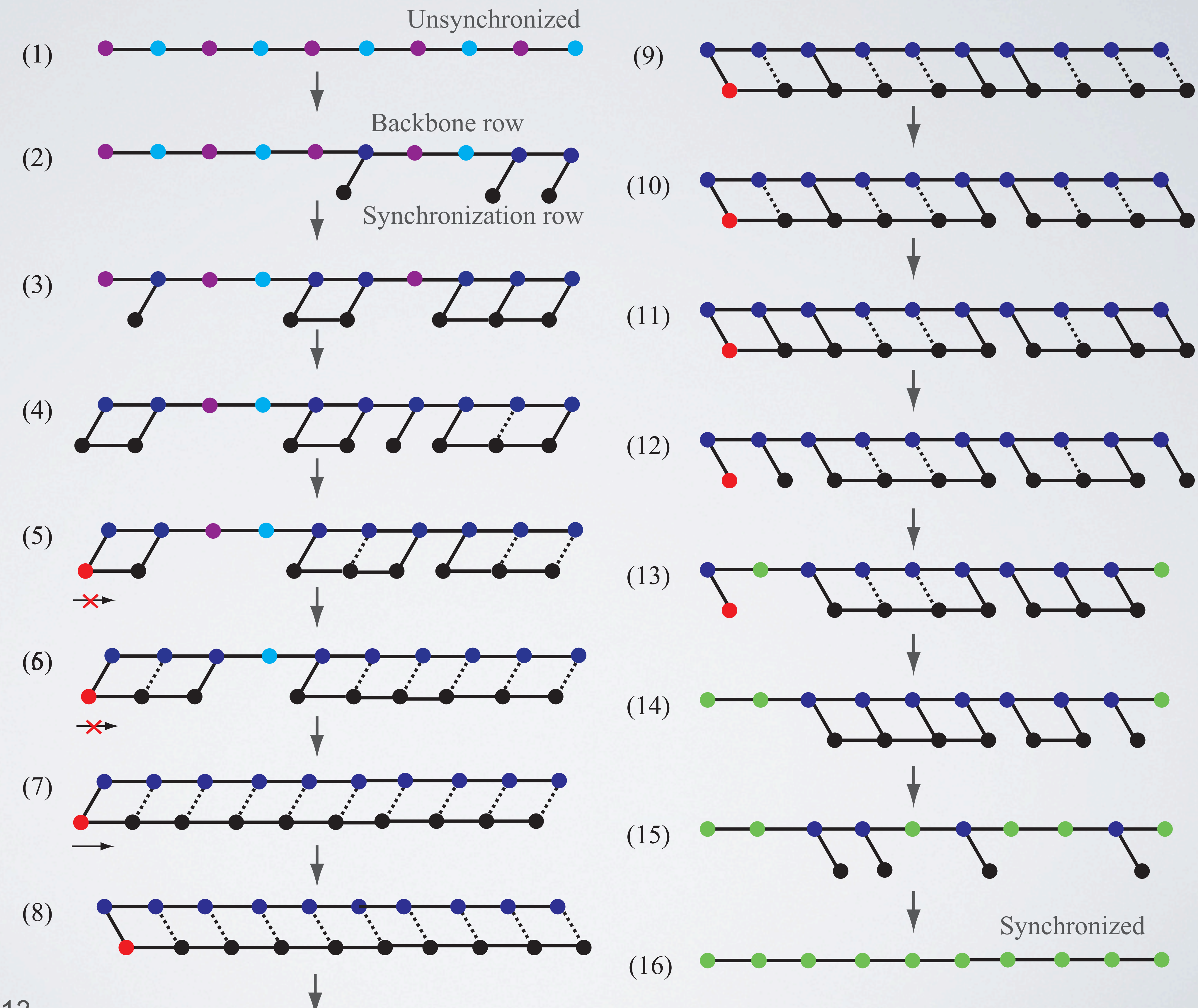


Growth of a line of length $2^6 = 64$

FAST SYNCHRONIZATION

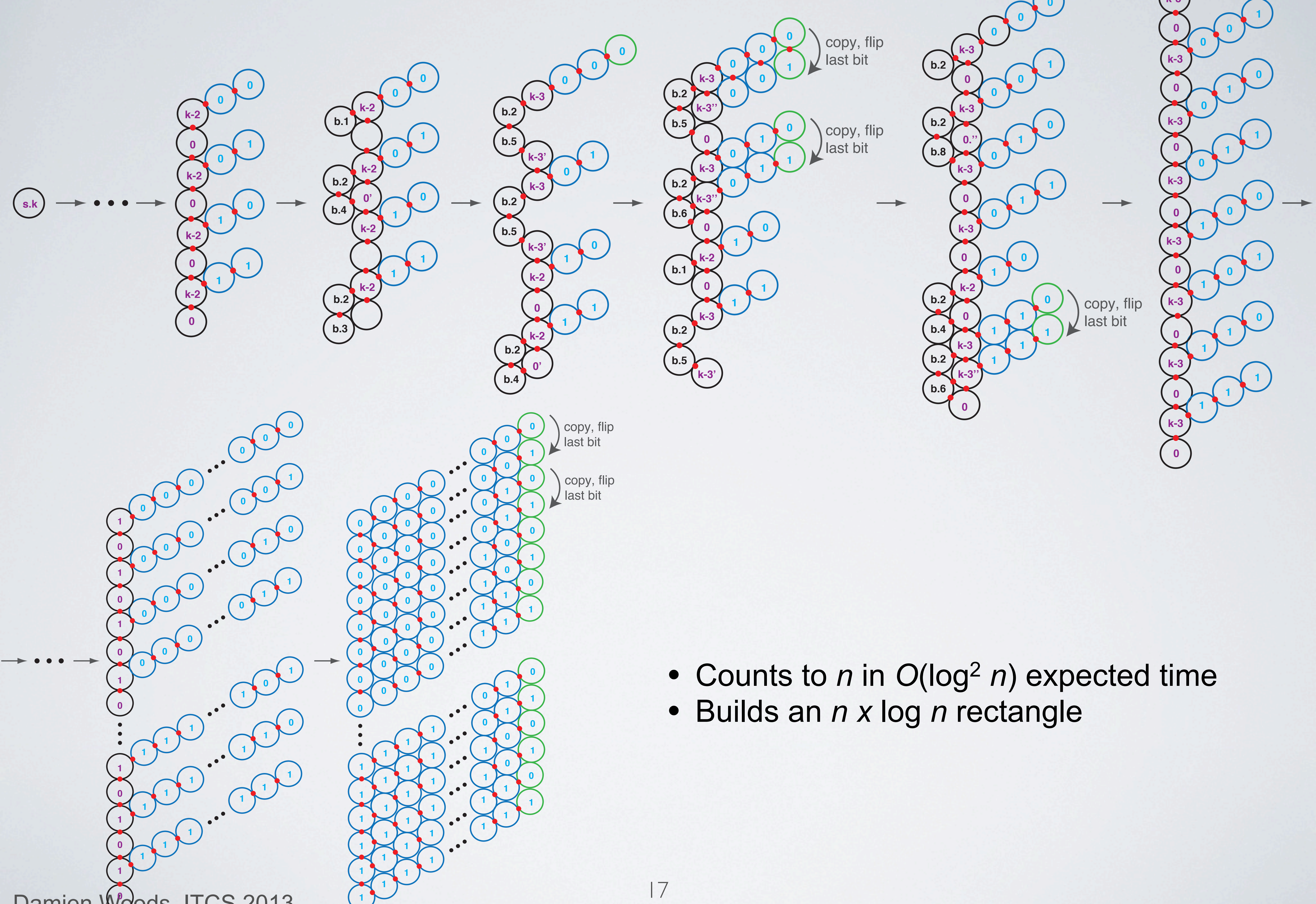
How to detect when the line is finished growing?





Expected time $O(\log n)$ for a line of length n.

FAST GROWTH OF A COUNTER

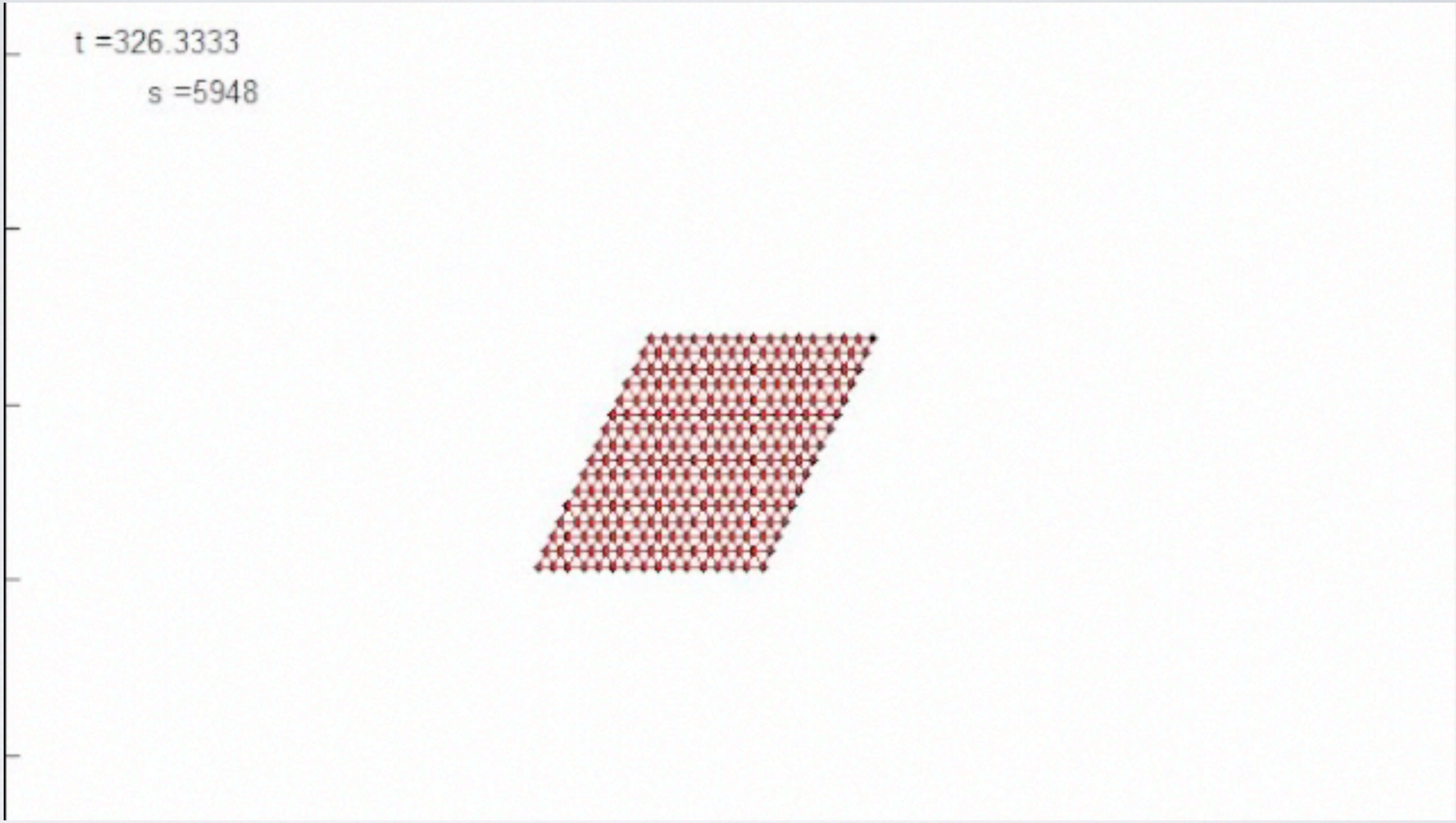


- Counts to n in $O(\log^2 n)$ expected time
- Builds an n x $\log n$ rectangle

FAST GROWTH OF A SQUARE

time

rule applications (steps)



n x *n* square in *O*(log *n*) time!
This example: 16 x 16 square

WHAT CAN ACTIVE SELF-ASSEMBLY DO?

We've exhibited a selection of specific shapes and behaviors

We've seen certain shapes can be built exponentially fast

What is the system capable of in general?

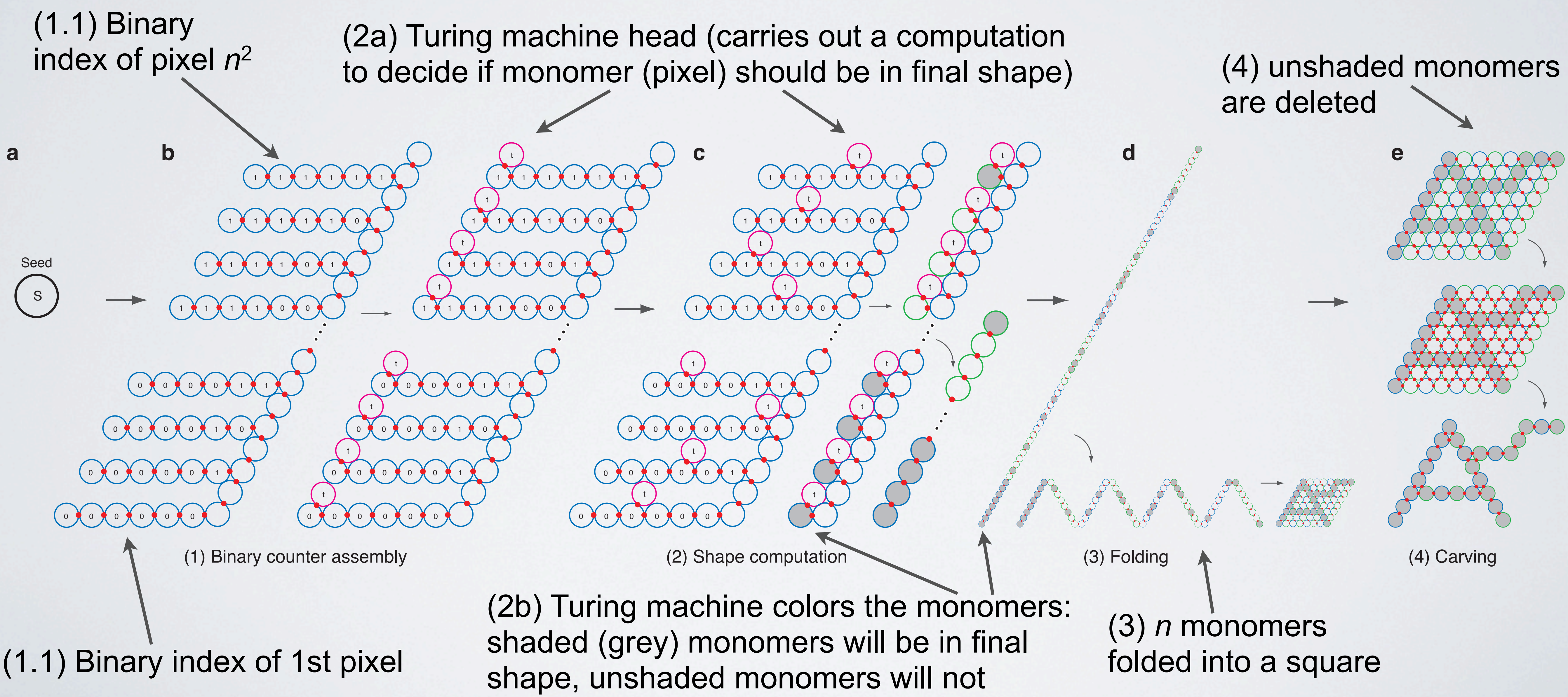
Can it make more general/complicated shapes quickly?

How complicated can a such shape be?

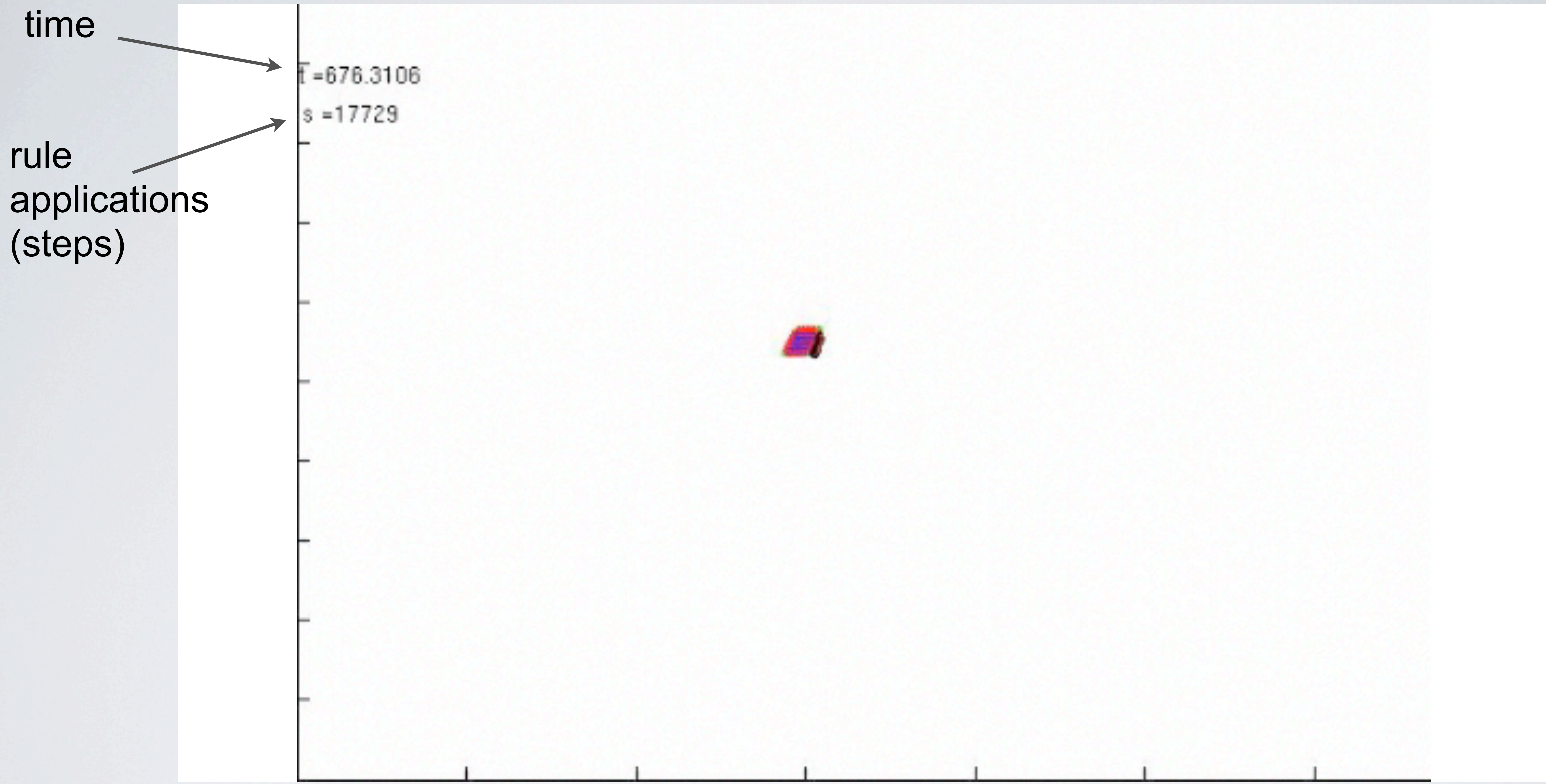
RESULT 1: EFFICIENTLY COMPUTABLE SHAPES

Result 1: An arbitrary connected computable 2D shape of size ≤ n x n can be constructed:

- in time $O(\log^2 n + t(|n|))$ and using $O(s + \log n)$ states,
- where $t(|n|)$ is the time for a program size s Turing machine to compute, given the pixel position index of as a length $|n|=O(\log n)$, whether the pixel is present in the shape.



COMPUTABLE SHAPES



Simulator by Scott Goodfriend

EFFICIENTLY COMPUTABLE PATTERNS: CAN WE DO BETTER FOR SIMPLER PATTERNS?

- In Result 1, the Turing machine computation time $t(|n|)$ is a bottleneck to fast (polylog(n)) computation. There is nothing we can do about this.
  - So let's restrict attention to shapes with polylog(n) time computable pixels.
- In Result 1, we had "non-monotone" growth (e.g., long length folded into a square, & lots of extra space used by Turing machines):
  - Can we restrict growth to be contained in an n x n "womb" without affecting computational power?
- In Result 1, we used synchronization over long distances:
  - Can we compute without it?

Can we build large structures?

http://www.djibnet.com/photo/2004/city-of-lights-358919966.html

Yes!

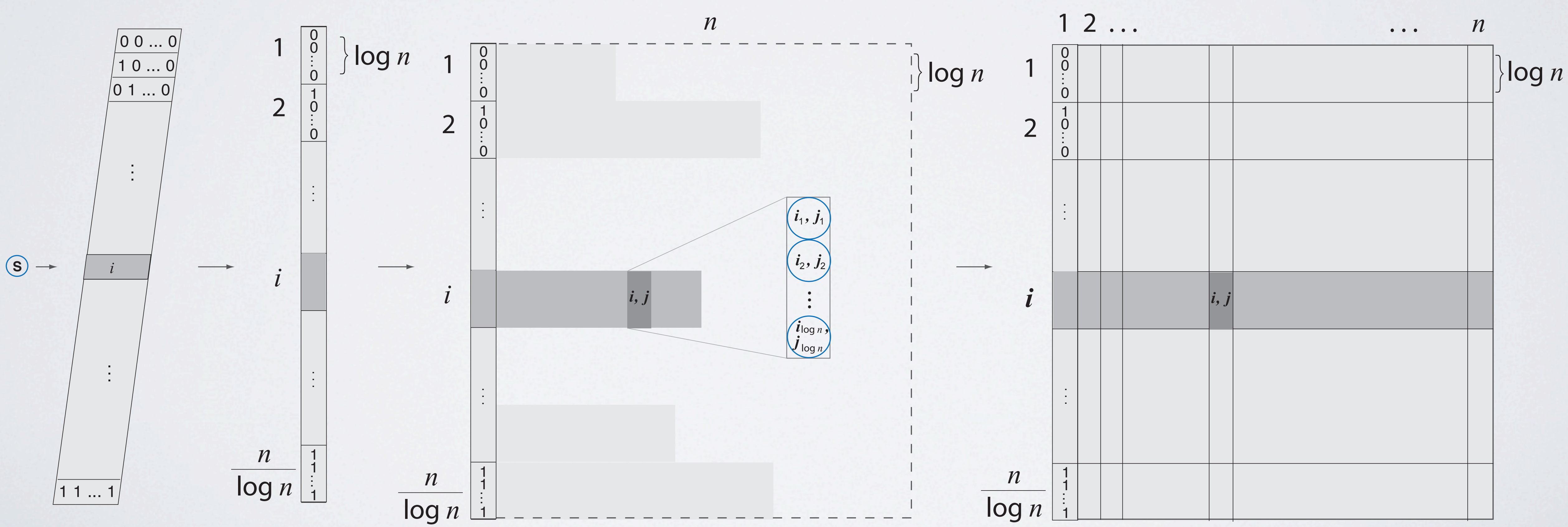
Quickly? In-place?
Without synchronization?
While being pushed around?

http://www.djibnet.com/photo/2004/city-of-

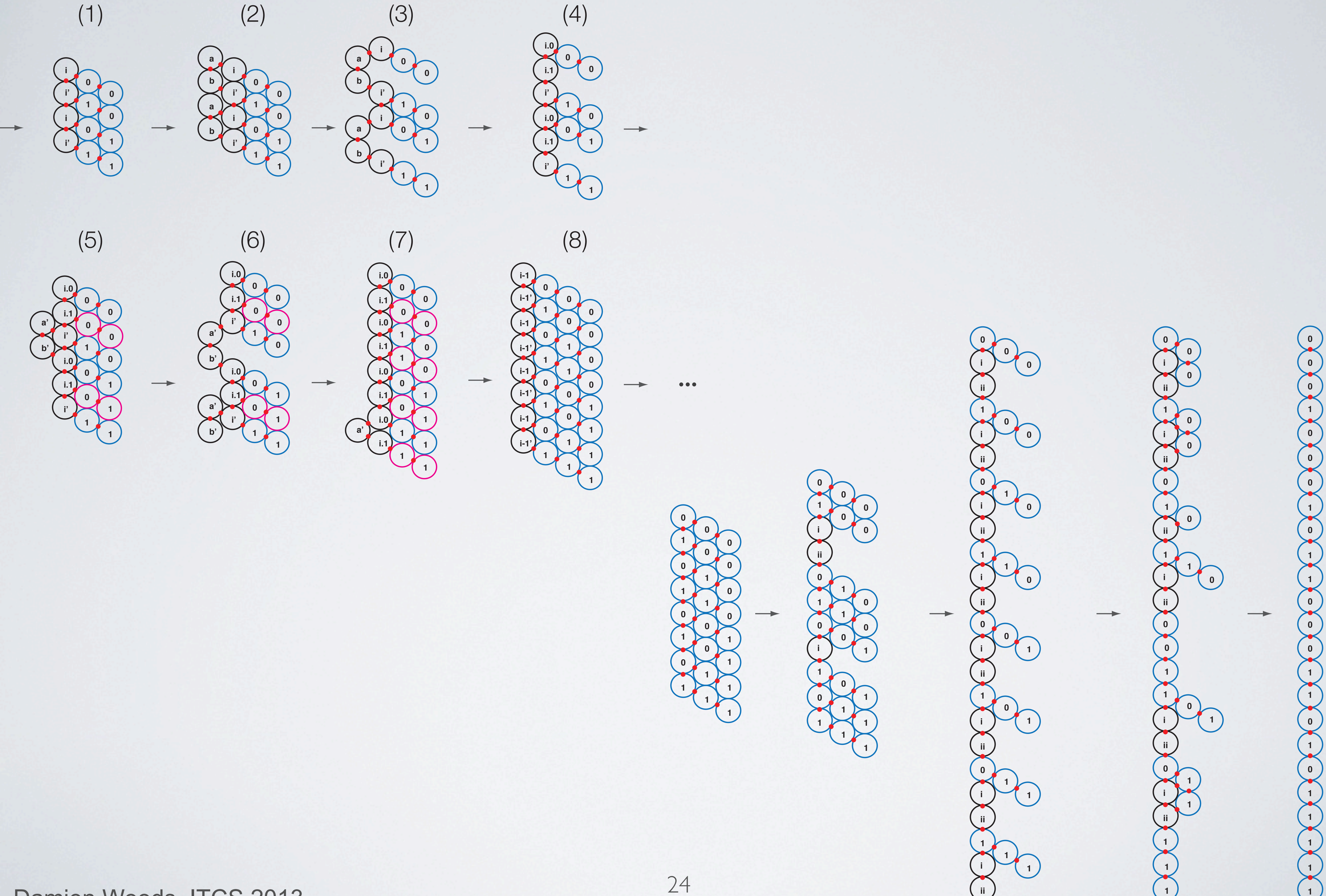
RESULT 2: EFFICIENTLY COMPUTABLE PATTERNS

Result 2: An arbitrary computable 2D *pattern* of size ≤ *n* x *n,* whose pixels are computable in (polynomial) time O($\log^l$ n*)* and (linear) space O*(log n),* can be constructed:

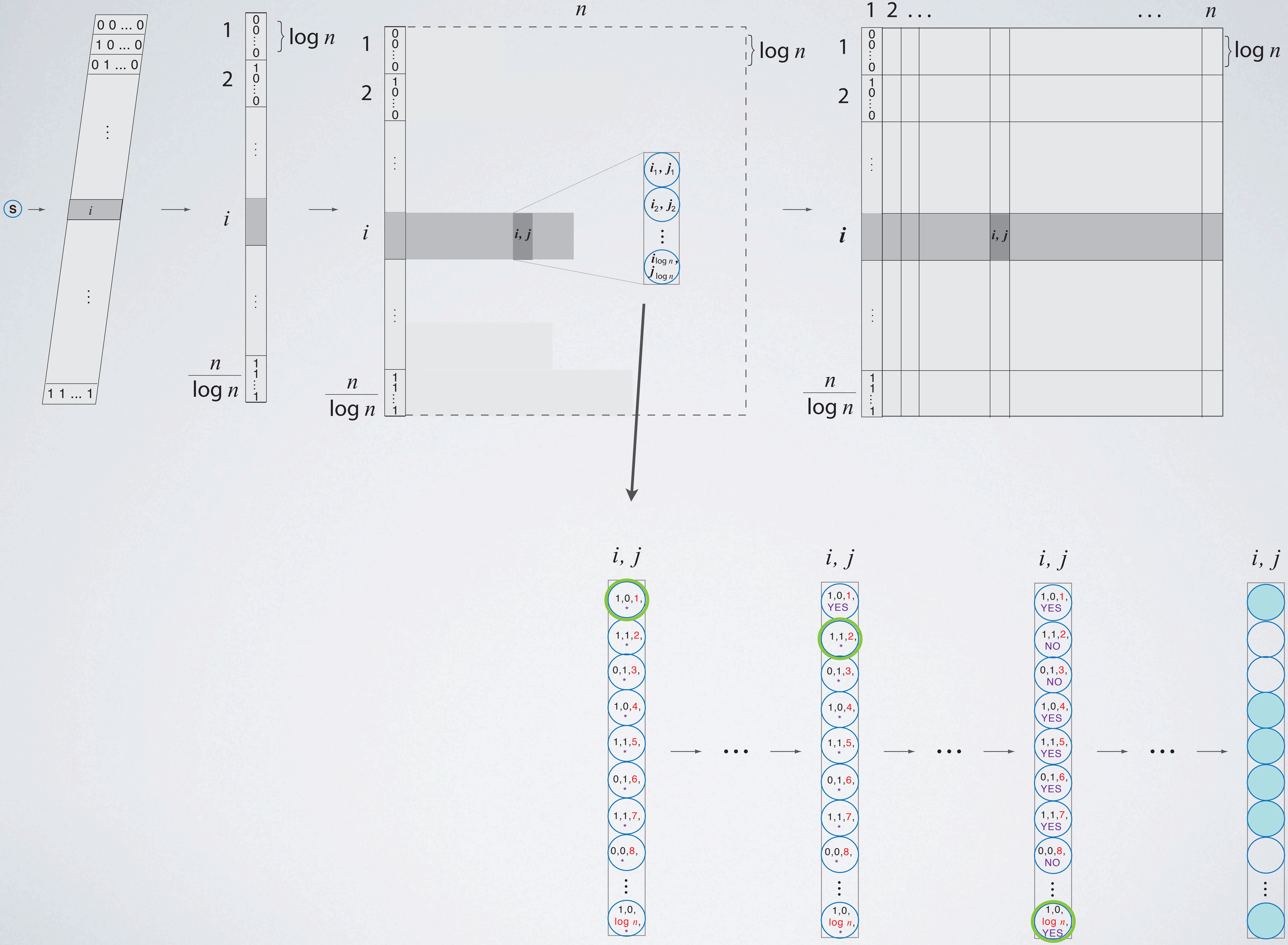
- in time O($\log^{l+1}$ n*)*,
- using *O*(*s* + log *n*) states where *s* is the program size of a Turing machine to compute, given the position index of a pixel, whether the pixel should be present in the shape,
- this can be done in-place (monotone growth) in a region of size *n* x *n,*
- and without using long-range synchronization.



RESULT 2: EFFICIENTLY COMPUTABLE PATTERNS



RESULT 2: EFFICIENTLY COMPUTABLE PATTERNS



SUMMARY

- Using active self assembly we can:
  - Model simple walkers and other motors
  - *Quickly* grow exponentially large computable shapes
  - Quickly grow exponentially large patterns in place (without synchronization, and while being “pushed around”)
- The addition of the movement rule to a cellular automaton is sufficient for exponential growth
- Future work:
  - model other growth processes: dynamic processes, rearrangement, persistence length, tensile strength, movement, signaling with/without perfect synchronization
  - General time analysis tools?
  - Computational complexity: Are polylog time nubots = NC?
- Future work: implement the model in the lab!

Full paper:
D. Woods, H.L. Chen, S. Goodfriend, N. Dabby, E. Winfree, P. Yin.
Active Self-Assembly of Algorithmic Shapes and Patterns in Polylogarithmic Time.

THANKS!

NSF
Molecular Programming Project (0832824)

NSF
CCF-1219274
CCF-1162589

Caltech Center for Biological Circuit Design

Full paper:
D. Woods, H.L. Chen, S. Goodfriend, N. Dabby, E. Winfree, P. Yin.
Active Self-Assembly of Algorithmic Shapes and Patterns in Polylogarithmic Time.

FIN