Structural DNA nanotechnology

a.k.a. DNA carpentry

a.k.a. DNA self-assembly

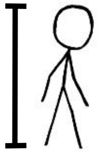
slides © 2021, David Doty

ECS 232: Theory of Molecular Computation, UC Davis

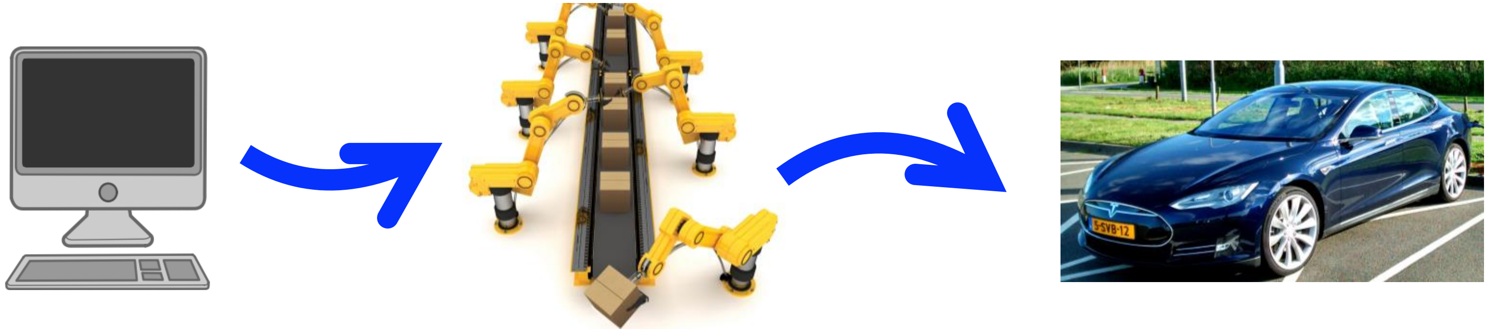
Ljubljana Marshes Wheel. 5k years old

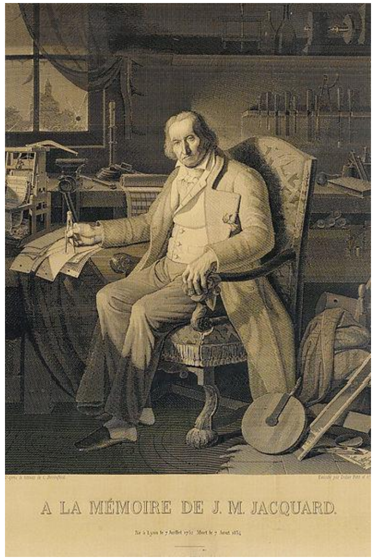
Building things

Newgrange, Ireland. 5.2k years old

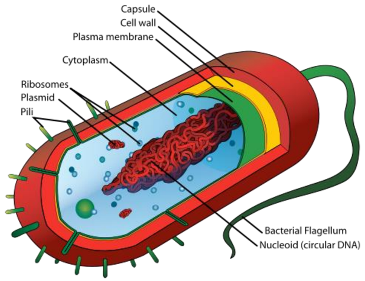
Building things by hand: use tools! Great for scale of $10^{\pm 2}$ × 

Building tools that build things: specify target object with a computer program





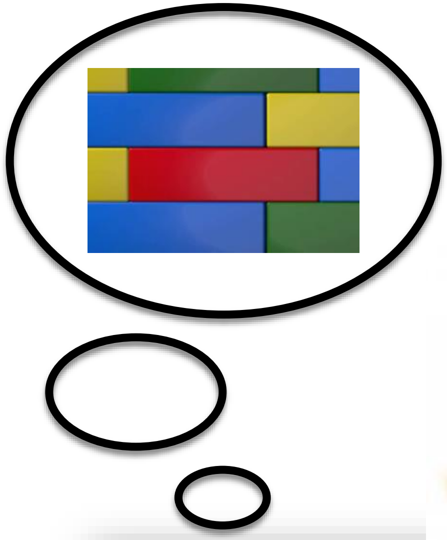
Programming **things to build themselves**: for building in small wet places where our hands or tools can't reach



Mariana Ruiz Villarreal

[slides credit: Damien Woods]

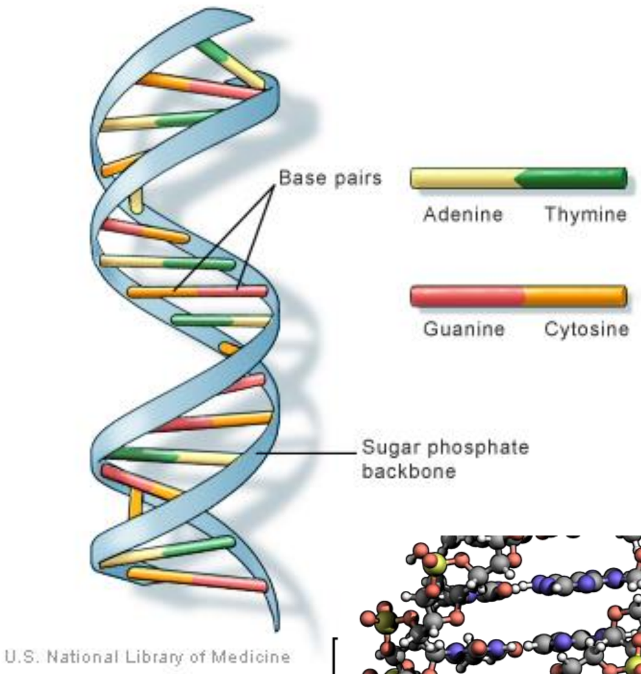
Things that build themselves

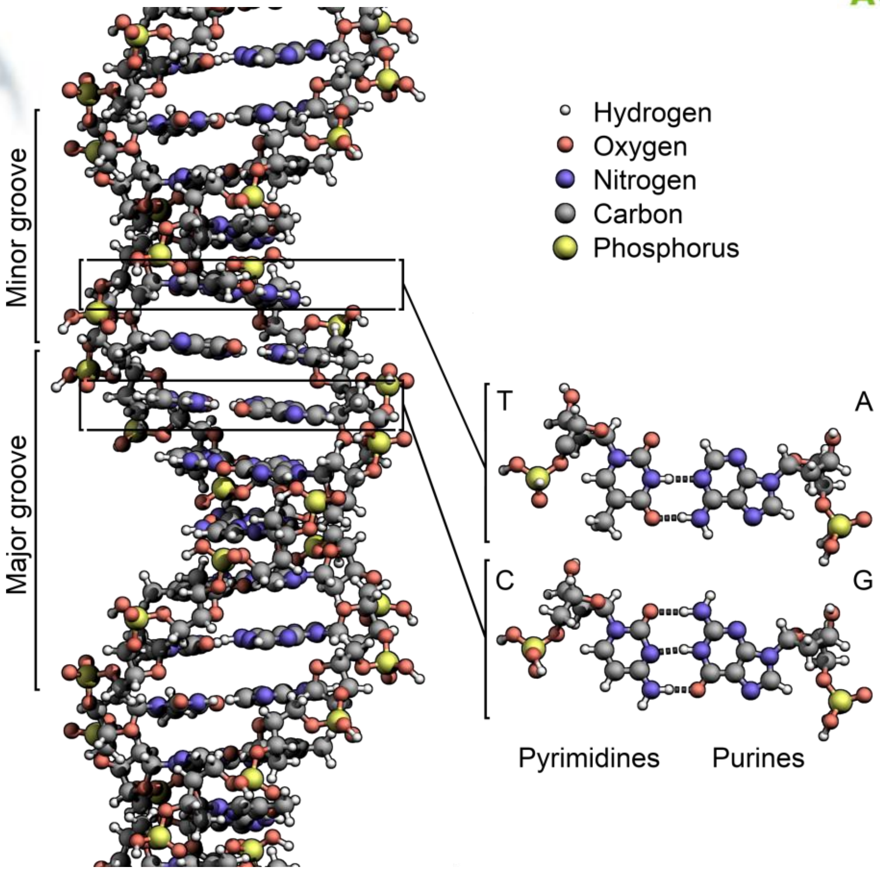


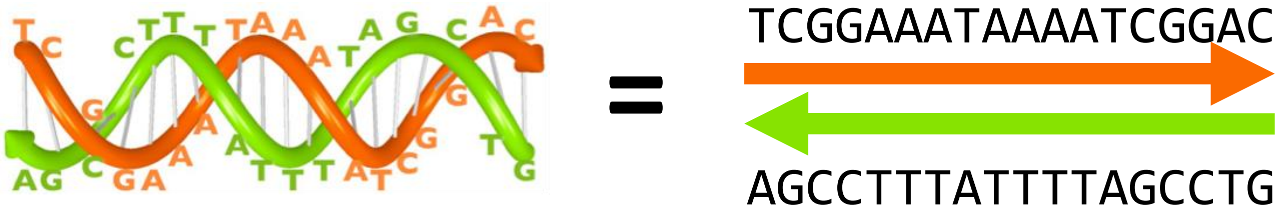
Our topic: self-assembling molecules that compute as they build themselves

[slides credit: Damien Woods]

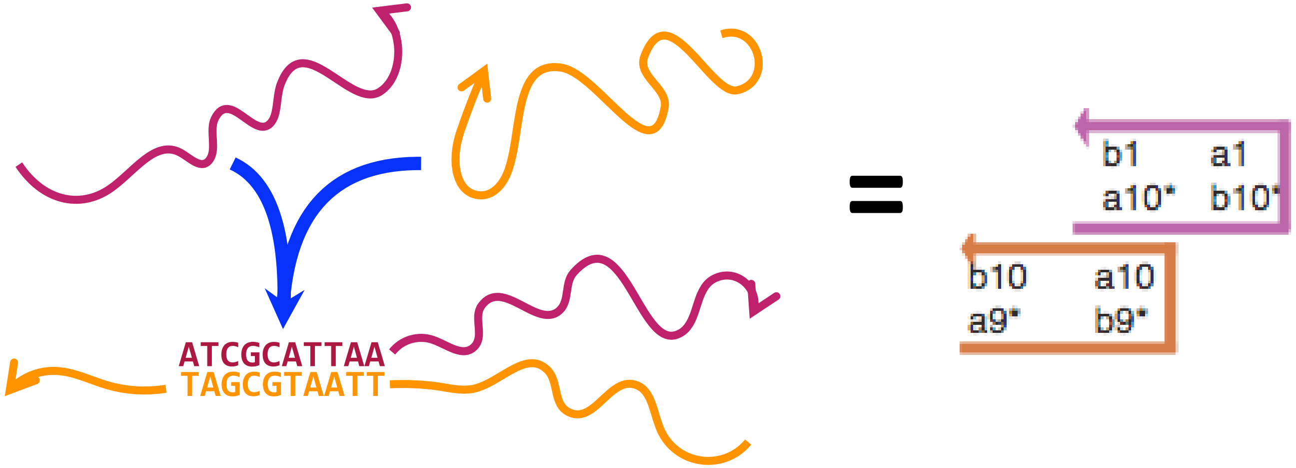
DNA as a building material



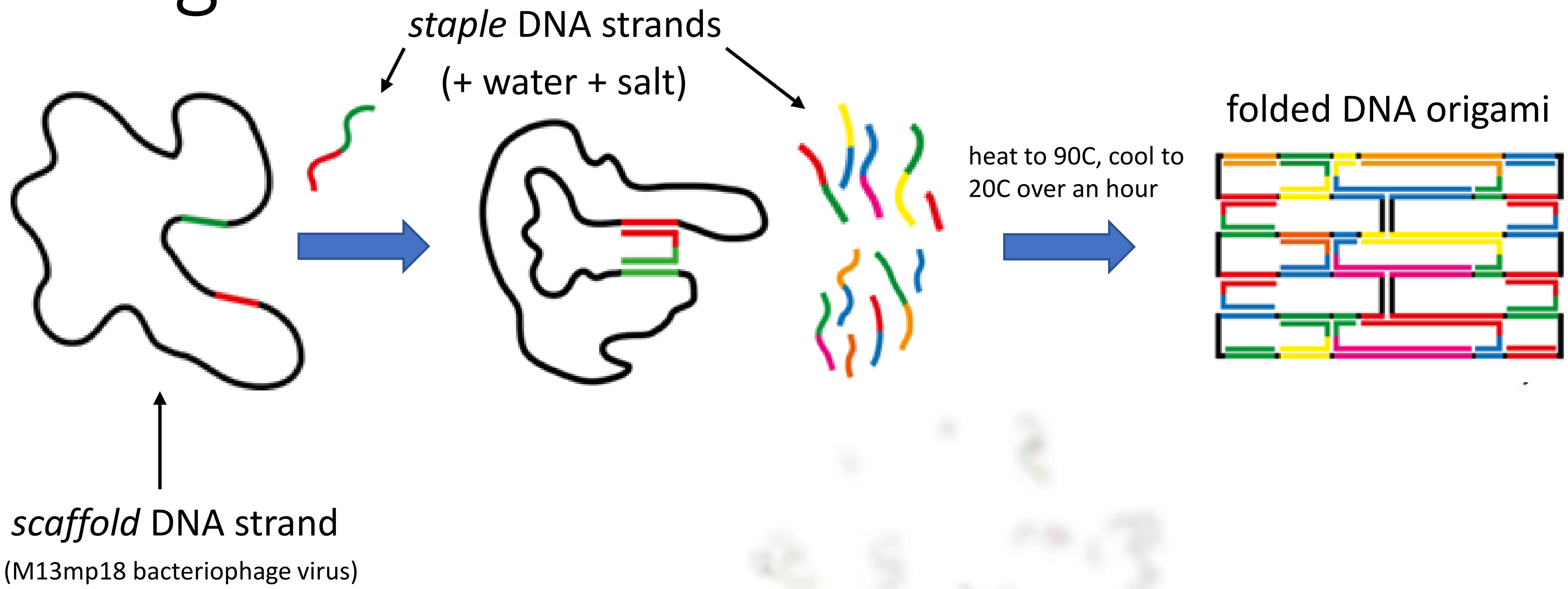




DNA strands bind even if only *part* of strands are complementary:



DNA origami



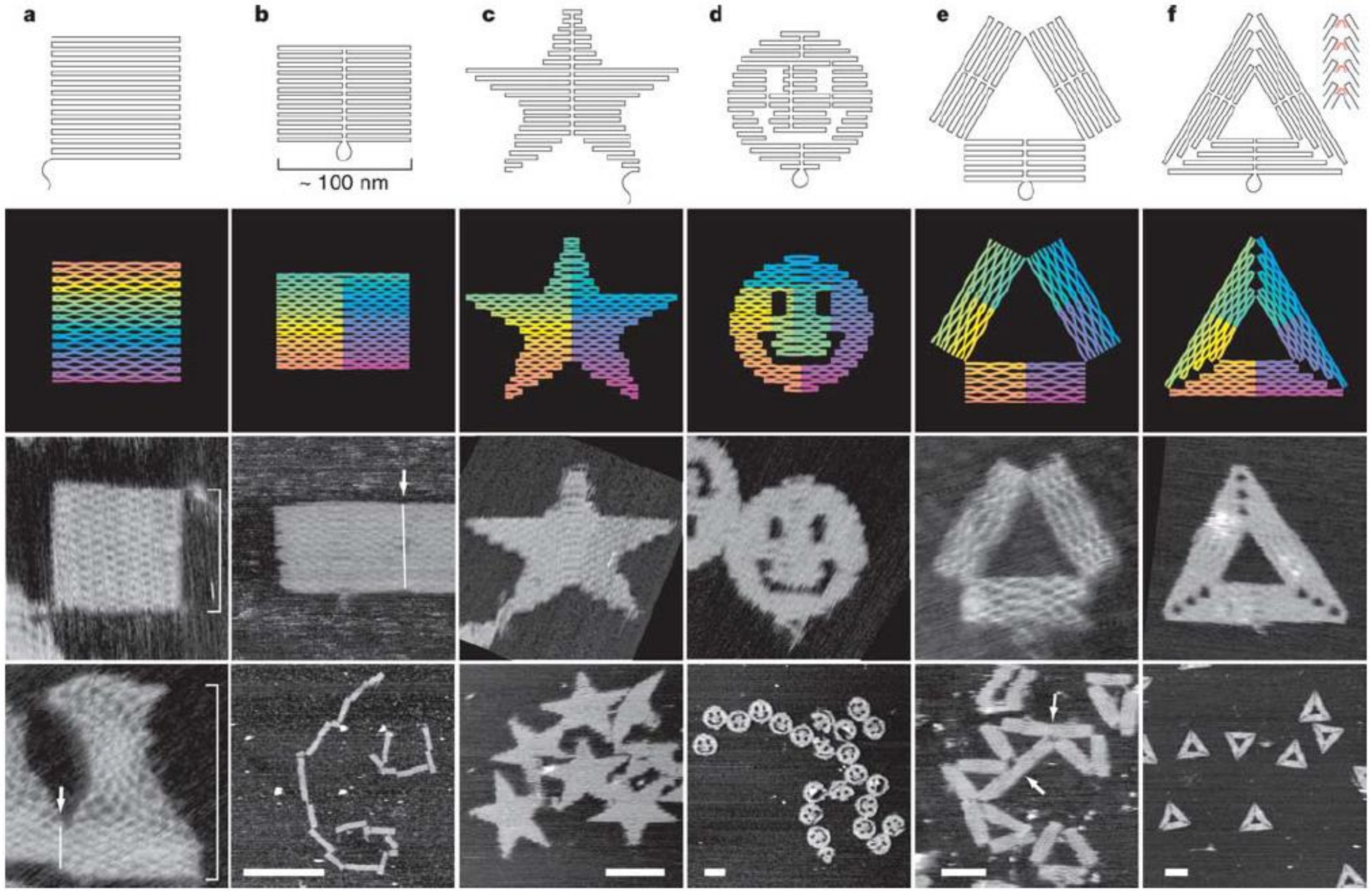
Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006

© http://openwetware.org/wiki/Biomod/2014/Design

© Shawn Douglas

DNA origami

Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006

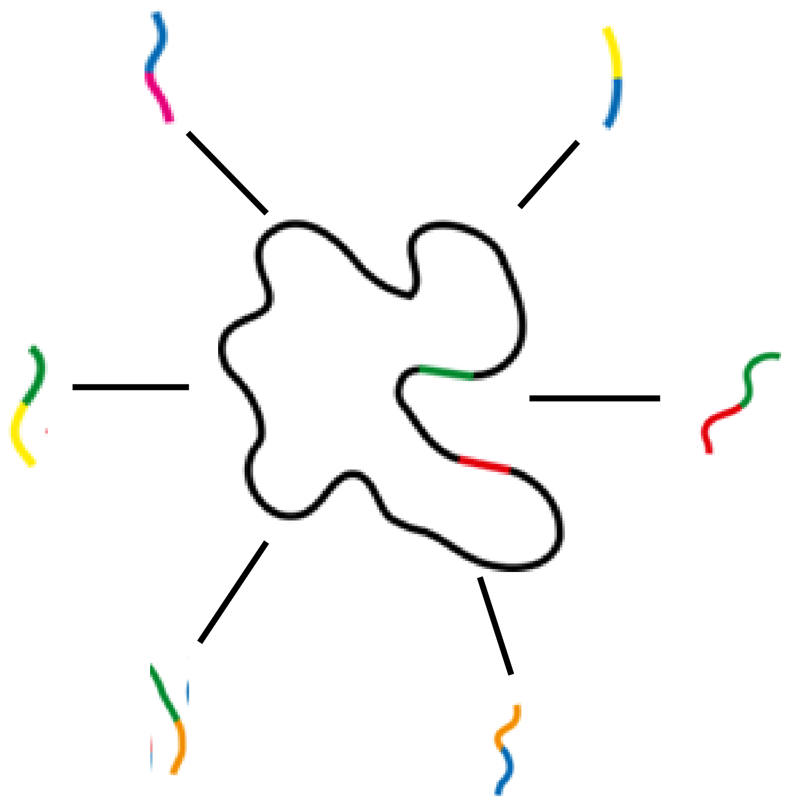


Atomic force microscope images

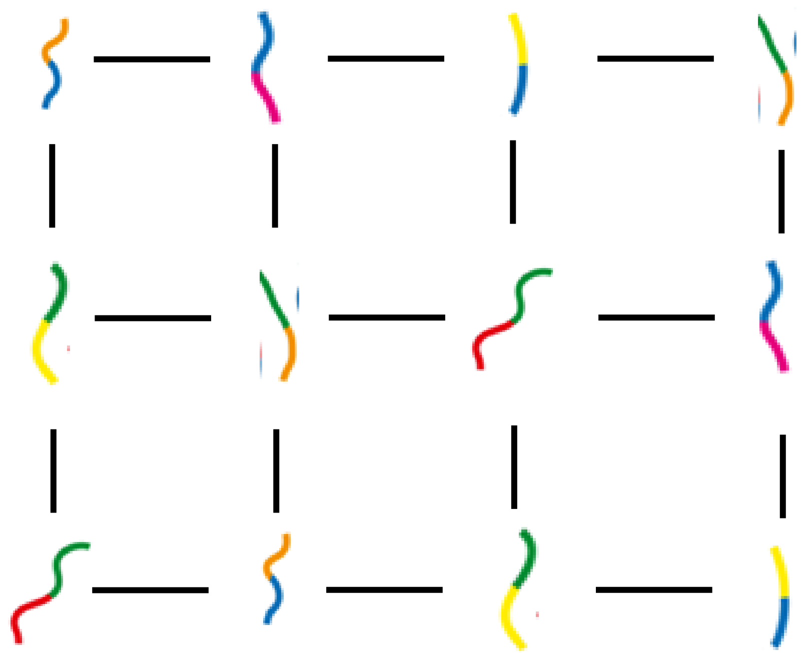
100 nm

Binding graphs

DNA origami: **star graph**
(all binding is between staples and scaffold)



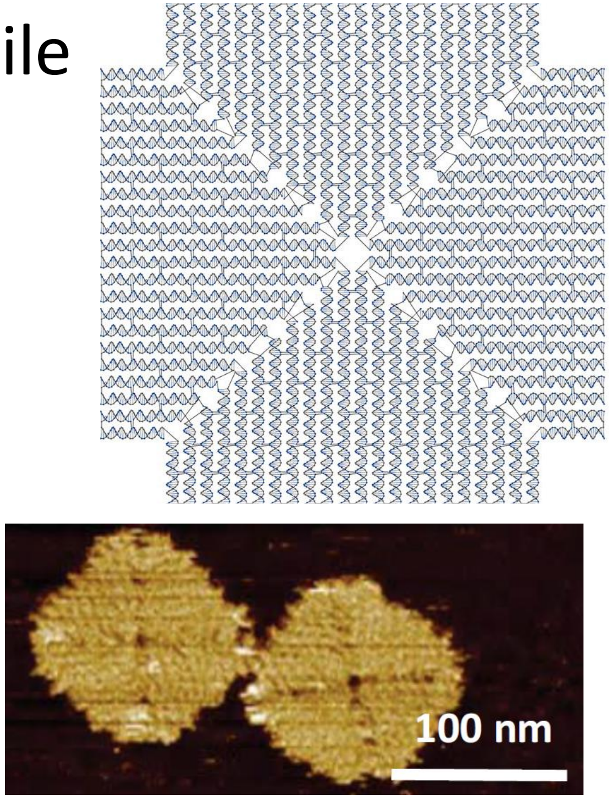
DNA tiles: **grid graph**
(tiles bind to each other, each has ≤ 4 neighbors)



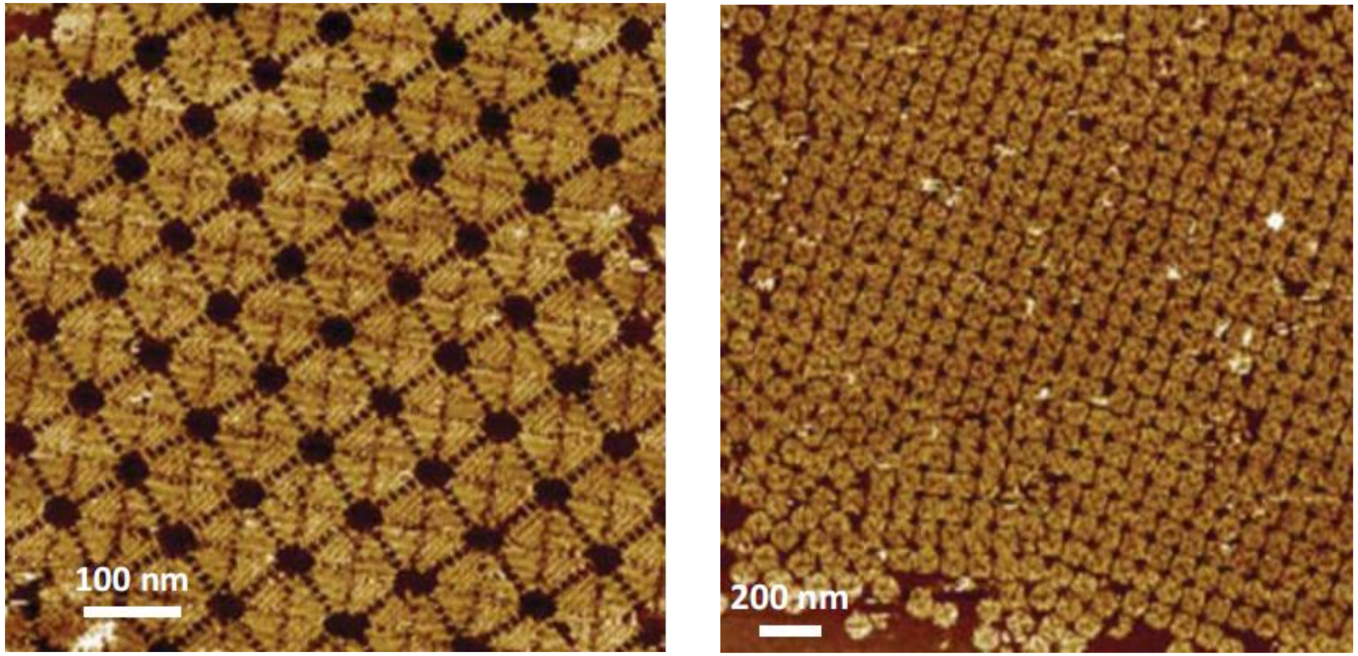
DNA tile self-assembly

monomers (“tiles” made from DNA) bind into a crystal lattice

tile



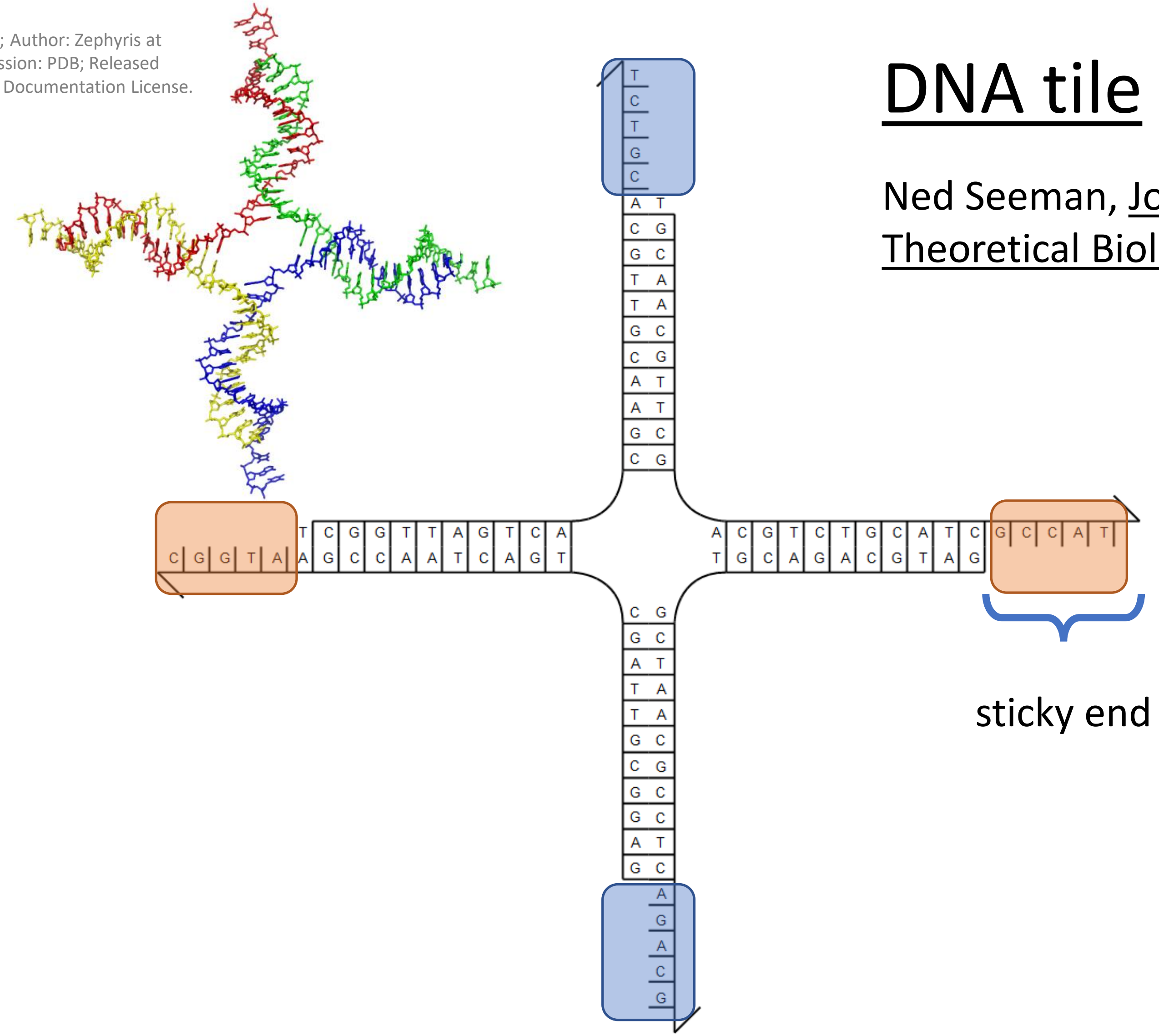
lattice



Source: *Programmable disorder in random DNA tilings*. Tikhomirov, Petersen, Qian, Nature Nanotechnology 2017

Practice of DNA tile self-assembly

Source:en.wikipedia; Author: Zephyris at en.wikipedia; Permission: PDB; Released under the GNU Free Documentation License.



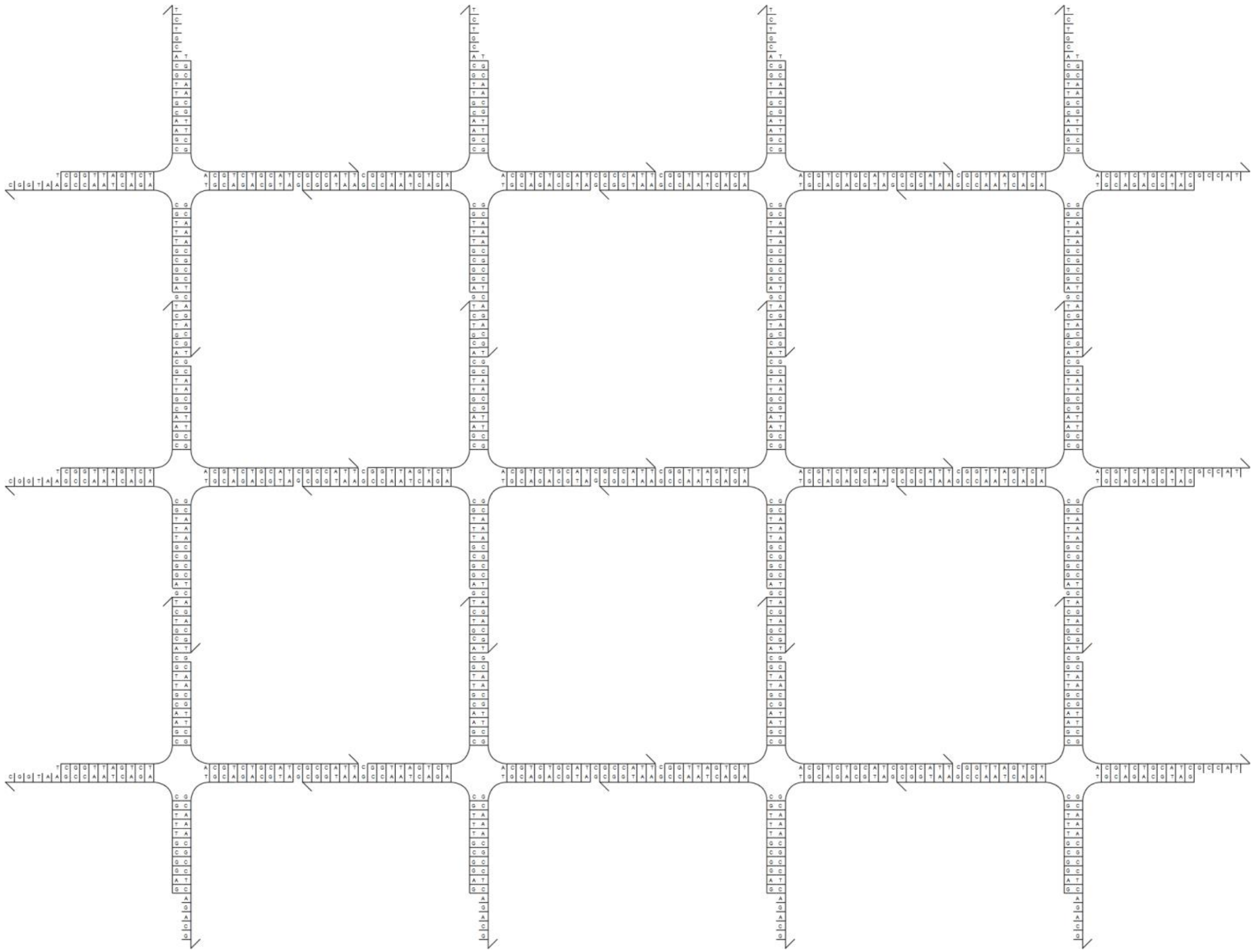
sticky end

DNA tile

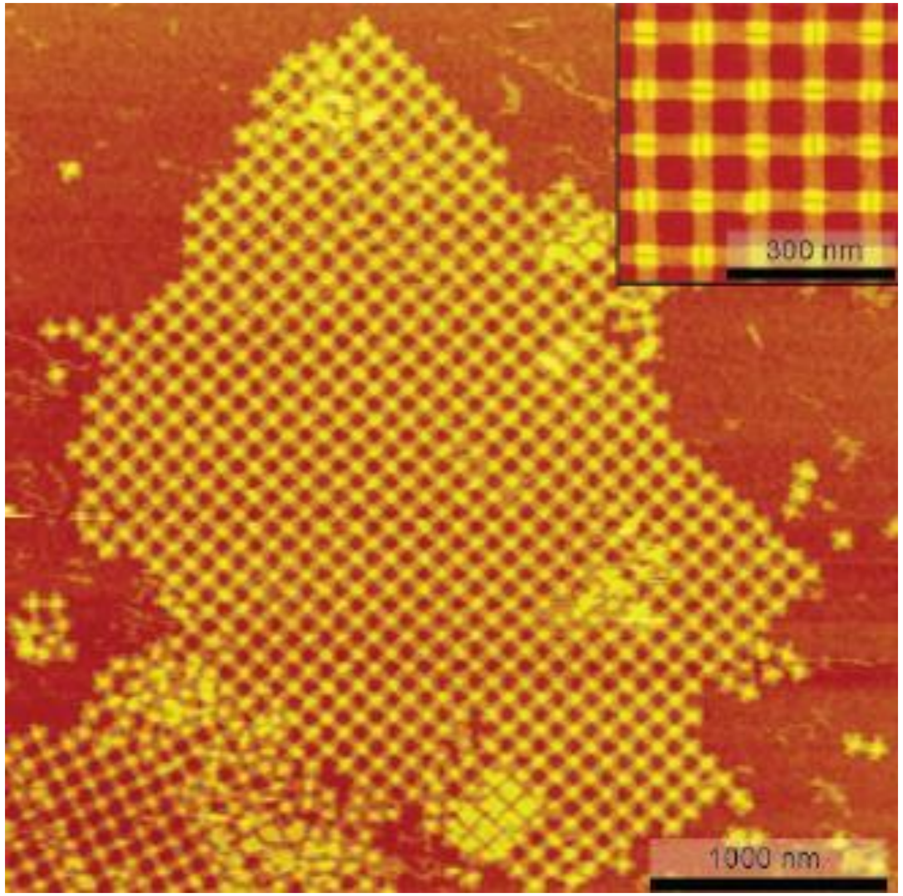
Ned Seeman, Journal of Theoretical Biology 1982

Practice of DNA tile self-assembly

Place many copies of DNA tile in solution…



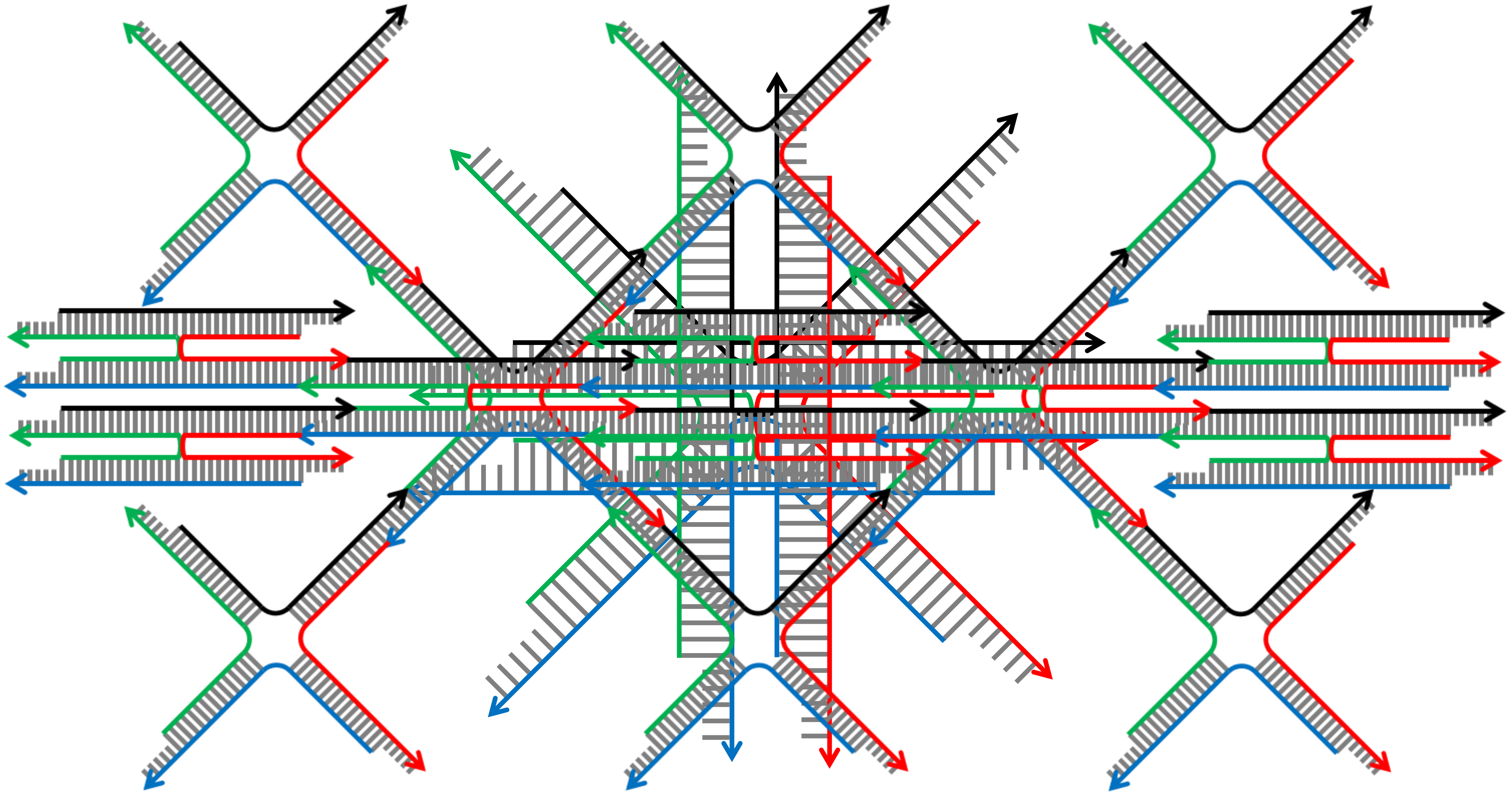
(not the same tile motif in this image)



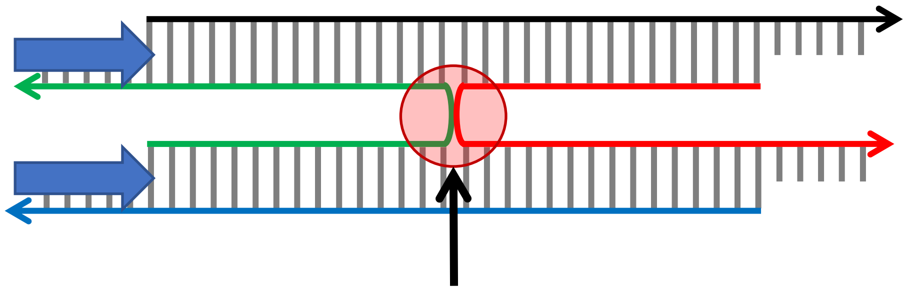
Liu, Zhong, Wang, Seeman, Angewandte Chemie 2011

Practice of DNA tile self-assembly

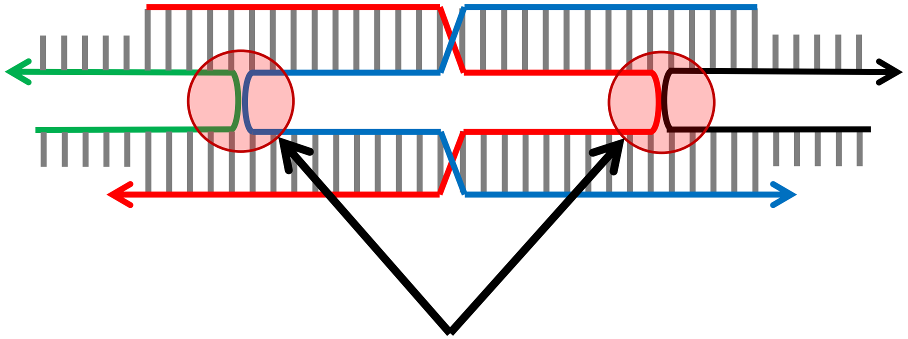
What really happens in practice to Holliday junction ("base stacking")



Practice of DNA tile self-assembly



single *crossover*



double crossover

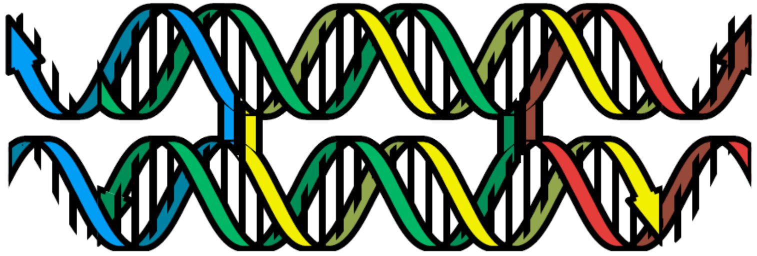
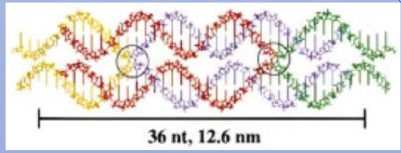
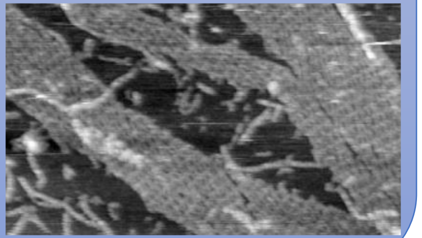


Figure from Schulman, Winfree, *PNAS* 2009

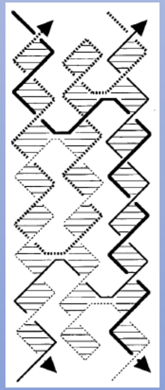
Practice of DNA tile self-assembly

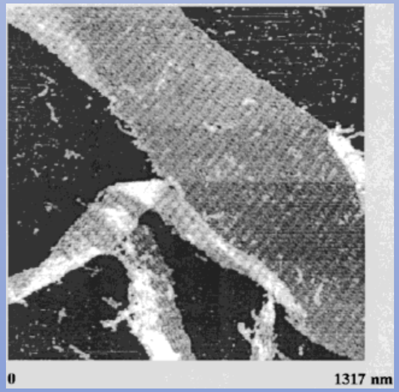
double-crossover tile (Winfree, Liu, Wenzler, Seeman, *Nature* 1998)



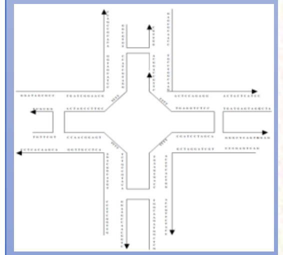


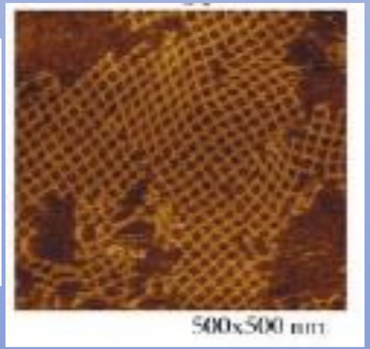
triple-crossover tile (LaBean, Yan, Kopatsch, Liu, Winfree, Reif, Seeman, *JACS* 2000)



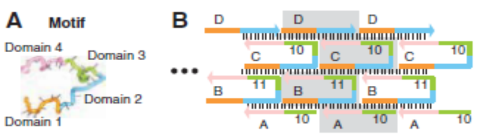


4x4 tile (Yan, Park, Finkelstein, Reif, LaBean, *Science* 2003)



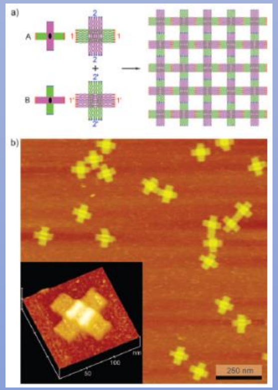


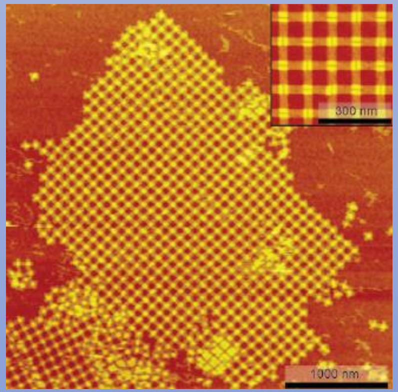
single-stranded tile (Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif, *Science* 2008)



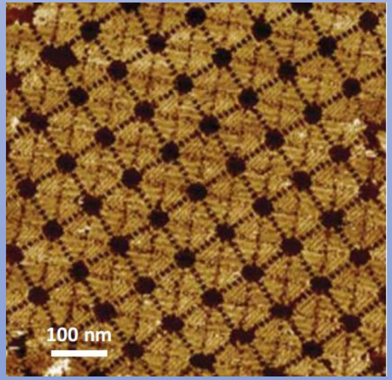
150 nm

DNA origami tile (Liu, Zhong, Wang, Seeman, *Angewandte Chemie* 2011)





Tikhomirov, Petersen, Qian, *Nature Nanotechnology* 2017



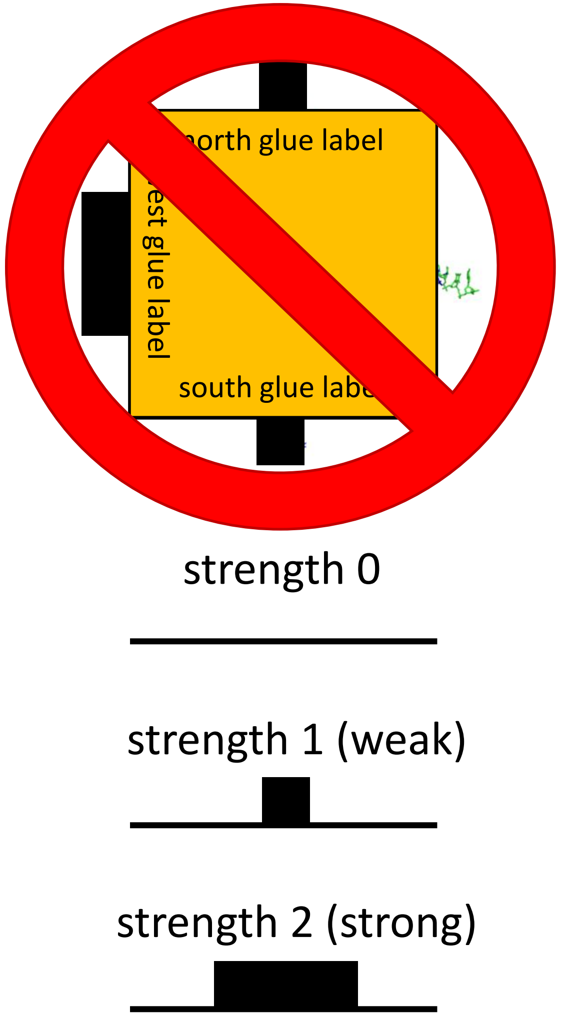
Theory of *algorithmic* self-assembly

What if…
… there is more than one tile type?
… some sticky ends are “weak”?

Erik Winfree

abstract Tile Assembly Model (aTAM)

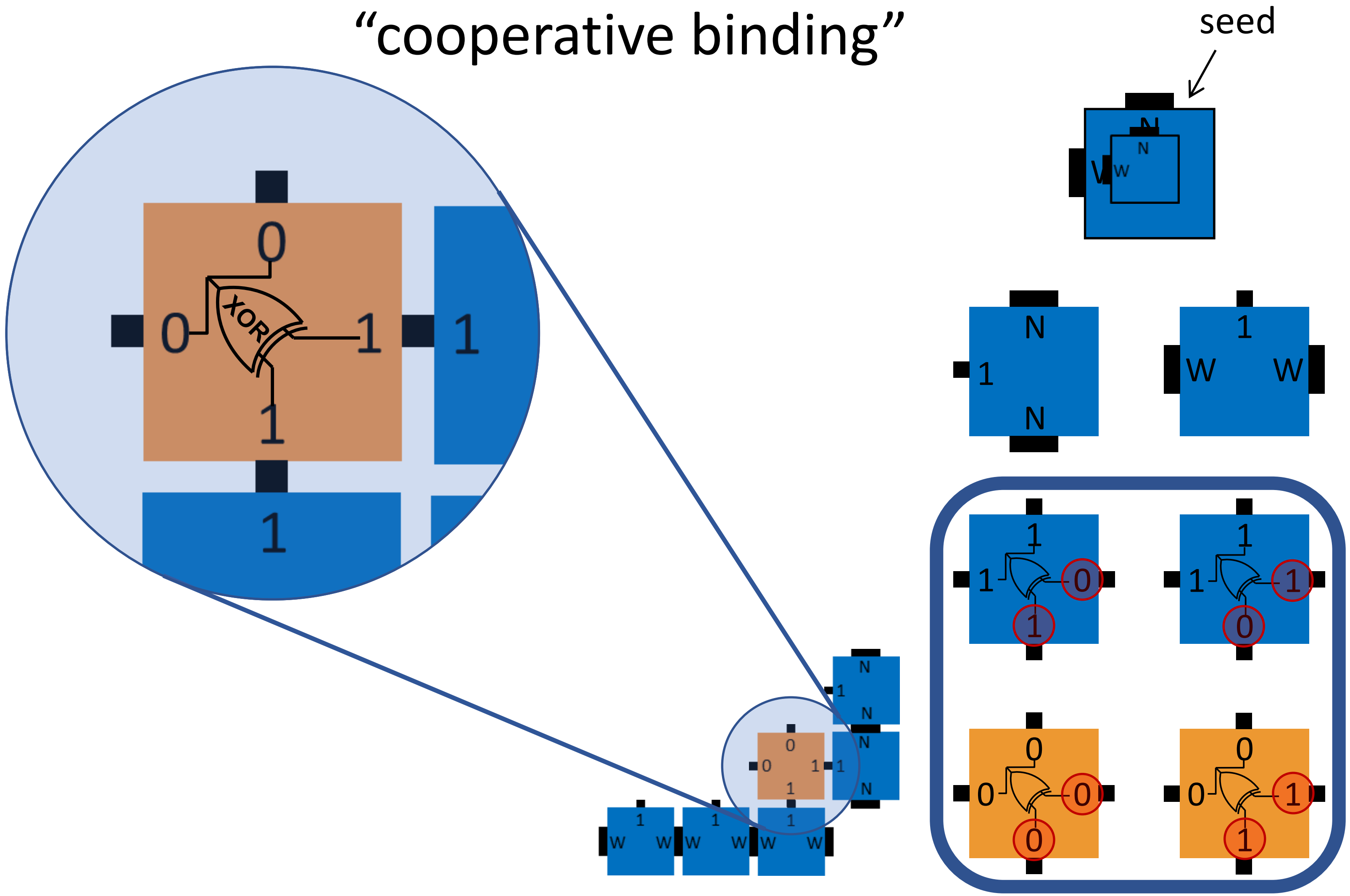
- **tile type** = unit square
- each side has a **glue** with a **label** and **strength** (0, 1, or 2)
- tiles cannot rotate

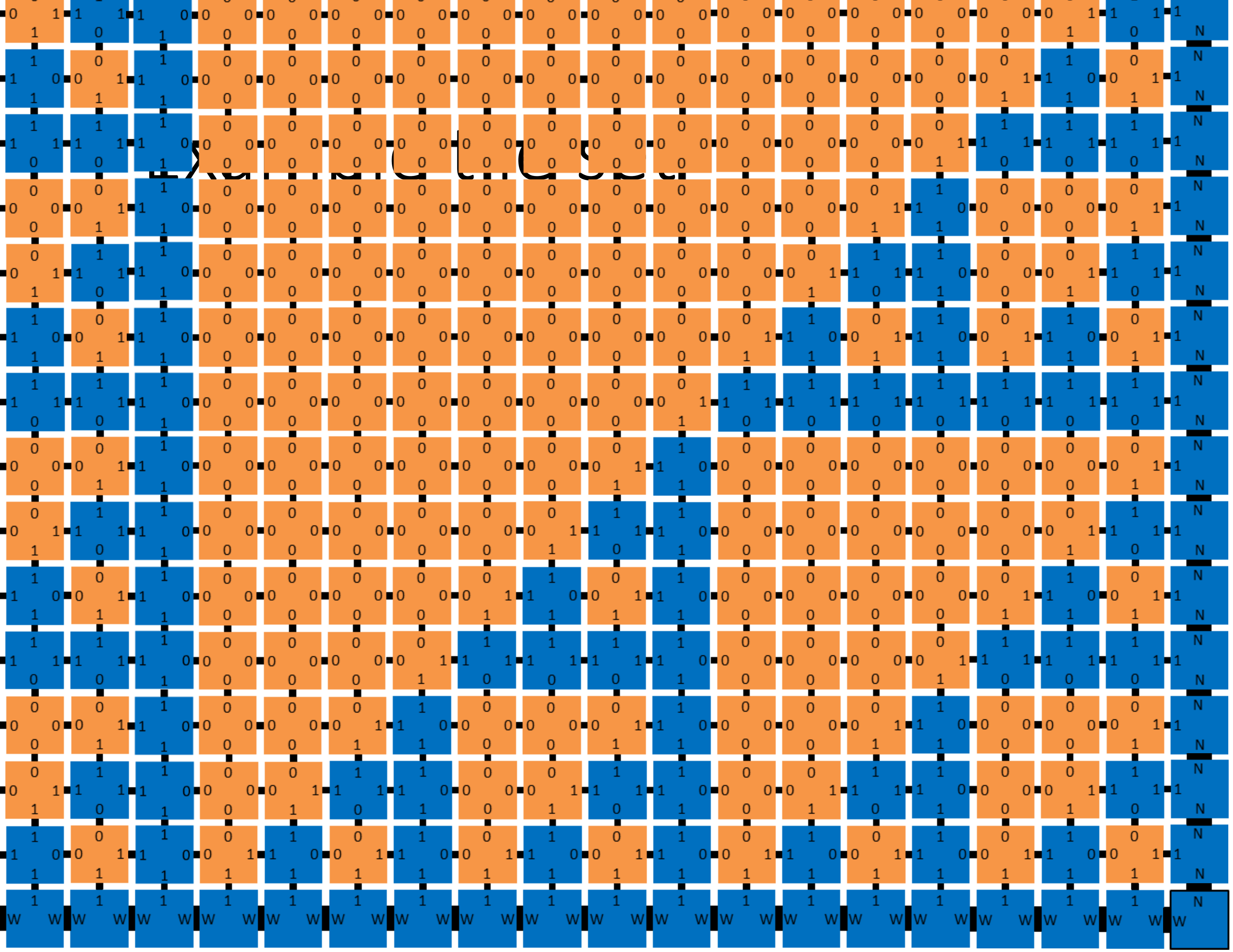


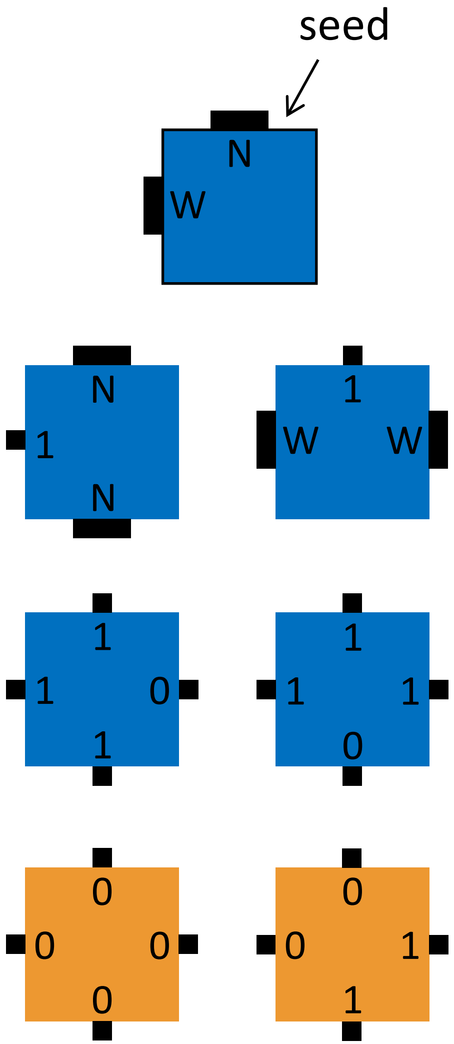
- finitely many tile **types**
- infinitely many **tiles**: copies of each type
- assembly starts as a single copy of a special **seed** tile
- tile can bind to the assembly if total binding strength ≥ 2 (two weak glues or one strong glue)

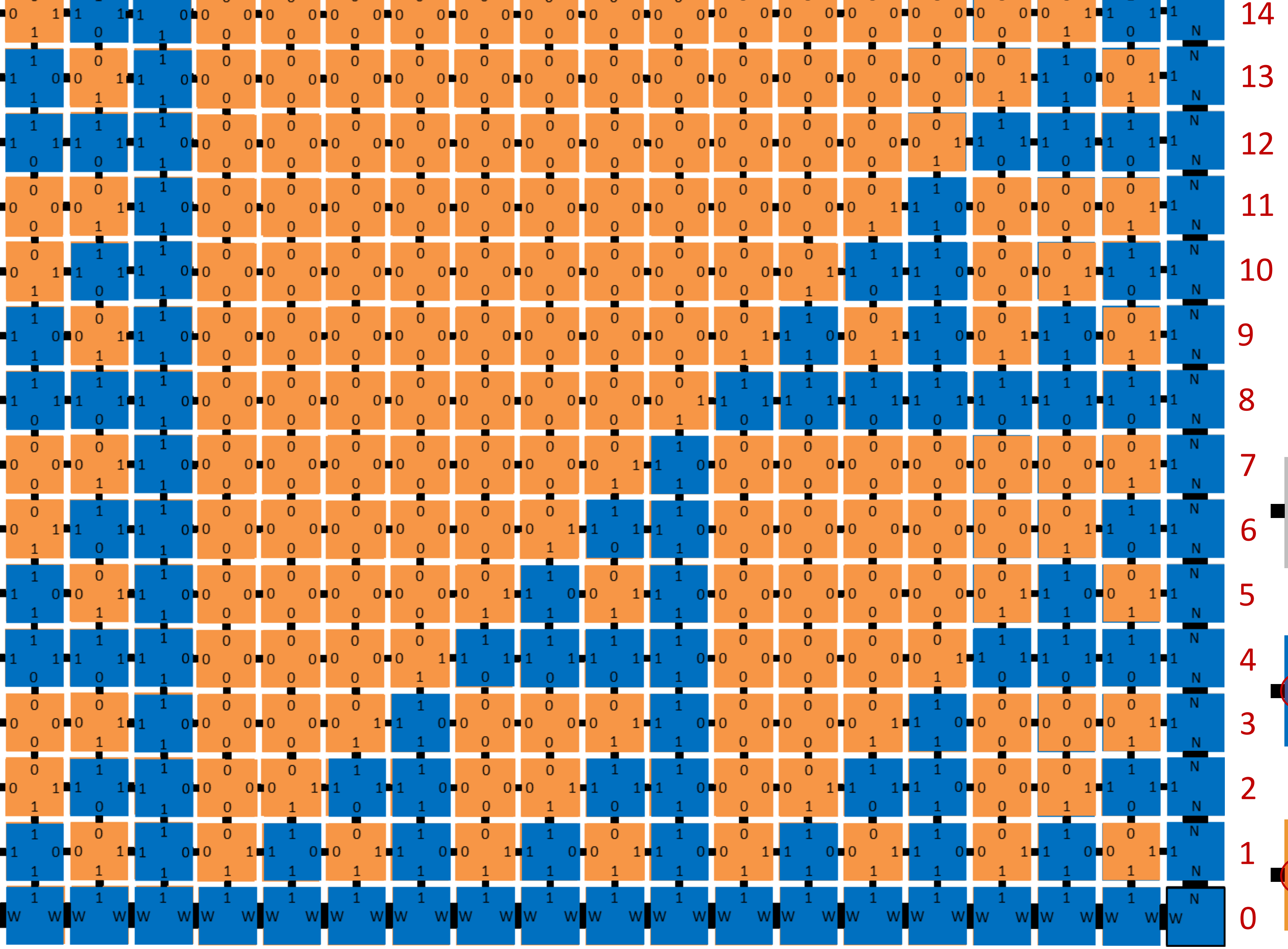
Erik Winfree, Ph.D. thesis, Caltech 1998

Example tile set

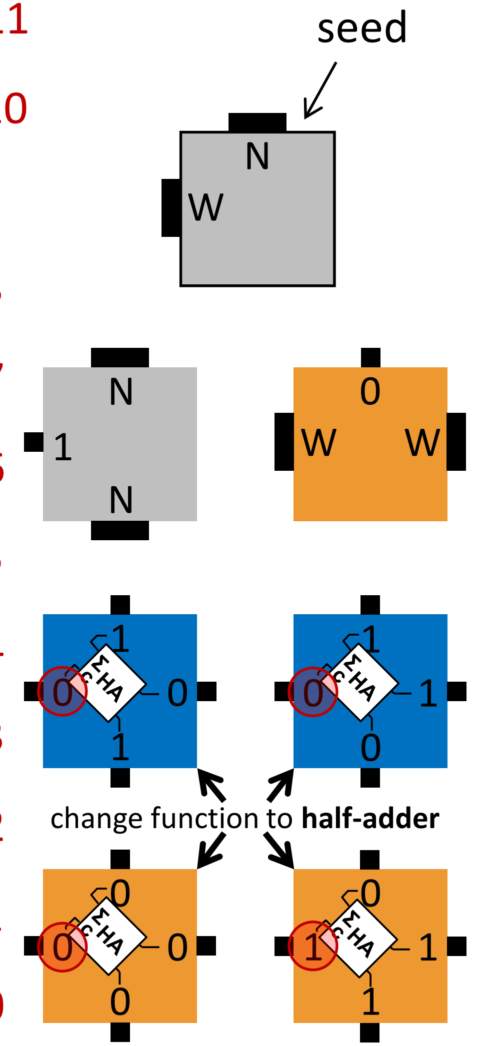





seed

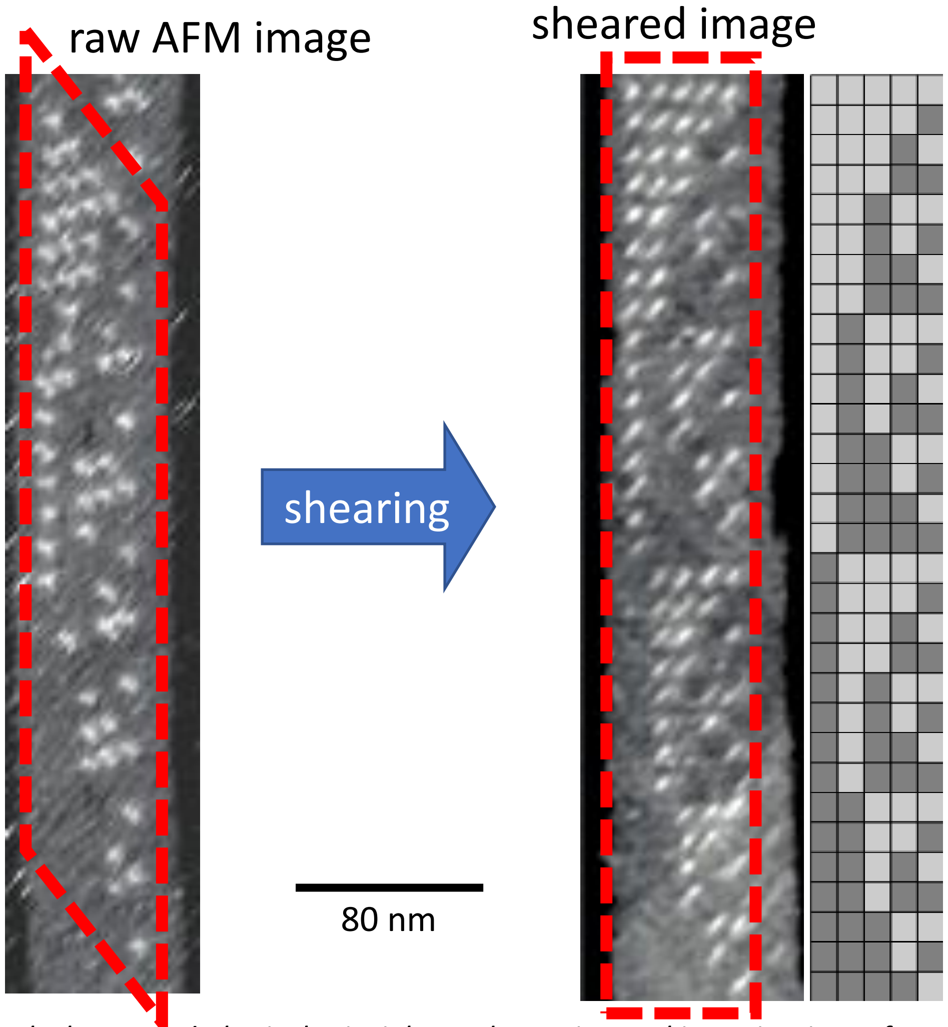


14
13
12
11
10
9
8
7
6
5
4
3
2
1
0



seed
N
W
N
1
N
0
W
W
change function to half-adder

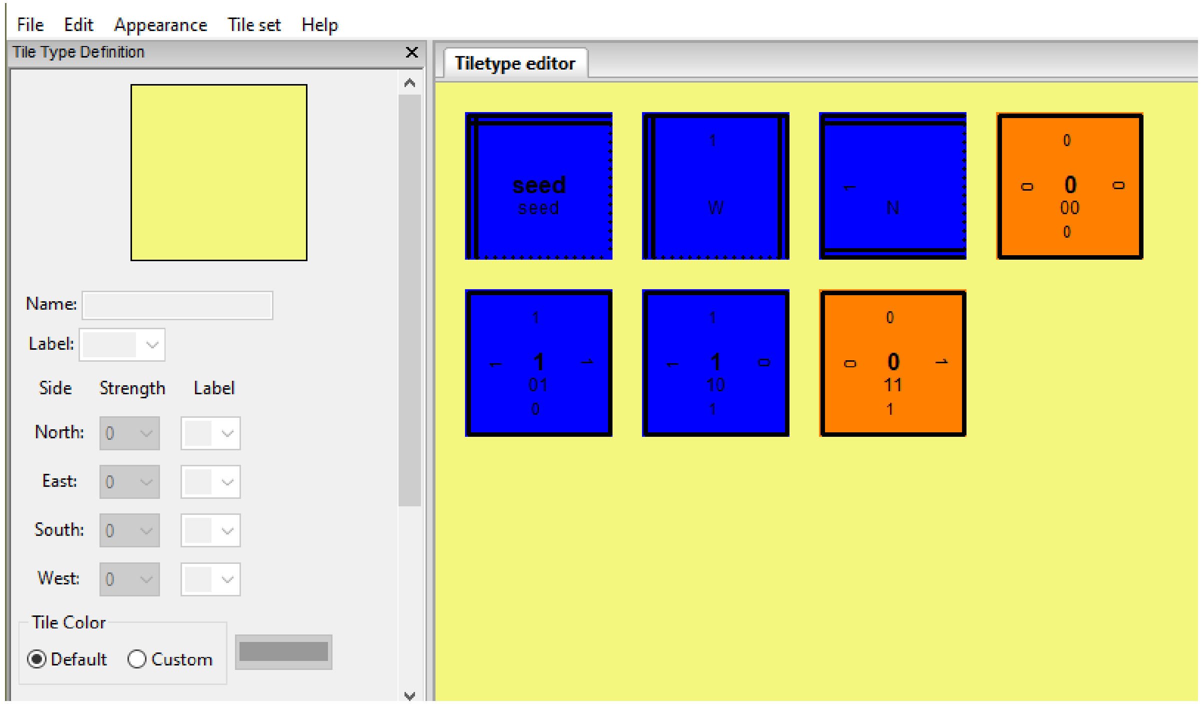
Algorithmic self-assembly in action

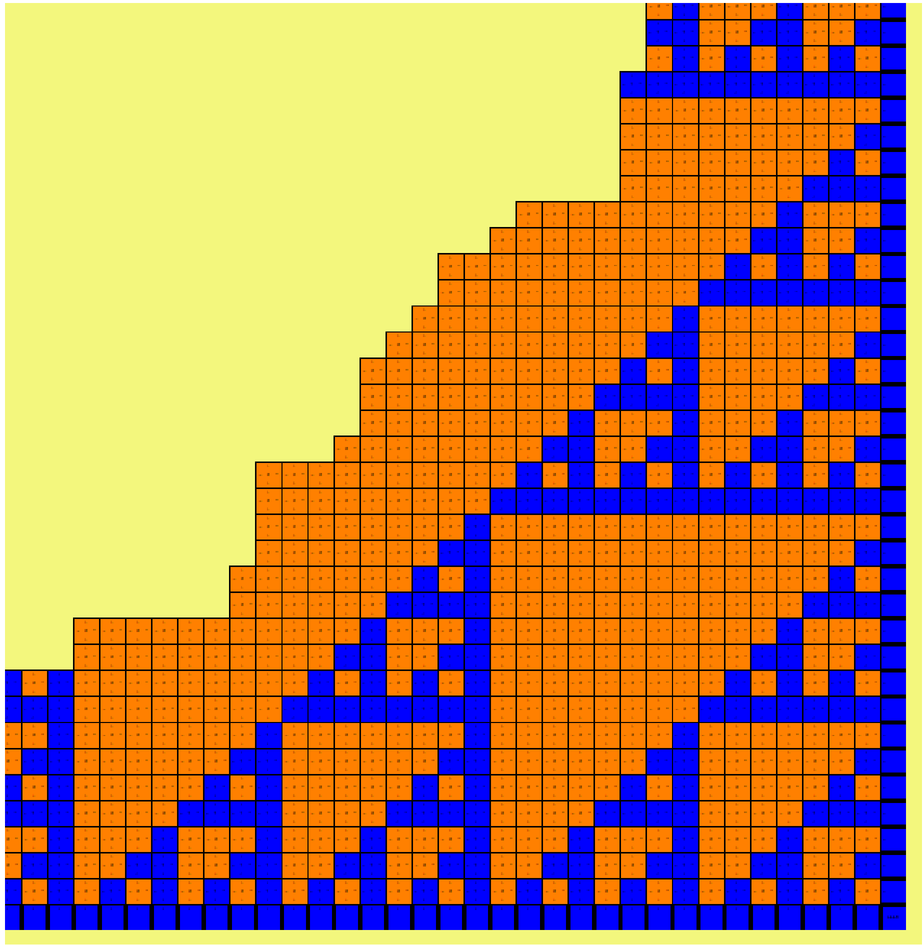


[*Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*, Constantine Evans, Ph.D. thesis, Caltech, 2014]

aTAM simulator (ISU TAS by Matt Patitz)

http://self-assembly.net/wiki/index.php?title=ISU_TAS
http://self-assembly.net/wiki/index.php?title=ISU_TAS_Tutorials
See also WebTAS by the same group:
http://self-assembly.net/software/WebTAS/WebTAS-latest/





VersaTile (by Eric Martinez and Cameron Chalk) https://github.com/ericmichael/polyomino
and xgrow (by Erik Winfree) https://www.dna.caltech.edu/Xgrow/

Tile complexity of squares

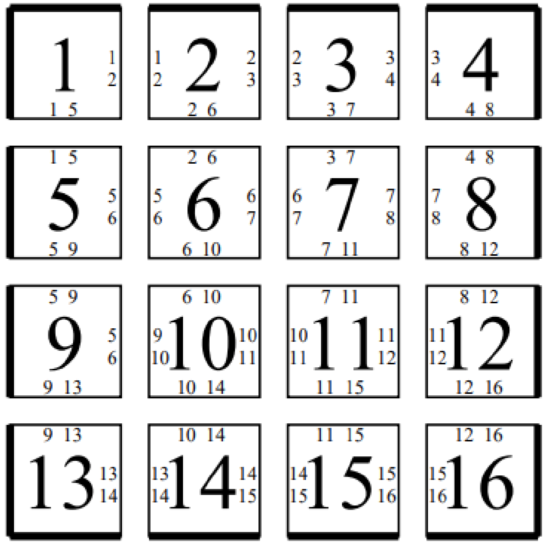
Tile complexity

- Resource bound to minimize, like time or memory with a traditional algorithm.
- Why minimize number of tile types?
  - Physically synthesizing new tile types is difficult.
  - Designing DNA sequences for new tile types is difficult. (DNA sequence design is tougher when more DNA sequences are present.)
  - But due to how modern synthesis technologies work, once a tile type is designed, making 50 quadrillion copies of the tile is as easy as making one copy.
- So, we ask: how many unique tile types to we need to self-assemble some shapes?
- We start with n x n squares as the "simplest" benchmark shape.
  - Why not a 1 x n line as an even simpler shape? What is its tile complexity?
- [*Note: we have not formally defined the aTAM yet… first let's build intuition.*]

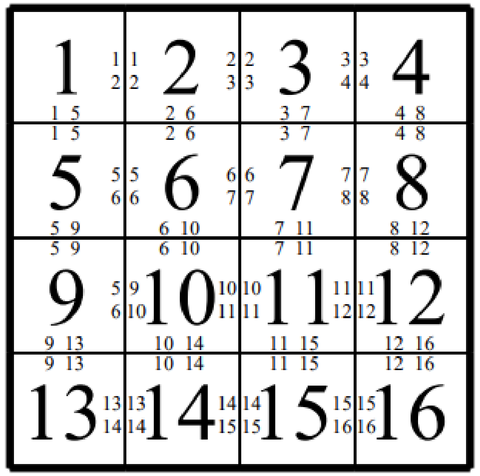
The program size complexity of self-assembled squares

Question: How many tile types do we need to self-assemble an *n* x *n* square?

Concretely: how to assemble a 4 x 4 square?



All glues are strength 2
(alternately: all are strength 1 and *temperature* τ = 1)



How many tile types does this construction need in general to assemble an *n* x *n* square?

n^2

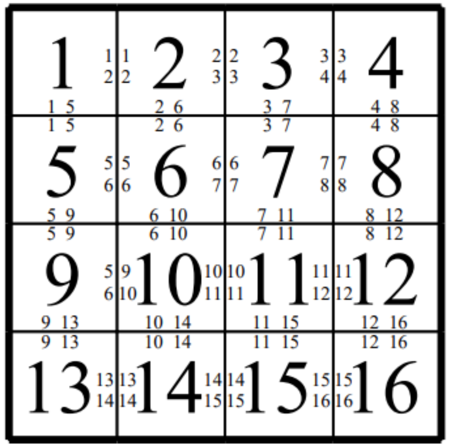
https://www.dna.caltech.edu/Papers/squares_STOC.pdf
This paper is directly responsible for convincing many theoretical computer scientists that DNA self-assembly is worth studying.

Tile complexity at temperature $\tau = 1$ (i.e., no cooperative binding allowed)

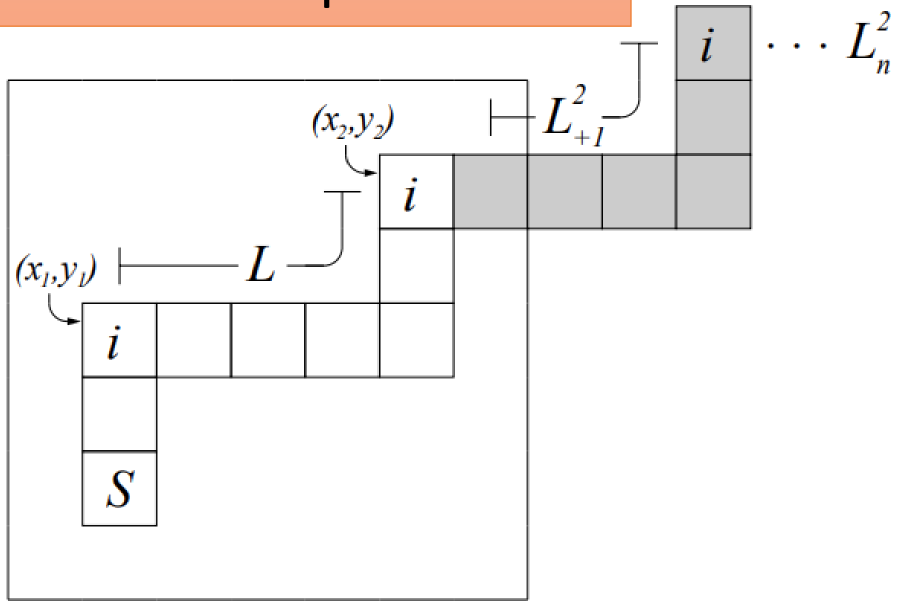
Is n^2 optimal?
Can we do better?

Theorem: At temperature $\tau = 1$, if all pairs of adjacent tiles bind with positive strength, then for every positive integer n, n^2 tile types are <u>necessary</u> to self-assemble an n x n square.

Note all pairs of adjacent tiles bind with positive strength:



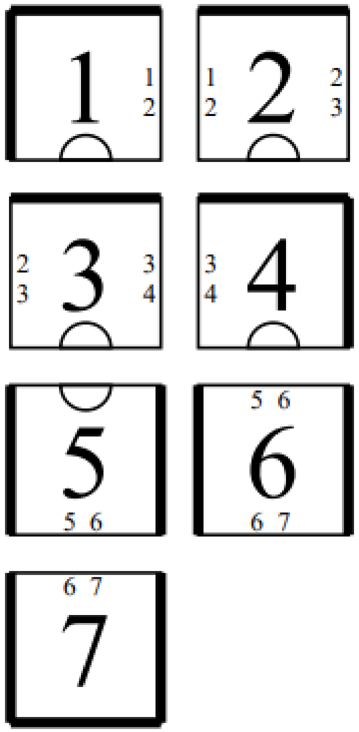
Proof: Suppose for contradiction we use the same tile type i at positions (x_1,y_1) and (x_2,y_2). Then they have a path L between them with all positive-strength glues, and this can happen instead:

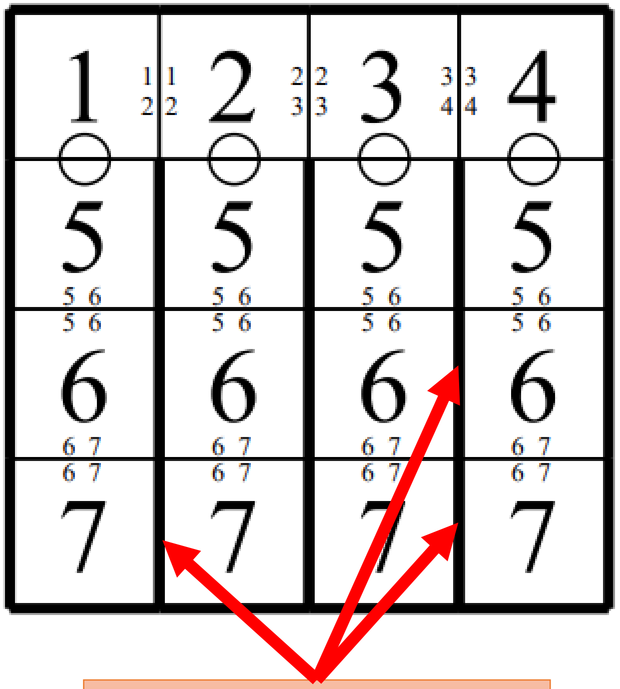


Tile complexity at temperature $\tau = 1$, where not all adjacent tiles are bound

Is n^2 still optimal?

No!





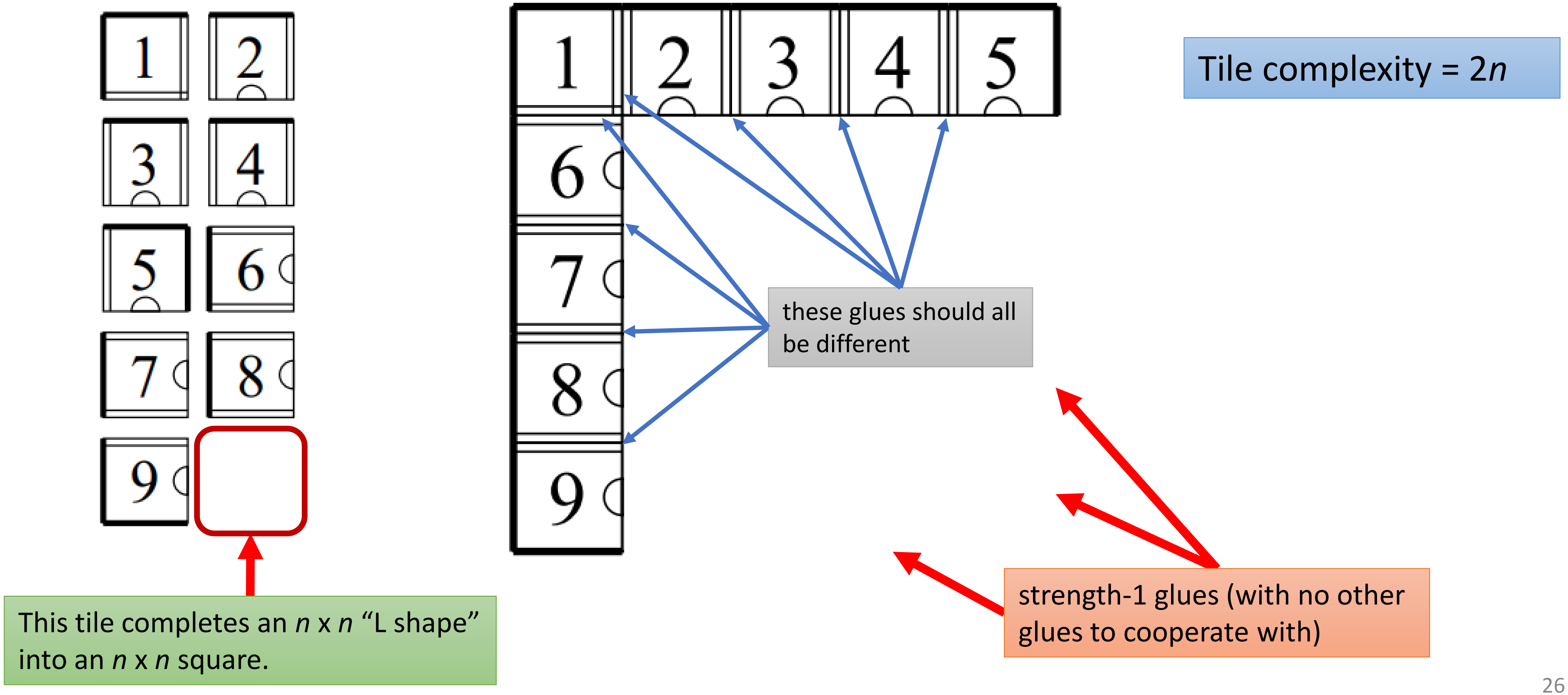
strength-0 glues

Tile complexity of this construction?

$2n - 1 = O(n)$

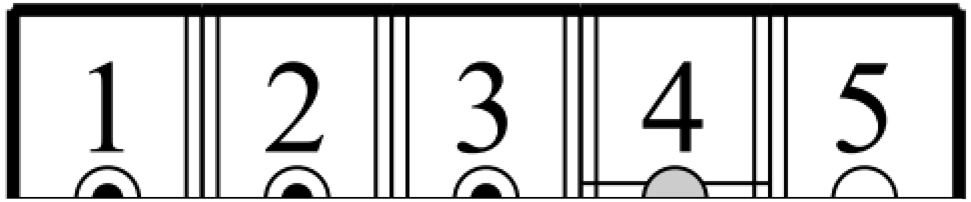
Conjecture: The temperature $\tau = 1$ tile complexity of an n x n square is $\Omega(n)$.
(most recent progress:
https://arxiv.org/abs/1902.02253
https://arxiv.org/abs/2002.04012)

Tile complexity at temperature $\tau = 2$ (i.e., cooperative binding allowed)

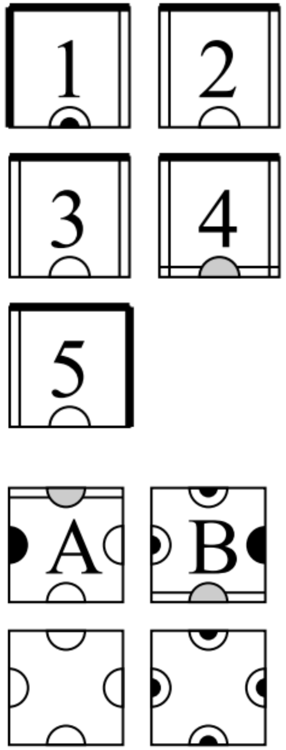


Tile complexity at temperature $\tau = 2$

Goal: complete a 1 x *n* line into an *n* x *n* square



Tile complexity = *n* + 4



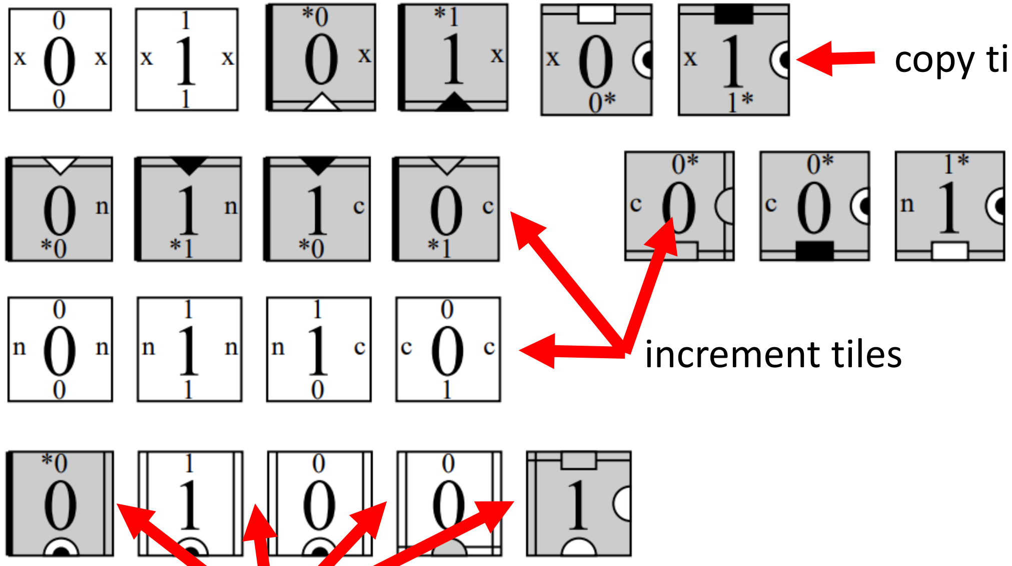
How to get *sublinear* tile complexity?

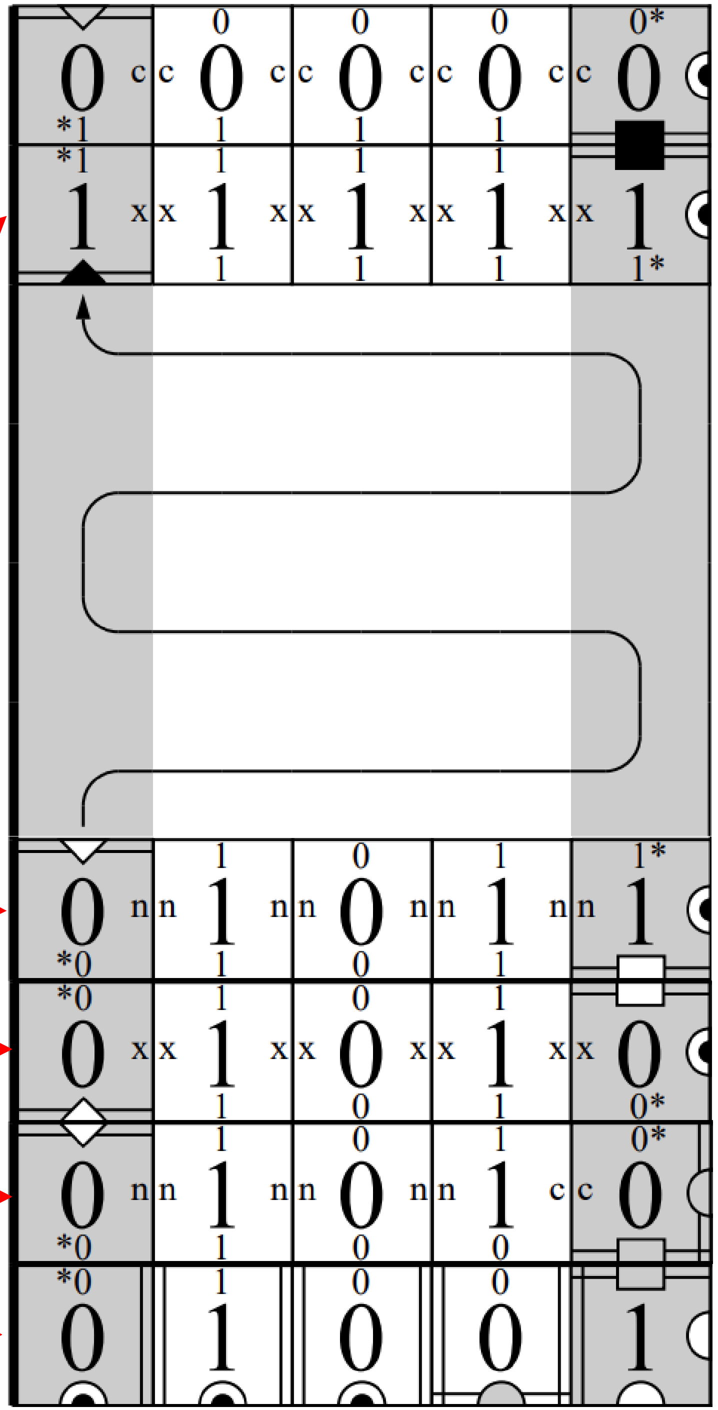
Logarithmic tile complexity at temperature $\tau = 2$

Goal: rectangle of height n using $O(\log n)$ tile types

for width of k bits, stops when it reaches what value?

“zig-zag counter”

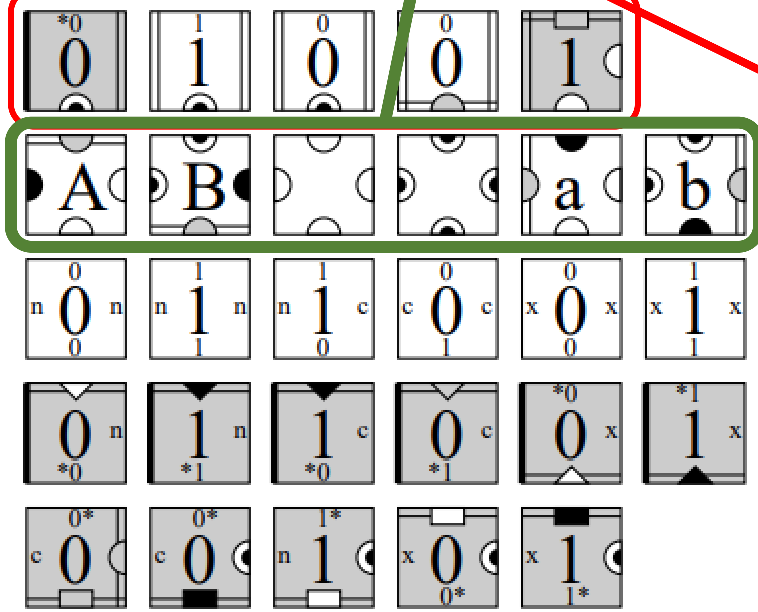


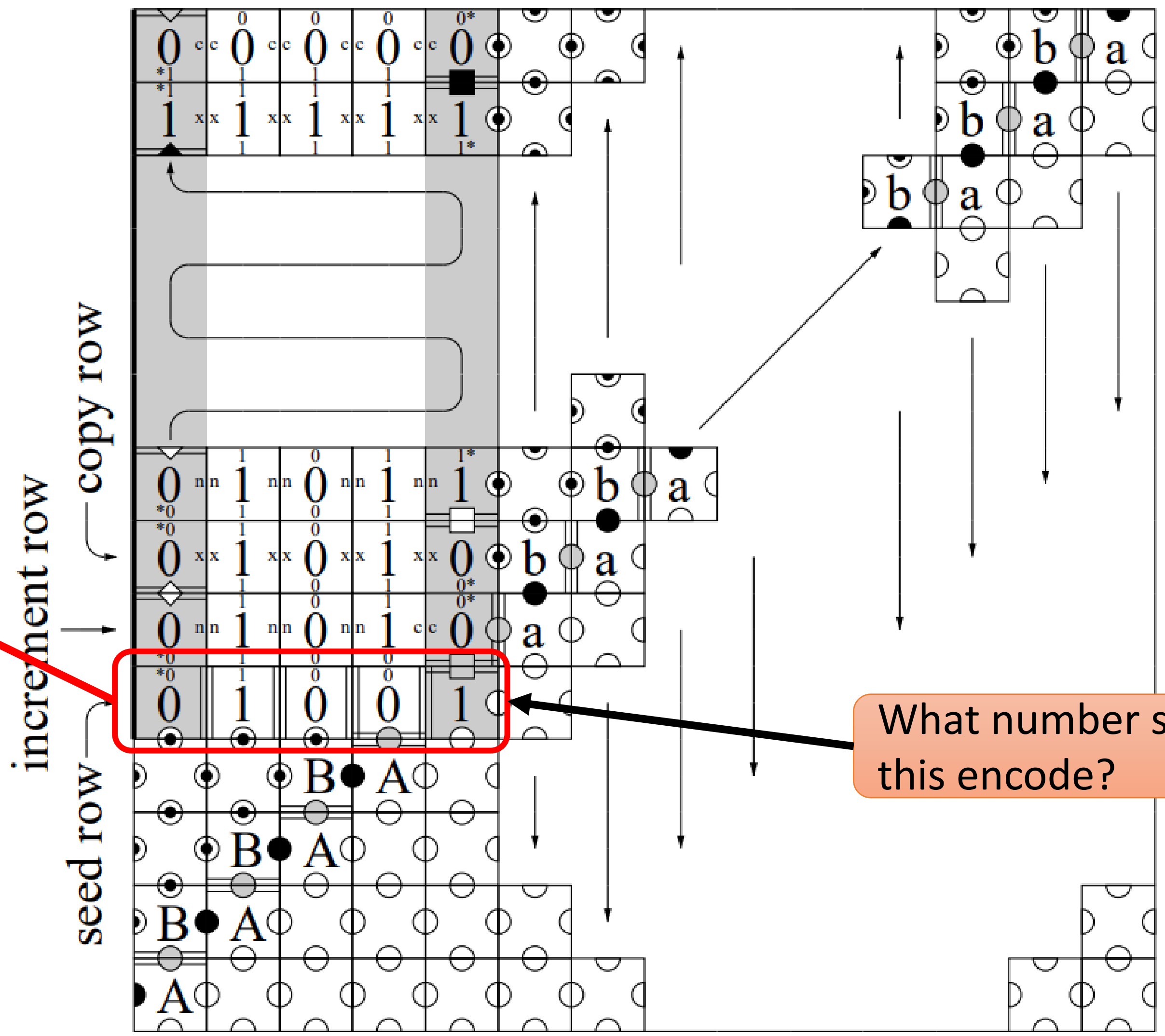


Logarithmic tile complexity at temperature $\tau = 2$

A few more "filler" tiles complete the ≈n x log n rectangle into an n x n square.

tile complexity = log n + 23





What number should this encode?

$\Omega(\log n / \log \log n)$ tile complexity lower bound for n x n squares

- What does $\Omega(\log n / \log \log n)$ tile complexity lower bound mean?
  - First let's think about what we already showed: what does $O(\log n)$ tile complexity upper bound mean? **For all *n***, $O(\log n)$ tile types is enough to self-assemble an n x n square.
  - A lower bound looks like: **For infinitely many *n***, $o(\log n / \log \log n)$ tile types is not enough to self-assemble an n x n square.
- How to prove? It's a counting argument:
  - Count number of (functionally distinct) tile systems with fewer than ¼ log p / log log p tile types.
    - We'll show that it's fewer than p.
  - There are p squares with width n between p+1 and 2p; each needs a different tile system.
  - By pigeonhole, some n x n square cannot be assembled with < ¼ log p / log log p tile types.
  - Since $p \leq n/2$, we have ¼ log p / log log p ≤ ¼ log n / log log n.
  - Since we can do this for every positive integer p, there are infinitely many n that require more than ¼ log n / log log n tile types (a stronger result holds: "most" values of n require that many)

How many tile systems with k tile types?

- **Goal**: show that there are fewer than p (“functionally distinct”) tile systems with $k = \frac{1}{4} \log p / \log \log p$ tile types.
- How many have exactly k tile types? Count each of the ways to define the tile system:
  a) How many different glues can we have? $4k$
  b) How many ways can we choose the 4 glues for one tile type? $a^4 = (4k)^4$
  c) How many ways to choose the glues for all k tile types? $b^k = (4k)^{4k}$
  d) How many ways to choose the seed tile? k
- How many tile systems? $c \cdot d = k(4k)^{4k}$

How many tile systems with k tile types?

- Number of tile systems with <u>exactly</u> k tile types: $\leq k(4k)^{4k}$
- Number of tile systems with <u>at most</u> k tile types: $\leq k^2(4k)^{4k}$
- Recall $k = ¼ \log p / \log \log p$; by algebra (see notes), $k^2(4k)^{4k} < p$.
- By pigeonhole principle, for some width n with $p < n \leq 2p$, the n x n square is not self-assembled by one of these $k^2(4k)^{4k}$ tile systems. Since those are all the tile systems with at most k tile types, the n x n square requires **more** than ¼ log p / log log p tile types to self-assemble. **QED**

"Descriptional Complexity" proof

Fact: "most" integers n require $\geq \log n$ bits to "describe".
(Though some require fewer: 1111111111111111111111 can be described by its length 22 in binary: 10110)

- Can be formalized with *Kolmogorov complexity*
  - https://en.wikipedia.org/wiki/Kolmogorov_complexity
- We can "describe" n with a tile system that self-assembles an n x n square.
- How many bits do we need to describe a tile system with k tile types?
  - $\log(4k)$ to describe one of the $4k$ glues, e.g., 8 glues: 000, 001, 010, 011, 100, 101, 110, 111
  - $4 \log(4k)$ to describe one tile type consisting of 4 glues, e.g., tile b = (010, 011, 111, 100)
  - $4k \log(4k)$ to describe all k tile types, plus $\log k$ to give index of the seed.
  - So $O(k \log k)$ bits total.
- For any n in the Fact, $\log n = O(k \log k)$, i.e., $k = \Omega(\log n / \log \log n)$.

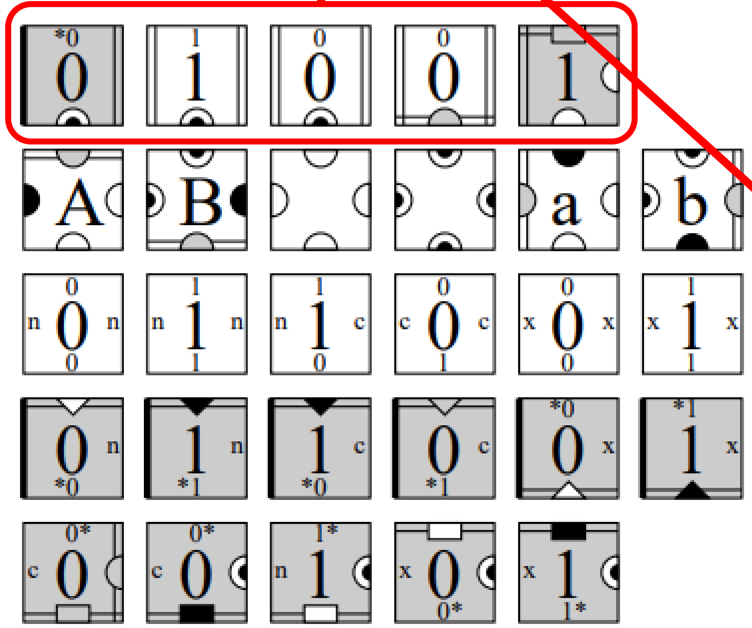
Note: we're ignoring glue strengths here; adds 2 bits per glue to describe at temperature 2.
(since there are 3 possible strengths 0, 1, 2);
see http://doi.org/10.1007/s00453-014-9879-3 for handling higher-temperature systems.

Which bound is tight?

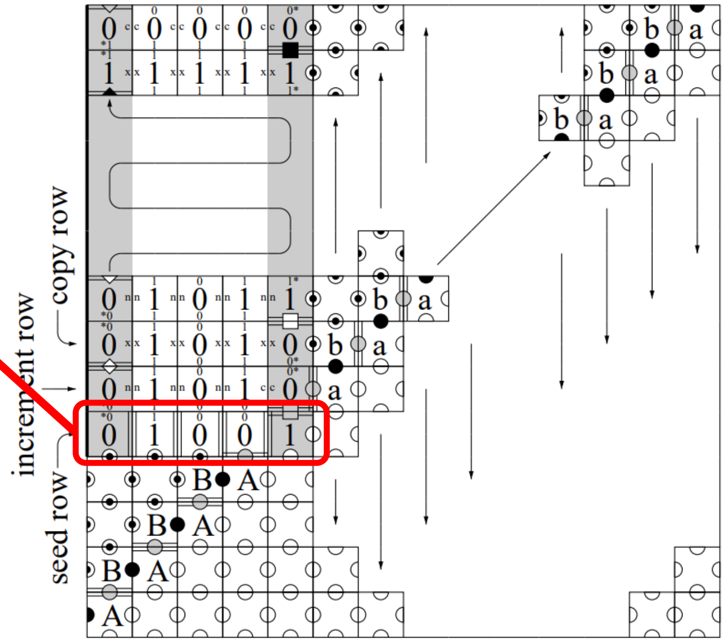
1. All n x n squares can be assembled with $O(\log n)$ tile types; can we get it down to $O(\log n / \log \log n)$?
2. Or do we need $\Omega(\log n)$ tile types to assemble infinitely many n x n squares?

Improved upper bound: self-assembling an n x n square with $O(\log n / \log \log n)$ tile types

tile complexity = $O(\log n)$ + 23



Recall:



Idea:

1) Use same 23 tiles that turn the seed row encoding a binary integer n' (related to n) into an n x n square.

2) Create the binary seed row from only $\log n / \log \log n$ tiles.

Creating a row of log n glues with arbitrary bit string $s \in \{0,1\}^{\log n}$ using O(log n / log log n) tile types

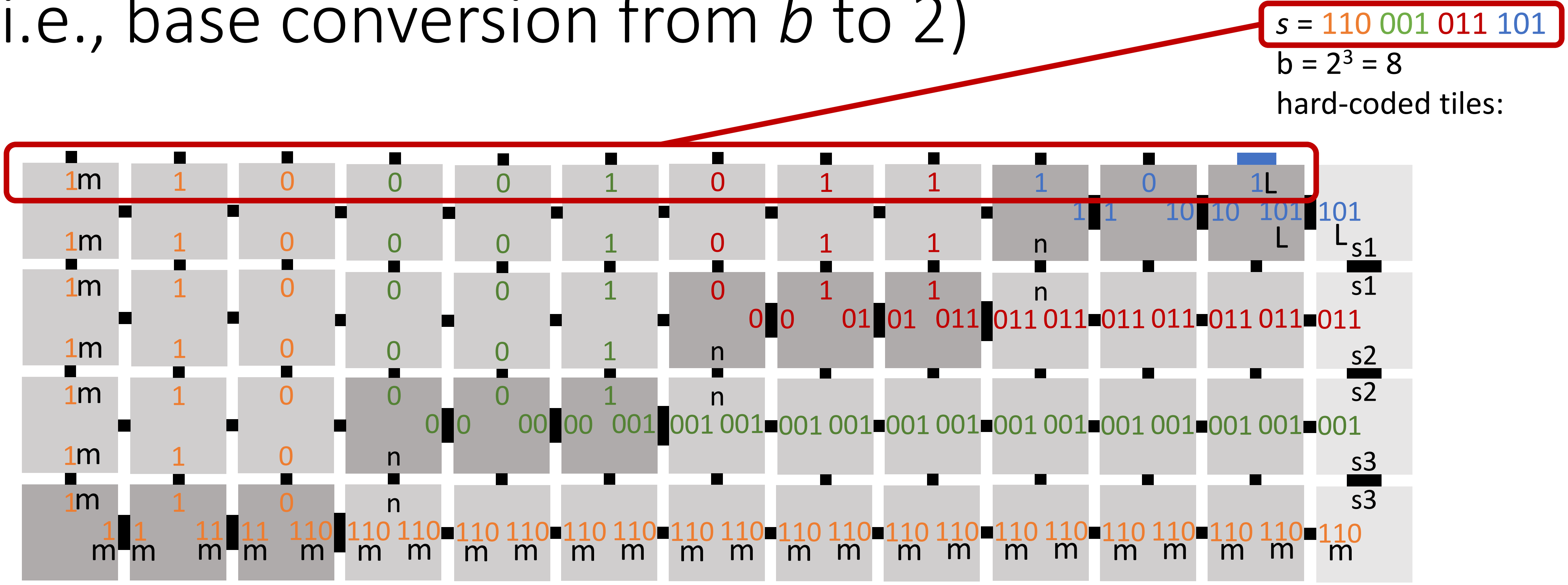
- Key idea: choose larger power-of-two base $b = 2^k$, with $b \approx \log n / \log \log n$, and convert from base b to base 2.
- How many base-b digits needed to represent a log(n)-bit integer?
- Each base-b digit is k bits
    - e.g., if $b=2^3=8$, then 0=000 1=001 2=010 3=011 4=100 5=101 6=110 7=111
    - e.g., the octal number 7125_8 in binary is 111001010101_2
    - need $\log(n) / k = \log(n) / \log(\log n / \log \log n) = \log(n) / (\log \log n - \log \log \log n) \approx \log(n) / \log \log n$ base-b digits.

Creating a row of log n glues with arbitrary bit string $s \in \{0,1\}^*$ using log n / log log n tile types (i.e., base conversion from b to 2)

s = 110 001 011 101

$b = 2^3 = 8$

hard-coded tiles:



"almost" works… what's missing?

mark glues of most and least significant bit

Formal definition of aTAM

abstract Tile Assembly Model (aTAM), formal definition

- Fix a finite alphabet Σ. A **glue** is a pair $g = (\ell,s) \in \Sigma^* \times \mathbb{N}$, with **label** ℓ and **strength** s.
- A **tile type** is a 4-tuple of glues $t \in (\Sigma^* \times \mathbb{N})^4$, with each glue listed in order north, east, south, west.
  - Define unit vectors N = (0,1), S = (0,–1), E = (1,0), W = (–1,0)
  - For $d \in$ {N, E, S, W}, let d^* denote the **opposite direction** of d, i.e., N* = S, S* = N, E* = W, W* = E.
  - Let t[N], t[E], t[S], t[W] be the glues of t in order.
  - T denotes the set of tile types.
- An **assembly** is a partial function $\alpha: \mathbb{Z}^2 \dashrightarrow T$, such that dom α (set of points where α is defined) is connected.
  - a partial function indicating, for each $(x,y) \in \mathbb{Z}^2$, which tile is at (x,y), with $\alpha(x,y)$ undefined if no tile appears there.
- Let S_α = dom α denote the **shape** of α. Let $|\alpha| = |S_\alpha|$.
- Given $p,q \in S_\alpha$, two tiles $t_p = \alpha(p)$ and $t_q = \alpha(q)$ **interact** (a.k.a. **bind**) if:
  - $\|p - q\|_2 = 1$ (*positions* $p \in \mathbb{Z}^2$ *and* $q \in \mathbb{Z}^2$ *are adjacent*)
  - letting $d = q - p$ (*the direction pointing from p to q*), $t_p[d] = t_q[d^*]$ (*the glues match where* t_p *and* t_q *touch*)
  - $t_p[d]$ has positive strength (*the glues are not zero-strength*)
- Let $B_\alpha = (V,E)$ denote the **binding graph** of α, where
  - $V = S_\alpha$
  - E = { (p,q) | $\alpha(p)$ and $\alpha(q)$ interact }
  - B_α is a *weighted*, *undirected* graph: Each edge's weight is the strength of the glue it represents.
- Given $\tau \in \mathbb{N}^+$, α is **τ-stable** if the minimum weight cut of B_α is at least τ.
  - i.e., to separate α into two pieces requires breaking bonds of strength at least τ.

abstract Tile Assembly Model (aTAM), formal definition

- Given assemblies $\alpha,\beta: \mathbb{Z}^2 \dashrightarrow T$, we say α is a **subassembly** of β, written $\alpha \sqsubseteq \beta$ if
  - $S_\alpha \subseteq S_\beta$ (*α is contained in* β), and
  - for all $p \in S_\alpha$, $\alpha(p) = \beta(p)$ (α *and* β *agree on tile types wherever they share a position*)

Question: If $\alpha \sqsubseteq \beta$, can α grow into β?

- We say $\Theta = (T,\sigma,\tau)$ is a **tile system**, where T is a finite set of tile types, $\tau \in \mathbb{N}^+$ is the **temperature**, and $\sigma: \mathbb{Z}^2 \dashrightarrow T$ is the finite, τ-stable **seed assembly**.
- We say **α produces β in one step**, denoted $\alpha \to_1 \beta$, to denote that $\alpha \sqsubseteq \beta$, $|S_\beta \setminus S_\alpha| = 1$, and letting $\{p\} = S_\beta \setminus S_\alpha$ be the point in β but not α, the cut $(\{p\},S_\alpha)$ of the binding graph B_β has weight $\geq \tau$.
  - (*one new tile* $\beta(p)$ *attaches to* α *with strength at least* τ *to create* β)
  - If the tile type added is t, write $\beta = \alpha + (p \mapsto t)$.
- The **frontier** of α is denoted $\partial\alpha = \bigcup_{\alpha \to_1 \beta} (S_\beta \setminus S_\alpha)$ (*empty locations adjacent to α where a tile can stably attach to α.*)
- A sequence of $k \in \mathbb{N}\cup\{\infty\}$ assemblies $\alpha_0, \alpha_1, \ldots$ is an **assembly sequence** if for all $0 \leq i < k$, $\alpha_i \to_1 \alpha_{i+1}$.
- We say that **α produces β** (in 0 or more steps), denoted $\alpha \to \beta$, if there is an assembly sequence $\alpha_0, \alpha_1, \ldots$ of length $k \in \mathbb{N}\cup\{\infty\}$ such that
  - $\alpha = \alpha_0$
  - for all $0 \leq i < k$, $\alpha_i \sqsubseteq \beta$, and
  - $S_\beta = \bigcup_i S_{\alpha i}$

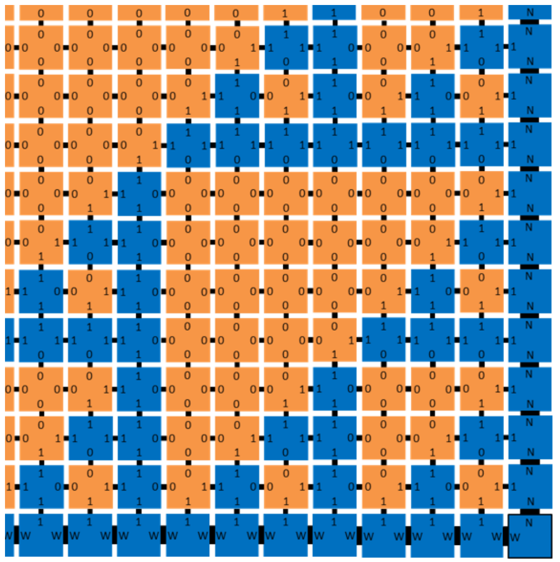
Why can't we just say $\to$ is the reflexive, transitive closure $\to_1^*$ of $\to_1$?

Sometimes we write $\alpha \to^\Theta \beta$ to emphasize this is with respect to a particular tile system Θ.

- We say β is the **result** of the assembly sequence.
- If k is finite, it is routine to verify that $\beta = \alpha_k$, and $\to$ <u>is</u> the reflexive, transitive closure $\to_1^*$ of $\to_1$.

abstract Tile Assembly Model (aTAM), formal definition

- Given tile system $\Theta = (T,\sigma,\tau)$, we say α is **producible** if $\sigma \rightarrow \alpha$.
  - Write $A[\Theta]$ to denote the set of all producible assemblies.
- We say α is **terminal** if α is stable and $\partial\alpha = \emptyset$. (*no tile can stably attach to it*)
  - Write $A_{\square}[\Theta] \subseteq A[\Theta]$ to denote the set of all producible, terminal assemblies.
- We say Θ is **directed** (a.k.a., **deterministic**) if
  - $|A_{\square}[\Theta]| = 1$. (*this is what we want it to mean: only one terminal producible assembly*)
  - equivalently, the partially ordered set $(A[\Theta], \rightarrow)$ is *directed*: for each $\alpha,\beta \in A[\Theta]$, there exists $\gamma \in A[\Theta]$ such that $\alpha \rightarrow \gamma$ and $\beta \rightarrow \gamma$.
  - equivalently, for all $\alpha,\beta \in A[\Theta]$ and all $p \in S_\alpha \cap S_\beta$, $\alpha(p) = \beta(p)$.
- Let X be a **shape**, a connected subset of $\mathbb{Z}^2$. Θ **strictly self-assembles** X if, for all $\alpha \in A_{\square}[\Theta]$, $S_\alpha = X$. (*every terminal producible assembly has shape X*)
  - Note X can be infinite.
  - Example: strict self-assembly of entire second quadrant $X = \{ (x,y) \in \mathbb{Z}^2 \mid x \geq 0 \text{ and } y \leq 0 \}$
  - Example of tile system Θ that does not strictly self-assemble any shape?
- Let $X \subseteq \mathbb{Z}^2$. Θ **weakly self-assembles** X if there is a subset $B \subseteq T$ (the "blue tiles") such that, for all $\alpha \in A_{\square}[\Theta]$, $X = \alpha^{-1}(B)$. (*every terminal producible assembly puts blue tiles exactly on X*.)
  - example: weak self-assembly of the discrete Sierpinski triangle.



Basic stability result

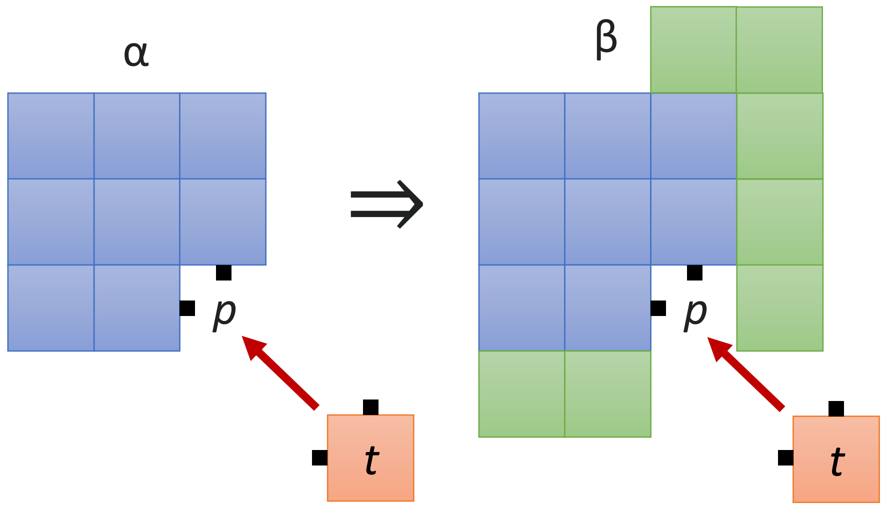
Observation: Let $\alpha \sqsubseteq \beta$ be stable assemblies and $p \in \mathbb{Z}^2 \setminus S_\beta$ such that $\alpha + (p \mapsto t)$ is stable. Then $\beta + (p \mapsto t)$ is also stable.

Intuition: if a tile can attach to α, it can attach in the presence of extra tiles on α.

Proof:

1. Since β is stable and glue strengths are nonnegative, the only *potentially* unstable cut is $(\{p\}, S_\beta)$.
2. But:
    1. $\alpha \sqsubseteq \beta$,
    2. $\alpha + (p \mapsto t)$ is stable,
    3. compared to α, β only has extra tiles on the other side of the cut (t, S_β).
    4. so the cut (t, S_β) is also stable. **QED**

example:



Basic reachability result

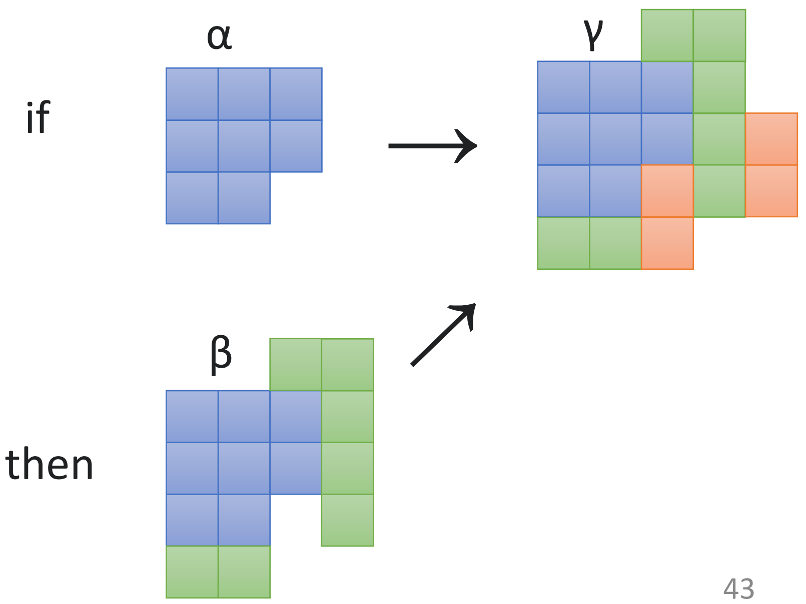
Rothemund's Lemma: Let $\alpha \sqsubseteq \beta \sqsubseteq \gamma$ be stable assemblies such that $\alpha \to \gamma$. Then $\beta \to \gamma$.

Intuition: if α can grow into γ, then if some of what will attach is already present (β), the remaining tiles can still attach.

Proof:

1. Let $\alpha=\alpha_0, \alpha_1, \ldots$ be an assembly sequence with result γ.
2. For each i, let $p_i = S_{\alpha i+1} \setminus S_{\alpha i}$ (*i'th attachment position*) and t_i the *i*'th tile added.
3. Let $i(0) < i(1) < \ldots$ such that $S_\gamma \setminus S_\beta = \{i(0), i(1), \ldots\}$ (*subsequence of indices of tile attached outside of* β).
4. Define assembly sequence $\beta=\beta_0,\beta_1,\ldots$ by $\beta_{j+1} = \beta_j+(p_{i(j)} \mapsto t_{i(j)})$. (*adding tiles to* $S_\gamma \setminus S_\beta$ *in order they were added to* α*, skipping tiles already in* S_β.)
5. Then for each j, $\alpha_{i(j)} \sqsubseteq \beta_j$, so previous Observation implies that $\beta_j + (p_{i(j)} \mapsto t_{i(j)})$ is stable.
6. Thus the assembly sequence is valid (each tile attachment is stable), showing $\beta \to \gamma$. **QED**

example:



example of usefulness of Rothemund's Lemma

- Recall two alternate characterizations of deterministic tile systems:
  (a) $|A_{\square}[\Theta]| = 1$.
  (b) for all $\alpha,\beta \in A[\Theta]$ and all $p \in S_\alpha \cap S_\beta$, $\alpha(p) = \beta(p)$.
- Rothemund's Lemma can be used to show that (b) implies (a)
  - will skip in lecture (optional problem on homework 1)

Fair assembly sequences

Definition: Let $\alpha_0, \alpha_1, \ldots$ be an assembly sequence. We say it is **fair** if, for all $i \in \mathbb{N}$ and all $p \in \partial\alpha_i$, there exists $j > i$ such that $p \in S_{\alpha j}$.

Intuition: Every frontier location eventually gets a tile; none are "starved"

Lemma: Let $\alpha_0, \alpha_1, \ldots$ be a fair assembly sequence. Then its result γ is terminal.

Corollary: For every assembly α, there is a terminal assembly γ such that $\alpha \to \gamma$.

Proof:

1. Suppose for the sake of contradiction that γ is not terminal, i.e., it has frontier location $p \in \partial\gamma$; note in particular $p \notin S_\gamma$.
2. Simpler if assembly sequence is finite:
   1. in this case, $\gamma = \alpha_{k-1}$, so p never receives a tile.
   2. Thus the assembly sequence is not fair. (*there is no $j > k-1$ such that $p \in S_{\alpha j}$*)
3. Now assume assembly sequence is infinite. (*actually, rest of proof works in finite case*)
4. Since $p \in \partial\gamma$, there are positions adjacent to p with enough strength to bind a tile t. Let N be the set of these positions. Note N is finite since p has at most four neighbors.
5. Since $S_\gamma = \bigcup_i S_{\alpha i}$, there exists i such that $N \subseteq \partial\alpha_i$ (*after some finite number of tile attachments, all of the positions in N are on the frontier of the current assembly*)
6. Thus $p \in \partial\alpha_i$. (*the tile t can attach to α_i, reached after only i steps*)
7. By fairness, there exists j such that $p \in S_{\alpha j} \subseteq S_\gamma$ (*eventually p gets a tile*), which contradicts the claim that $p \notin S_\gamma$. **QED**

Proof: Pick any fair assembly sequence $\alpha=\alpha_0, \alpha_1, \ldots$; its result γ is terminal and $\alpha \to \gamma$. **QED**

Concrete example of simulation algorithm creating a fair assembly sequence?

How computationally powerful are self-assembling tiles?

Turing machines

state ≈ line of code

initial state = s

current symbol — next symbol

current state — next state — next move

s,0: q,0,→

q,0: t,1,←

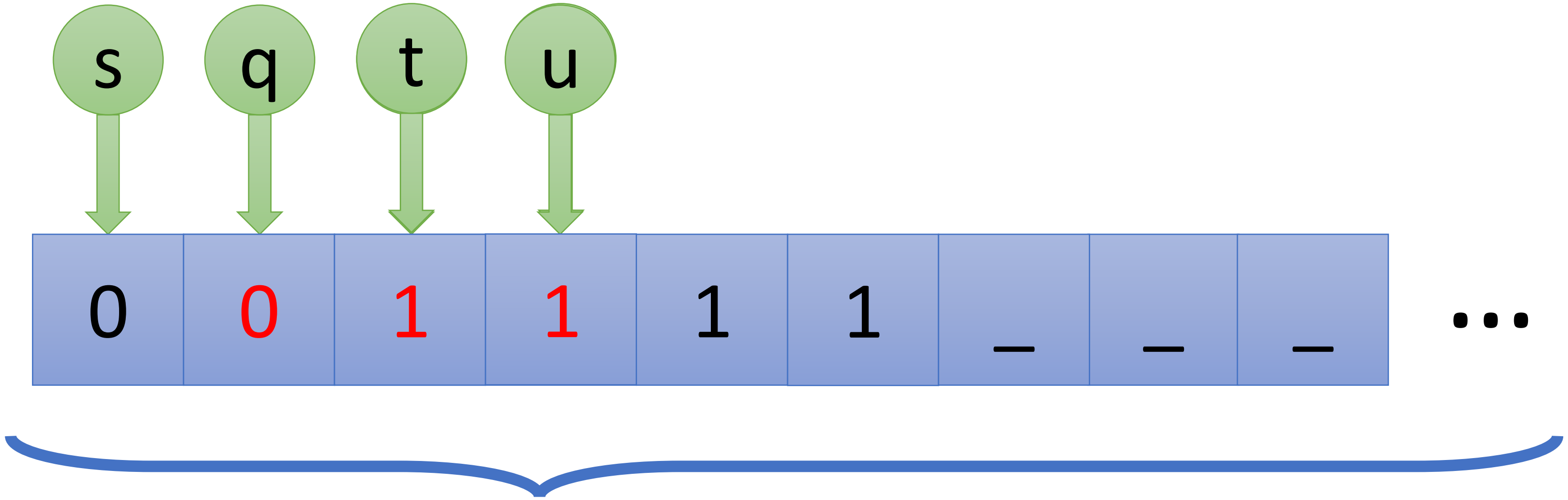
q,1: s,0,→

t,0: u,1,→

u,1: HALT

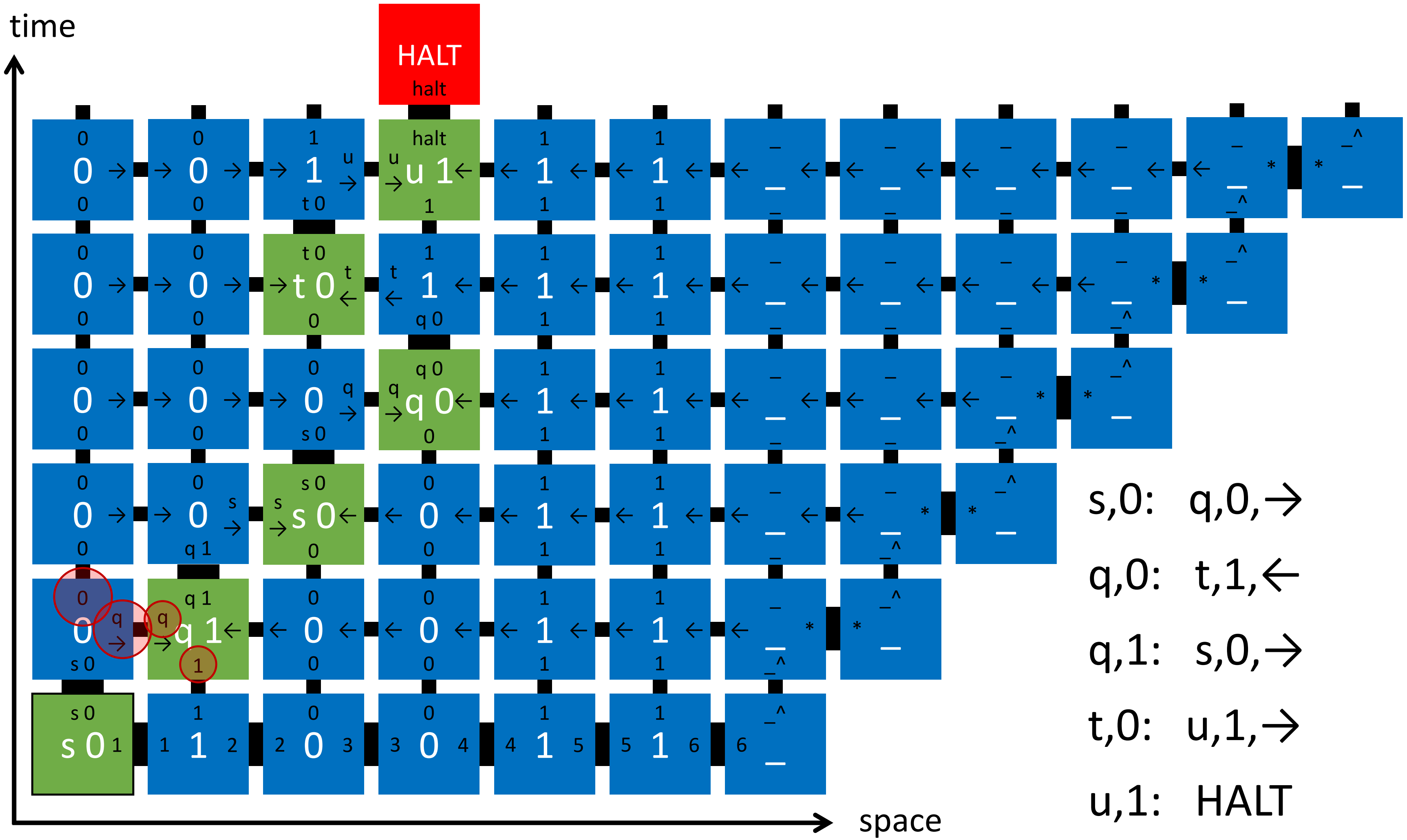
transitions

(instructions)



tape ≈ memory

Tile assembly is Turing-universal



s,0: q,0,→

q,0: t,1,←

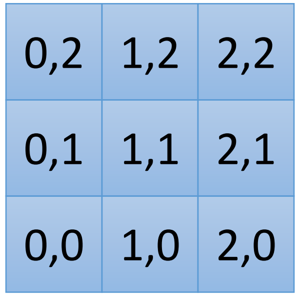
q,1: s,0,→

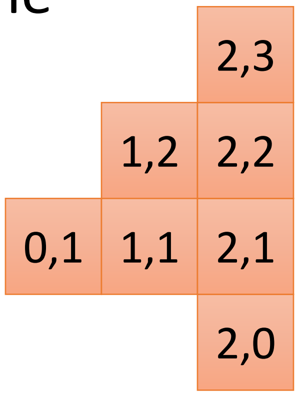
t,0: u,1,→

u,1: HALT

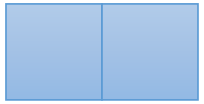
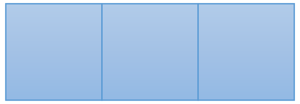
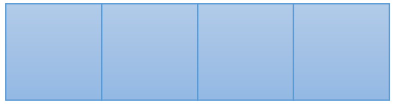
Complexity of self-assembled shapes

- We've seen how use algorithmic tiles to:
  - self-assemble n x n squares with "few" tile types $O(\log n / \log \log n)$
  - simulate a Turing machine that grows a "wedge" describing its space-time configuration history
- What other shapes can be self-assembled?
  - Define a shape to be a finite, connected subset of $\mathbb{N}^2$.
  - Any shape with n points can be self-assembled with <u>at most</u> how many tile types? n



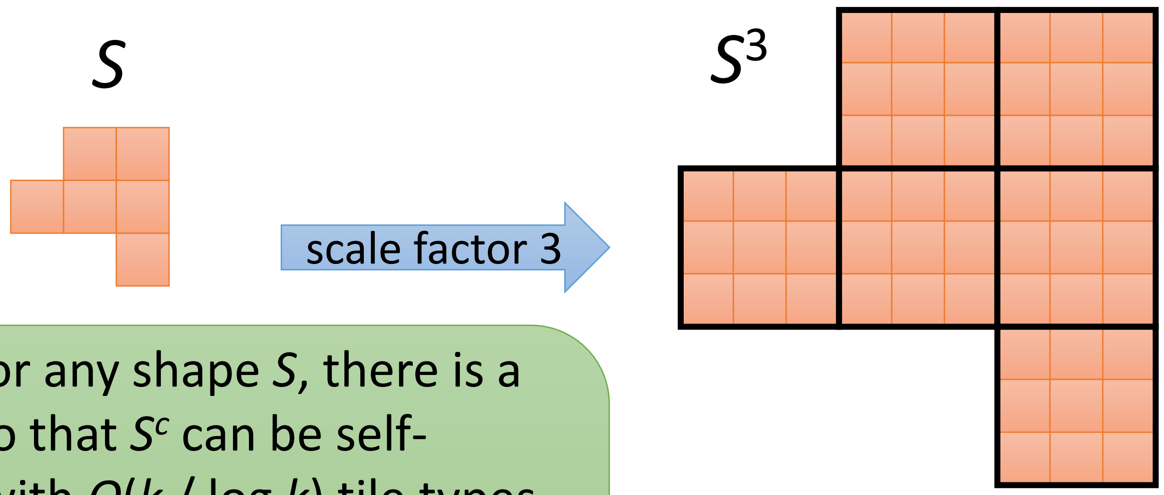


- Is there an infinite family of shapes $S_1, S_2, \ldots$, with $|S_n| = n$, such that each S_n requires <u>at least</u> n tile types to self-assemble?

$S_1 =$  $S_2 =$  $S_3 =$  $S_4 =$  …

Complexity of self-assembled shapes

Suppose we are content to create a scaled up version of the shape:



Theorem: For any shape *S*, there is a constant *c* so that S^c can be self-assembled with $O(k / \log k)$ tile types, where *k* is the length in bits of the shortest program (input to a universal Turing machine) that, on input (x,y), indicates whether $(x,y) \in S$.

Theorem (that we won't prove): This is optimal! No smaller tile system could self-assemble any scaling of *S*. If one existed, we could turn it into a program with < *k* bits "describing" *S* in this way. (*Why?*)

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

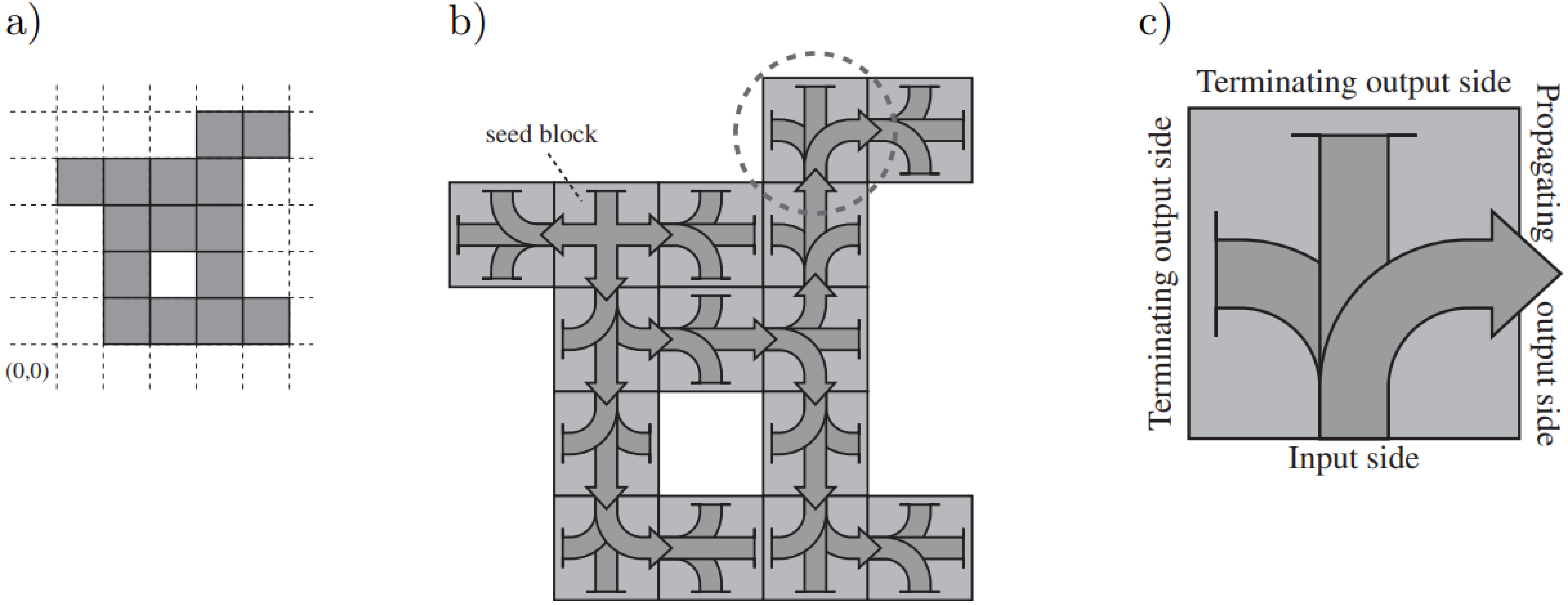
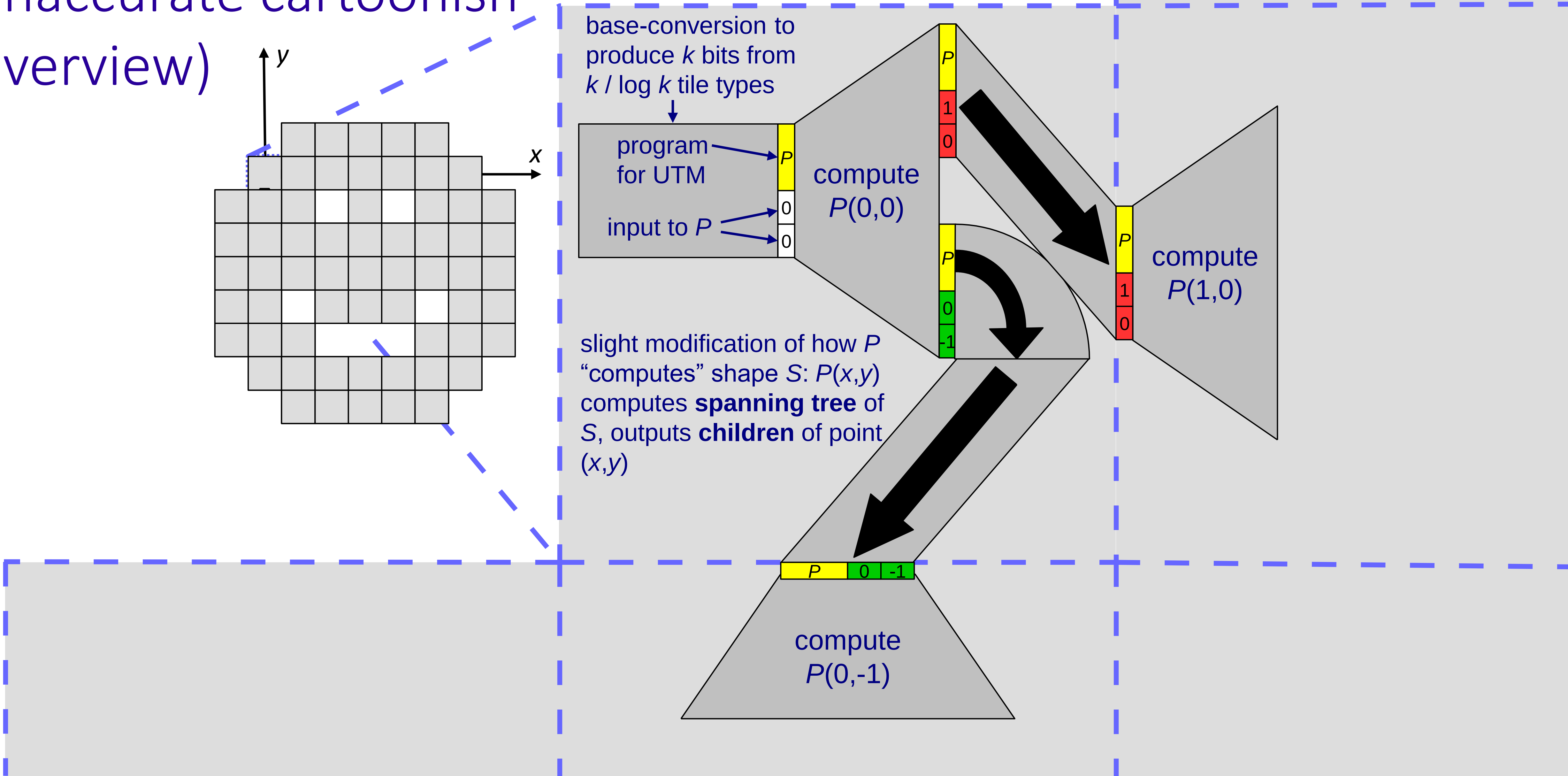


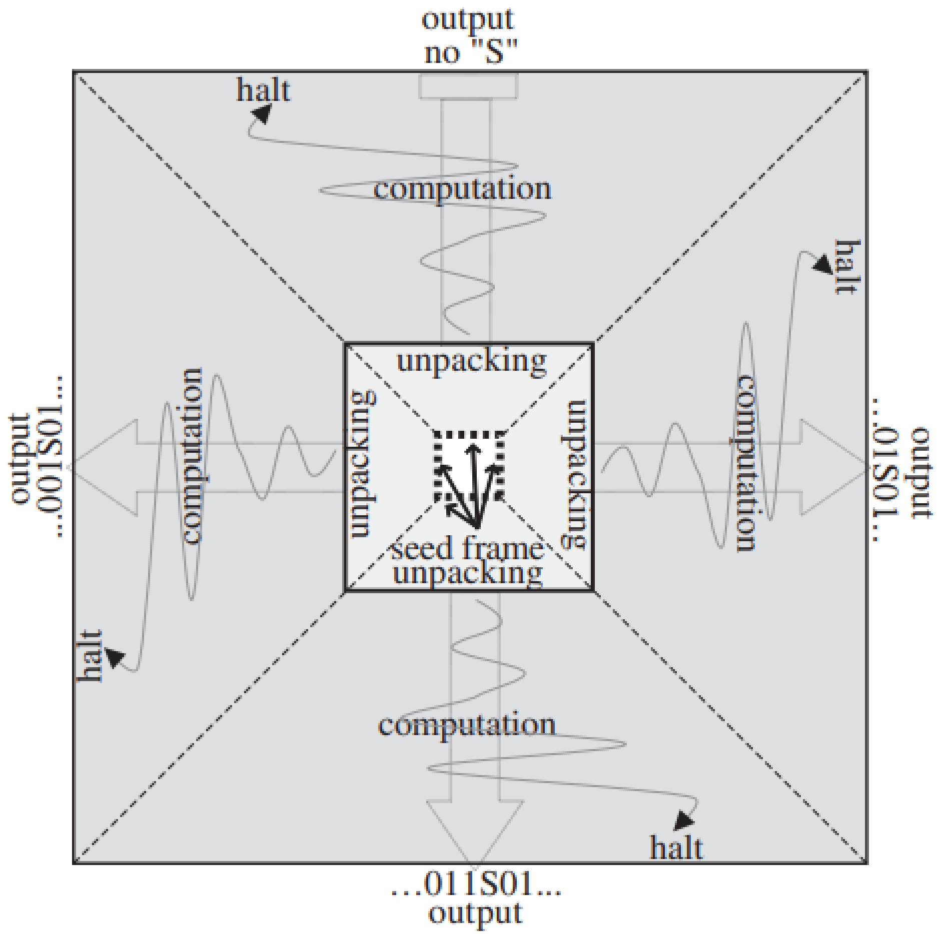
FIG. 5.1. *Forming a shape out of blocks:* (a) *A coordinated shape S.* (b) *An assembly composed of $c \times c$ blocks that grow according to transmitted instructions such that the shape of the final assembly is $\tilde{S}$ (not drawn to scale). Arrows indicate information flow and order of assembly. The seed block and the circled growth block are schematically expanded in Figure 5.2.* (c) *The nomenclature describing the types of block sides.*

Programming a shape (inaccurate cartoonish overview)

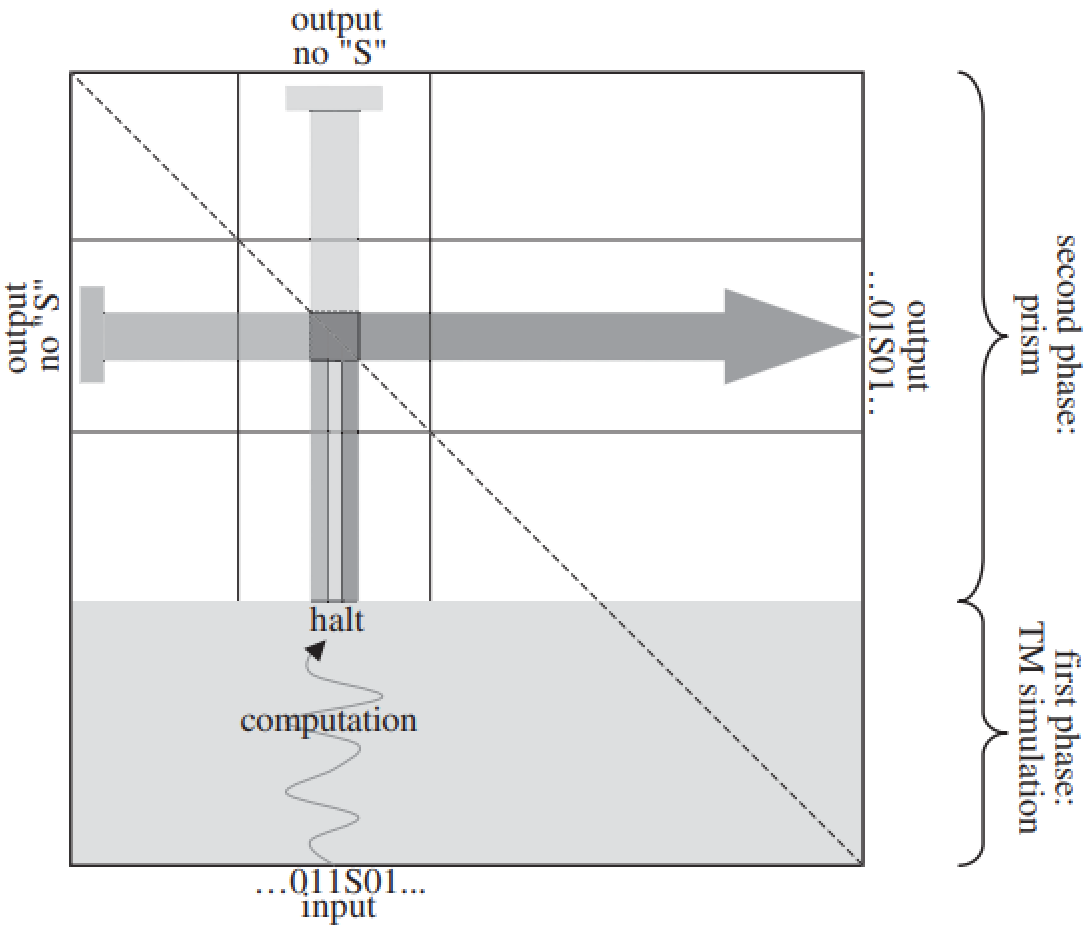


More accurate detailed overview

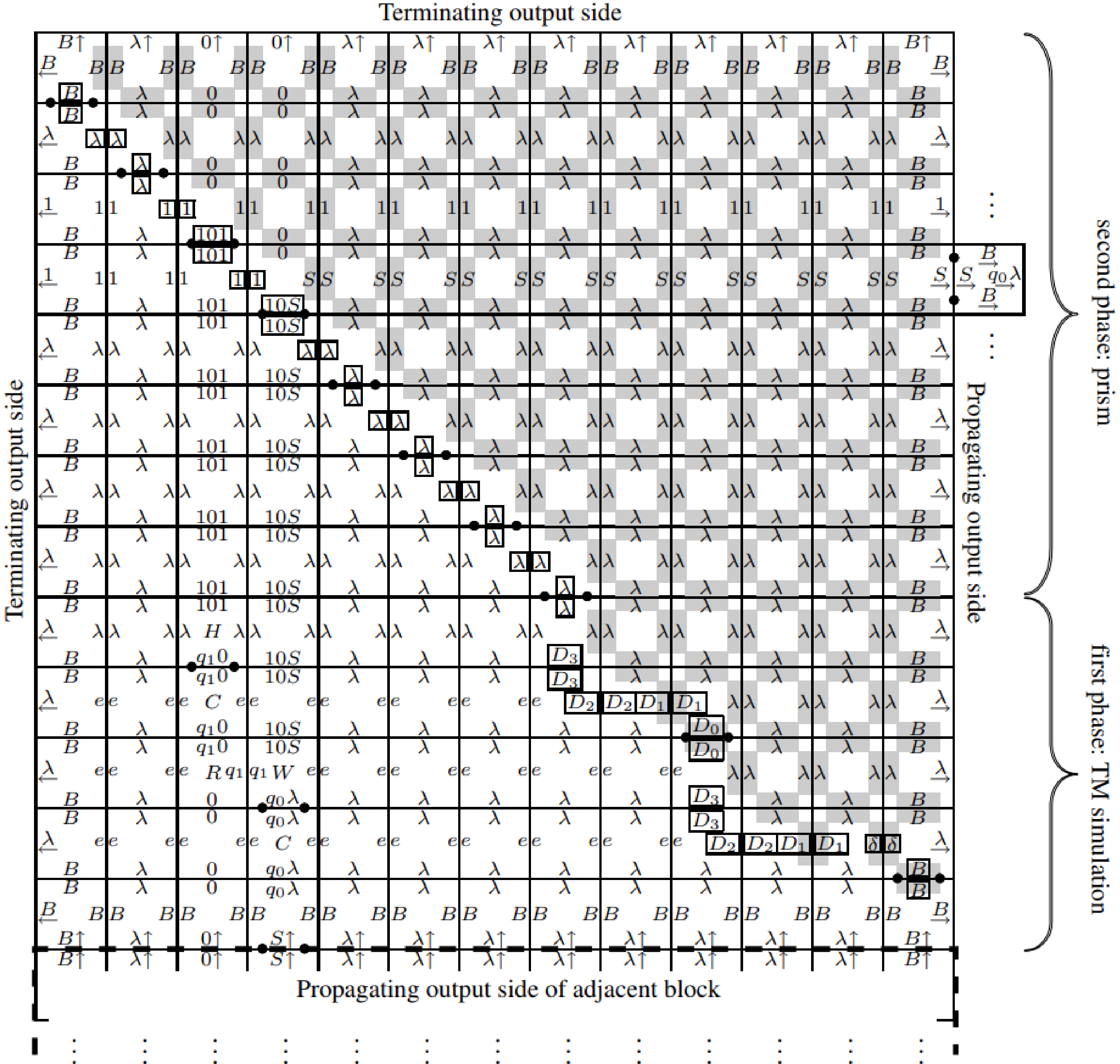
seed block



growth block



fully-detailed example of growth block



Two interpretations

as stated for single seed tile:

Theorem: For any shape S, there is a constant c so that S^c can be self-assembled with <u>$O(k$ / log $k)$ tile types</u>, where k is the length in bits of the shortest program (input to a universal Turing machine) that, on input (x,y), indicates whether $(x,y) \in S$.

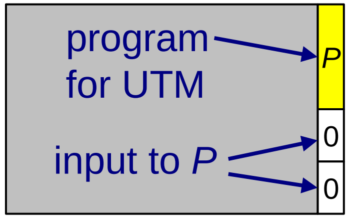
most of the tile complexity is encoding the binary string representing the program P that encodes shape S, and <u>$O(1)$ tile types</u> can read that string and self-assemble S^c from it.

i.e., T is a **universal** set of tile types that can self-assemble any shape, by giving it the right seed.

alternative statement for larger seed:

Theorem: There is a <u>single</u> set T of tile types (<u>$O(1)$ tile types</u>), so that, for any finite shape S, there a constant c and a seed assembly σ_S "encoding" S, so that T self-assembles S^c from σ_S.

σ_S =



Strict and weak self-assembly

Computability-theoretic questions about self-assembly

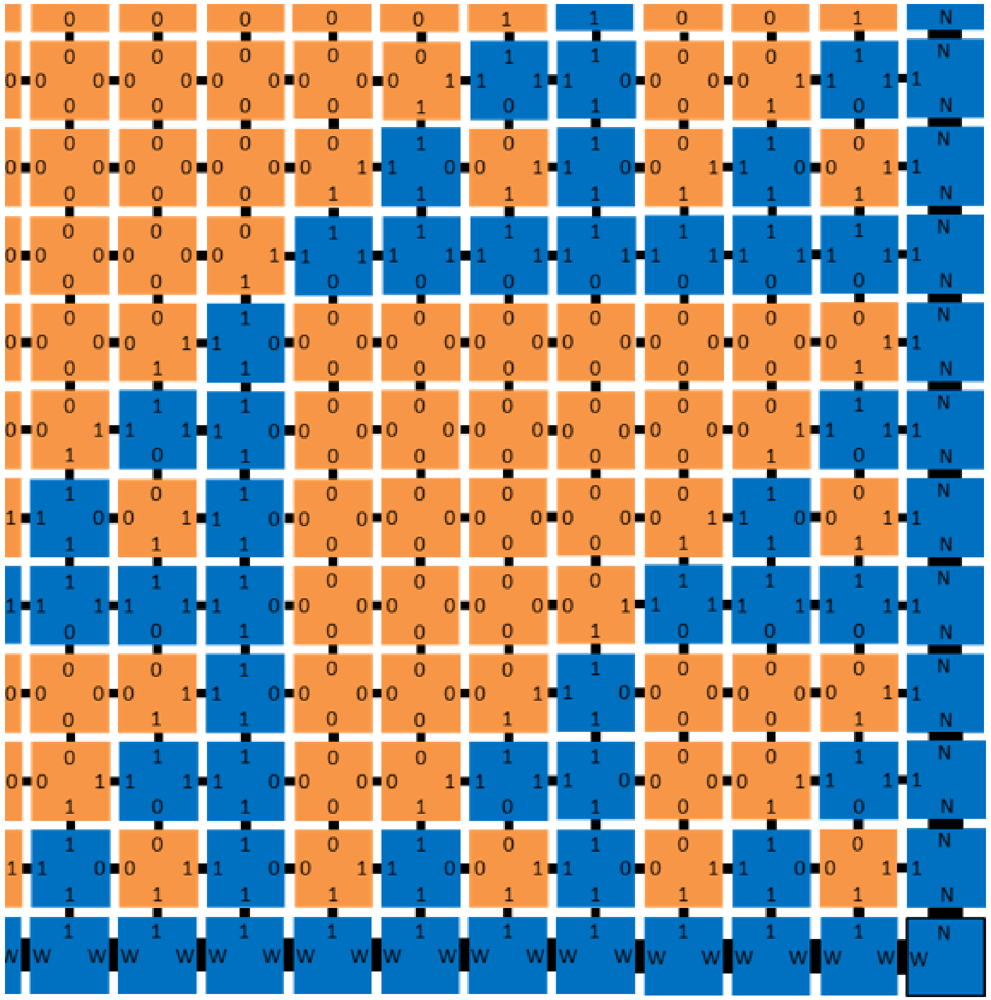
Strict and weak self-assembly

Recall:

Let $X \subseteq \mathbb{Z}^2$ be a **shape**, a connected subset of $\mathbb{Z}^2$. Θ **strictly self-assembles** X if, for all $\alpha \in A_{\Box}[\Theta]$, $S_\alpha = X$.
(*every terminal producible assembly has shape X*)

Let $X \subseteq \mathbb{Z}^2$. Θ **weakly self-assembles** X if there is a subset $B \subseteq T$ (the "blue tiles") such that, for all $\alpha \in A_{\Box}[\Theta]$, $X = \alpha^{-1}(B)$.
(*every terminal producible assembly puts blue tiles exactly on X.*)

Tile system on right strictly self-assembles the whole second quadrant, and it weakly self-assembles the discrete Sierpinski triangle.



Strict self-assembly

Observation: There is an infinite shape $S \subseteq \mathbb{Z}^2$ that cannot be strictly self-assembled by any tile system.

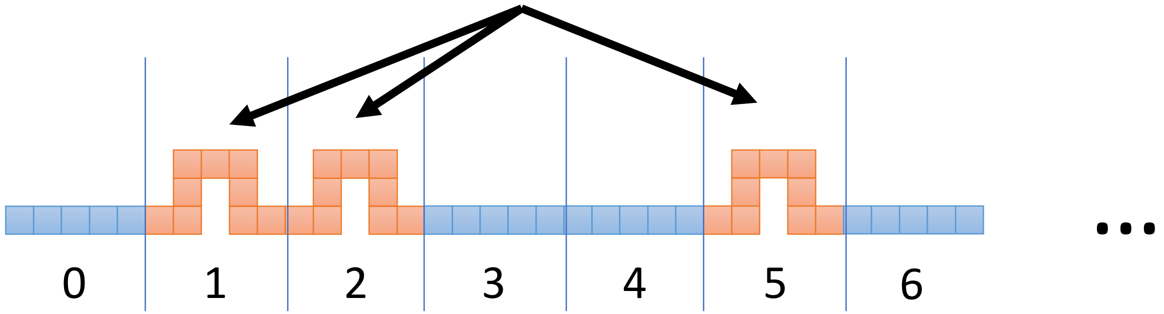
Proof:
There are uncountably many shapes but only countably many tile systems.

Observation is *non-constructive*: Doesn't tell us what is the shape *S*. Can we devise a concrete example of a shape that cannot be strictly self-assembled?

Homework problem: you will show that any shape $S \subseteq \mathbb{Z}^2$ that can be strictly self-assembled is also computably enumerable.

Use that fact now to define an explicit shape that cannot be strictly self-assembled.

path in block *n* has a "turnout" if and only if *n*'th Turing machine halts on empty input

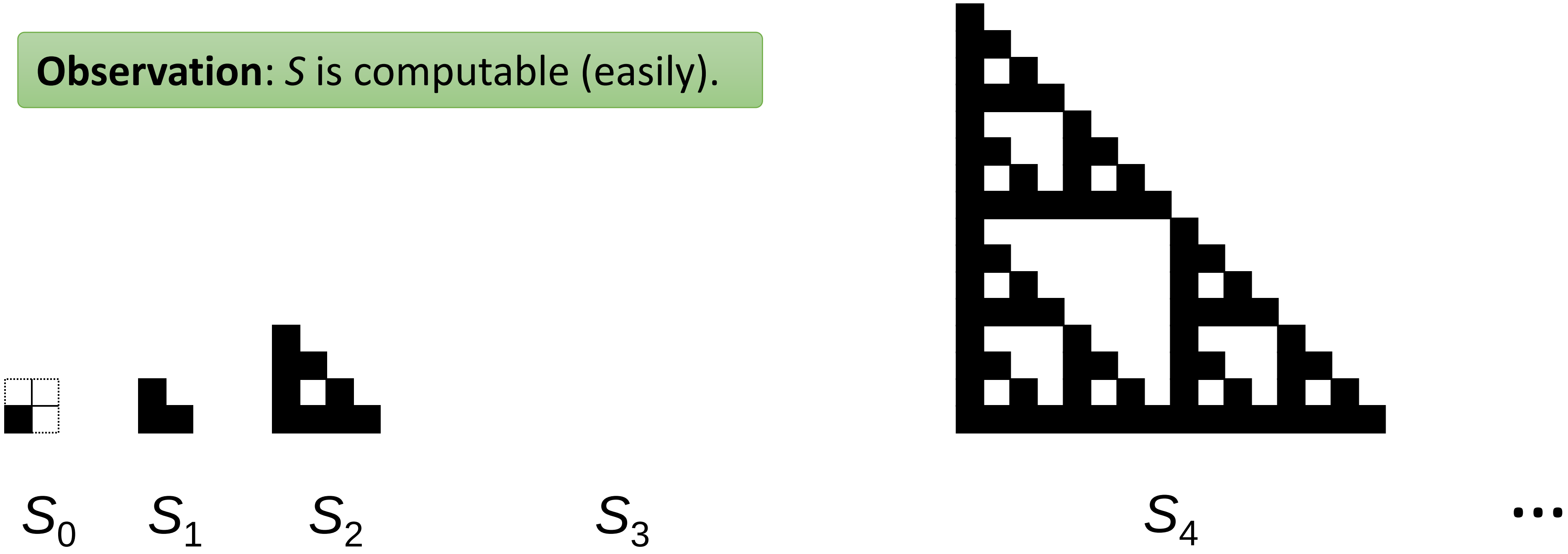


Question: Is there a computable shape $S \subseteq \mathbb{Z}^2$ that cannot be strictly self-assembled?

A famous fractal

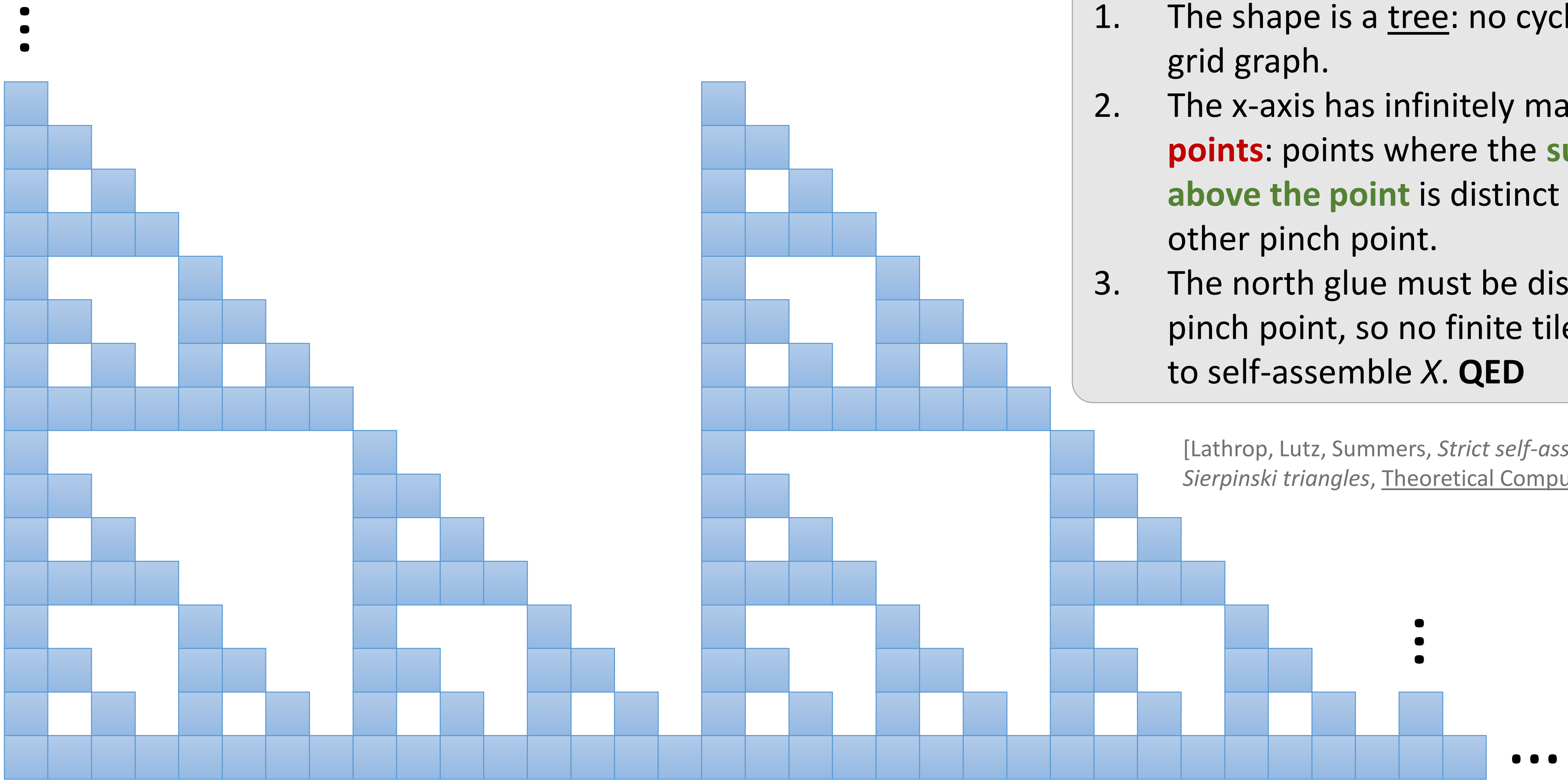
- Let S_0 = { (0,0) }
- Let V = { (0,0), (0,1), (1,0) } be three vectors for “recursive translation”.
- *S* is known as the *discrete Sierpinski triangle*…

Observation: *S* is computable (easily).



[slide credit:
Scott Summers]

The discrete Sierpinkski triangle cannot be strictly self-assembled



Proof:

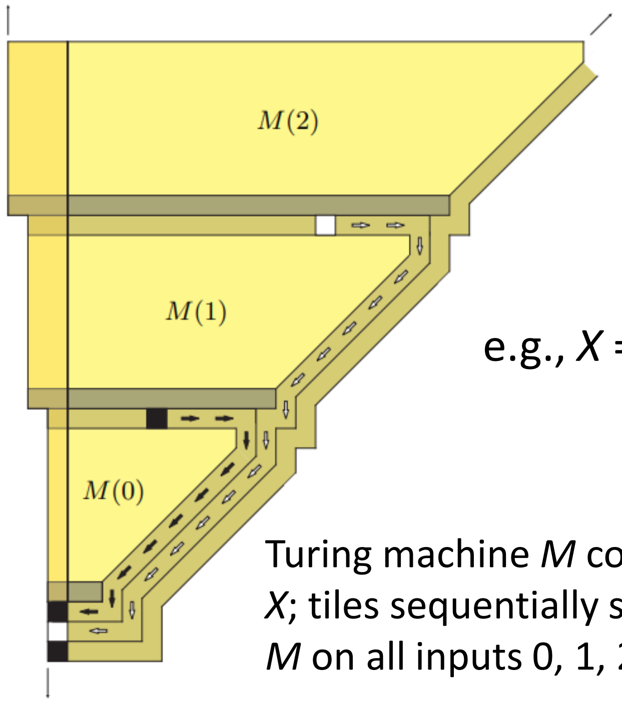
1. The shape is a tree: no cycles in the grid graph.
2. The x-axis has infinitely many **pinch points**: points where the **subtree above the point** is distinct from any other pinch point.
3. The north glue must be distinct at each pinch point, so no finite tile set suffices to self-assemble *X*. **QED**

[Lathrop, Lutz, Summers, *Strict self-assembly of discrete Sierpinski triangles*, Theoretical Computer Science 2009.]

Weak self-assembly

Theorem: Some computable sets $X \subseteq \mathbb{Z}^2$ cannot be weakly self-assembled.

Theorem: Every computable set $X \subseteq \mathbb{N}$, "embedded straightforwardly" in $\mathbb{Z}^2$, can be weakly self-assembled.

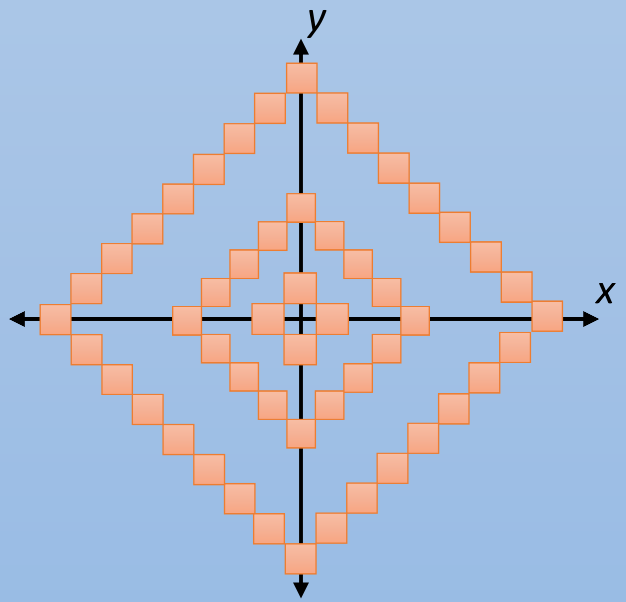


e.g., $X = \{0, 2, \ldots\}$

Turing machine M computes X; tiles sequentially simulate M on all inputs 0, 1, 2, …,

Proof:

1. The Time Hierarchy Theorem says there is a computable set $A \subseteq \{1\}^*$ not computable in $O(n^4)$ time.
2. Let $R = \{|x| : x \in A\}$ be the set of lengths of strings in A.
3. Define $X \subseteq \mathbb{Z}^2$ to be the set of "concentric diamonds" whose L_1 radii are in R, e.g., if $R = \{1, 4, 8, \ldots\}$



4. Suppose X could be weakly self-assembled. Then simulating self-assembly for $(2n)^2$ steps necessarily places a tile at some point at L_1 radius n from the origin; the tile's color tells us whether $n \in R \Leftrightarrow 1^n \in A$.
5. This can be done in time $O(n^4)$ time (why?), a contradiction. **QED**

[Patitz, Summers, *Self-assembly of decidable sets*, UCNC 2008.]

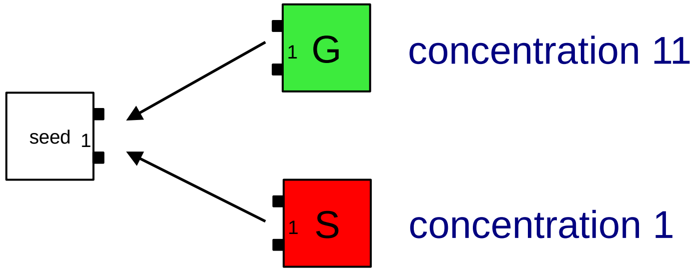
[Lathrop, Lutz, Patitz, Summers, *Computability and Complexity in Self-Assembly*, CiE 2008.]

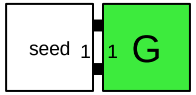
Randomized self-assembly

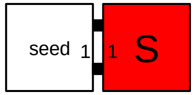
Tile complexity of universal shape construction

- Recall: if we can have a seed structure encoding a shape *S* (in a binary string $x \in \{0,1\}^*$, in glues on one side), we can self-assemble some scaling S^c of S with $O(1)$ additional tile types that read and interpret *x*.
- $\Theta(K(x) / \log K(x))$ tile types are necessary and sufficient to create *x* from a single seed tile in the aTAM. ($K(x)$ = length in bits of shortest program for universal Turing machine that prints *x*)
- We'll see how to get this down to $O(1)$ with high probability by *concentration programming*.
    - i.e., move the effort from designing new tile types to (*the plausibly simpler lab step of*) altering concentrations of existing tile types

Nondeterministic binding

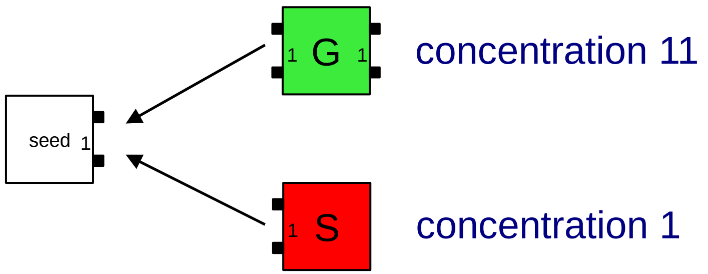


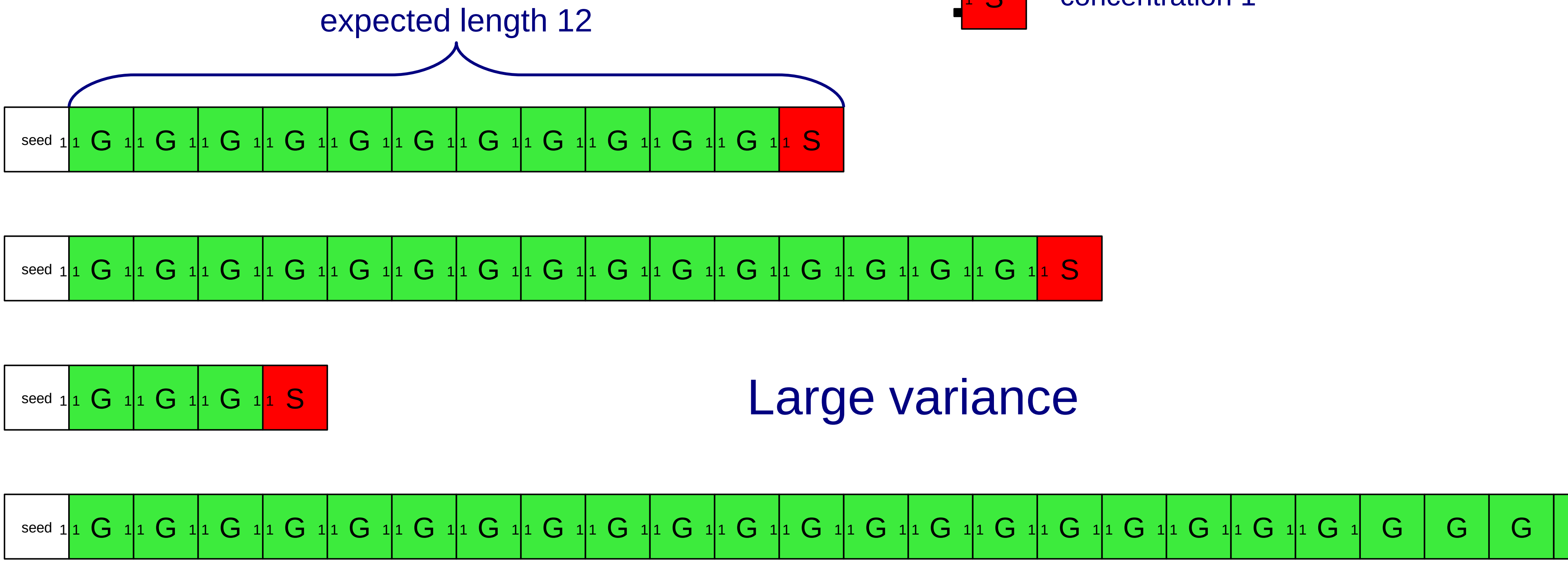
Pr[] = 11/12

Pr[] = 1/12

Programming polymer length with concentrations

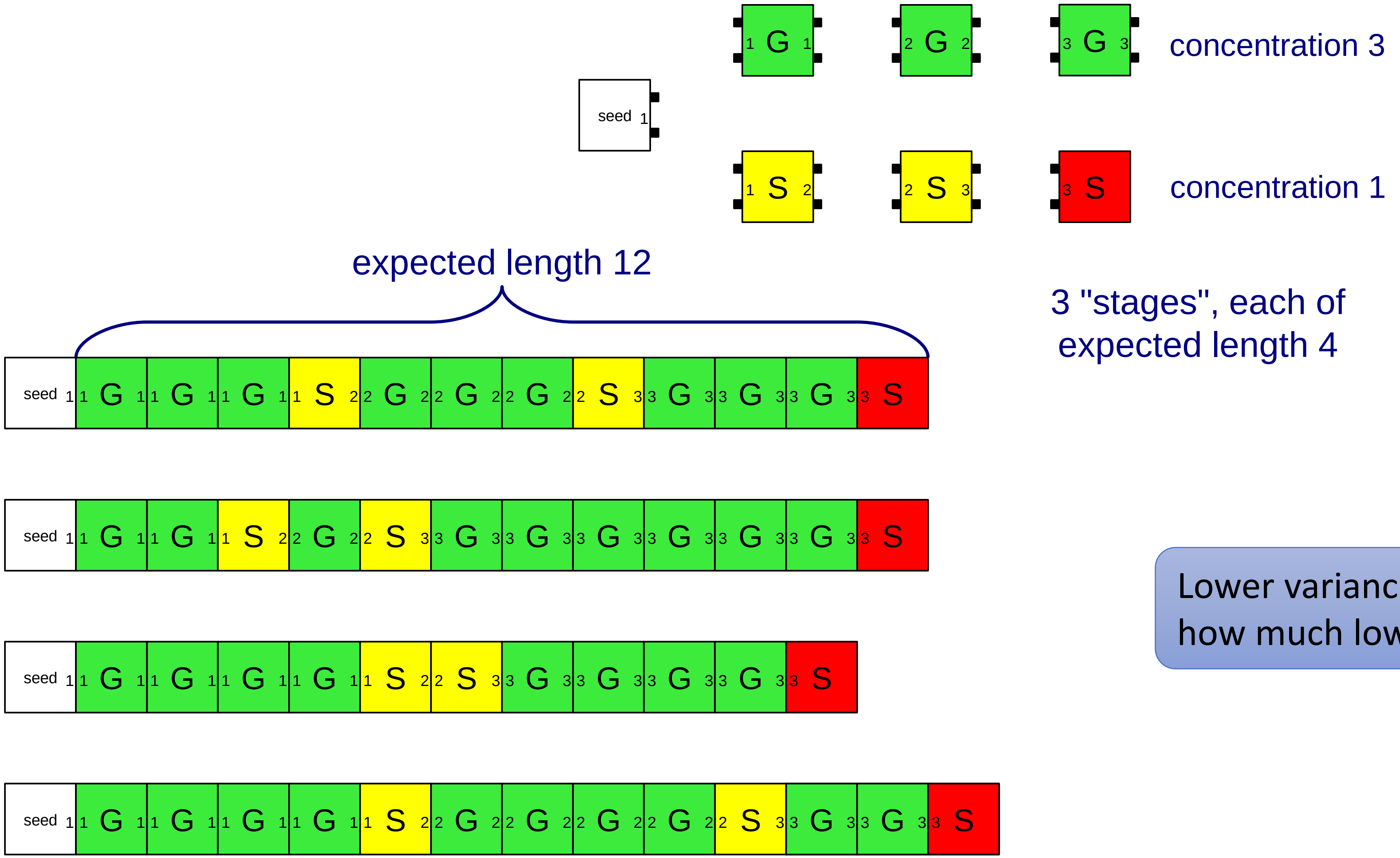
[Becker, Rapaport, Rémila, *FSTTCS* 2006]





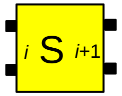
Large variance

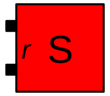
Programming polymer length (improved)



Lower variance…
how much lower?

Bounding the probability the length deviates much from its mean

- r total stages, each with Pr[next tile 

increments stage] = p.
- Let $\mathbf{L}(r,p)$ = total length; number of tile attachments until attaching 

- Expected total length $E[\mathbf{L}(r,p)] = r / p$.
- <u>Recall</u>: a binomial random variable $\mathbf{B}(n,p)$ = number of heads when flipping a coin n times, with Pr[heads] = p. $E[\mathbf{B}(n,p)] = np$.
- for any n,r,p: $\Pr[\mathbf{L}(r,p) \le n] = \Pr[\mathbf{B}(n,p) \ge r]$

flipping a coin until the r'th heads requires $\le n$ flips $\Leftrightarrow$ flipping a coin n times results in $\ge r$ heads

- similarly, $\Pr[\mathbf{L}(r,p) \ge n] = \Pr[\mathbf{B}(n,p) \le r]$

Chernoff bound

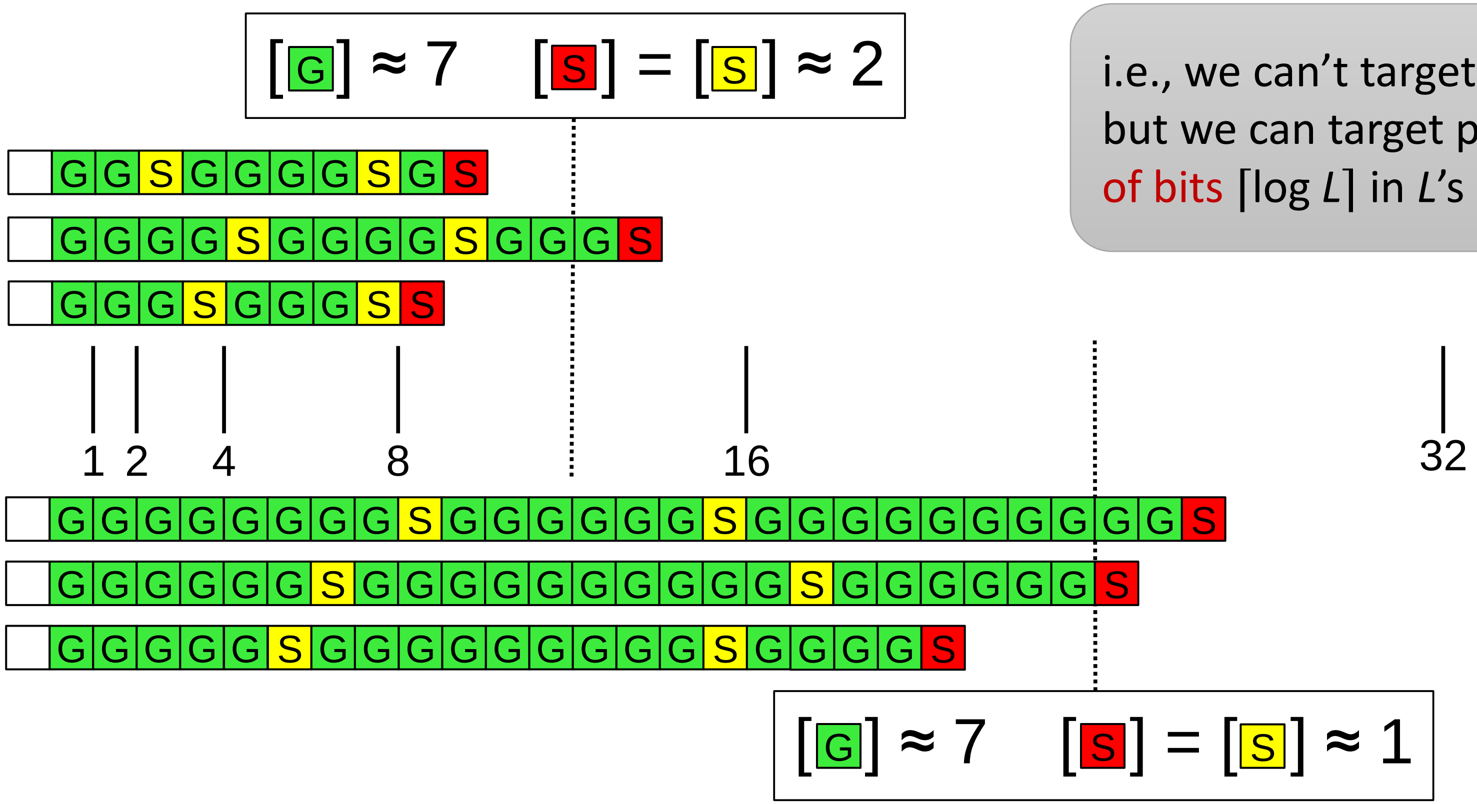
Chernoff bound: For a binomial random variable $\mathbf{B}(n,p)$ (recall $E[\mathbf{B}(n,p)] = np$), and for any $0 < \delta < 1$,
$\Pr[\mathbf{B}(n,p) > (1+\delta)np] < \exp(-\delta^2 np/3)$
$\Pr[\mathbf{B}(n,p) < (1-\delta)np] < \exp(-\delta^2 np/2)$

Let $\delta \approx 0.27$ and set p such that $r/p(1-\delta) = 2^k$.
Let $\delta' \approx 0.44$: then $r/p(1+\delta') \approx 2^{k-1}$.
Applying this to our setting gives
$\Pr[\mathbf{L}(r,p)$ is not between 2^{k-1} and $2^k] < 2 \cdot 0.9421^r$

Programming polymer length (improved)

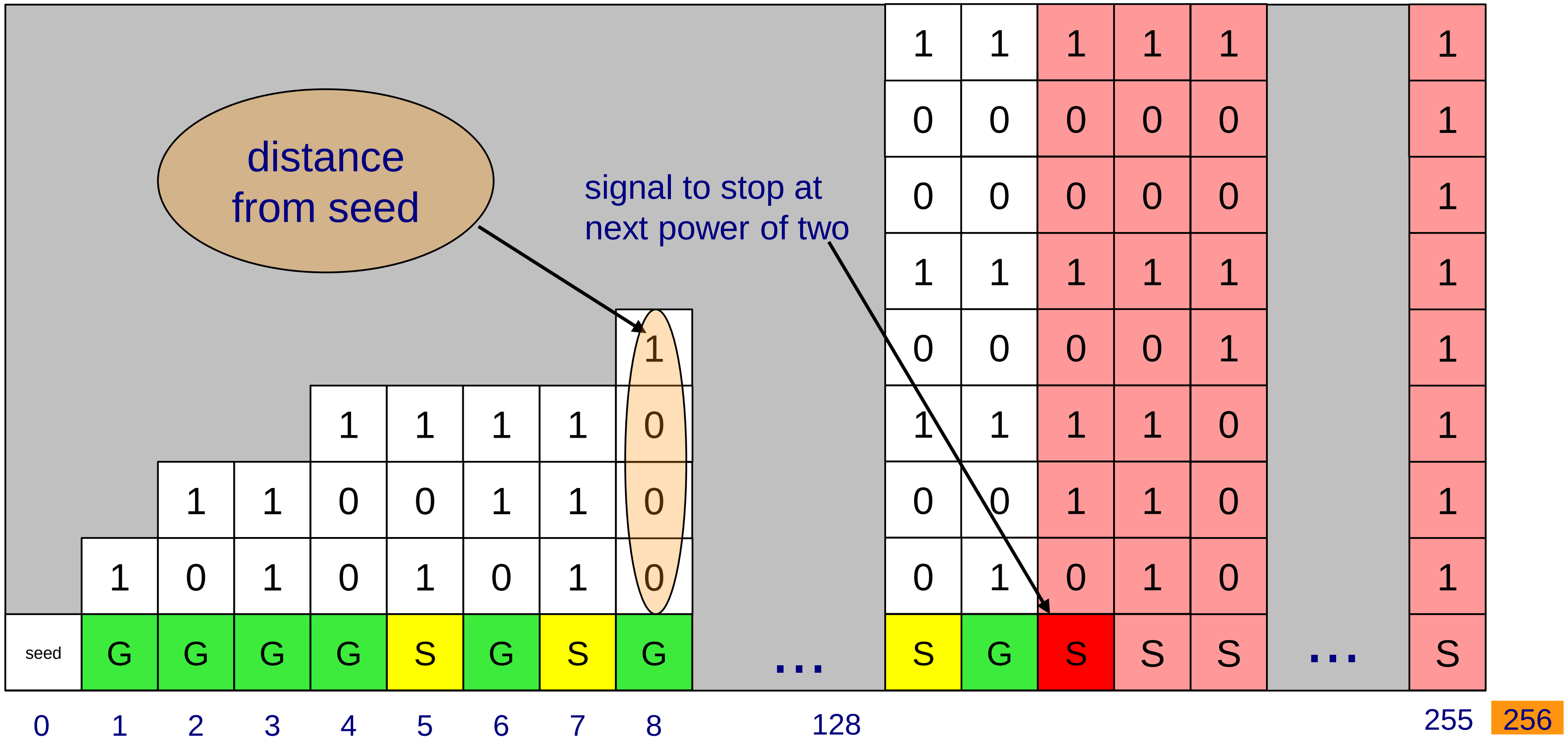
if r = 90 stages, expected length midway in $[2^{k-1}, 2^k)$

→ with probability > 99%, **actual** length in $[2^{k-1}, 2^k)$

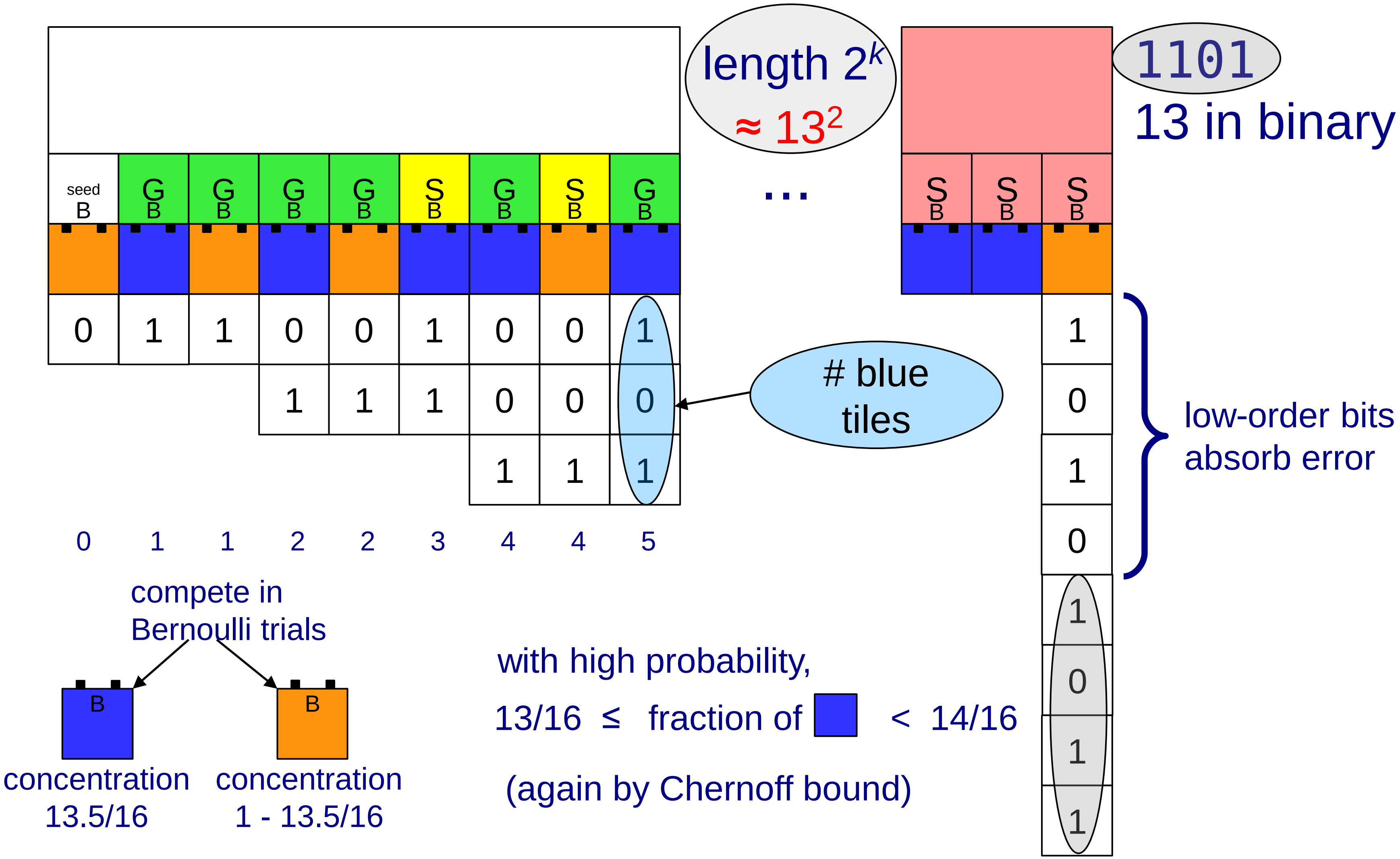


i.e., we can’t target a precise length *L*, but we can target precisely the number of bits ⌈log *L*⌉ in *L*’s binary expansion.

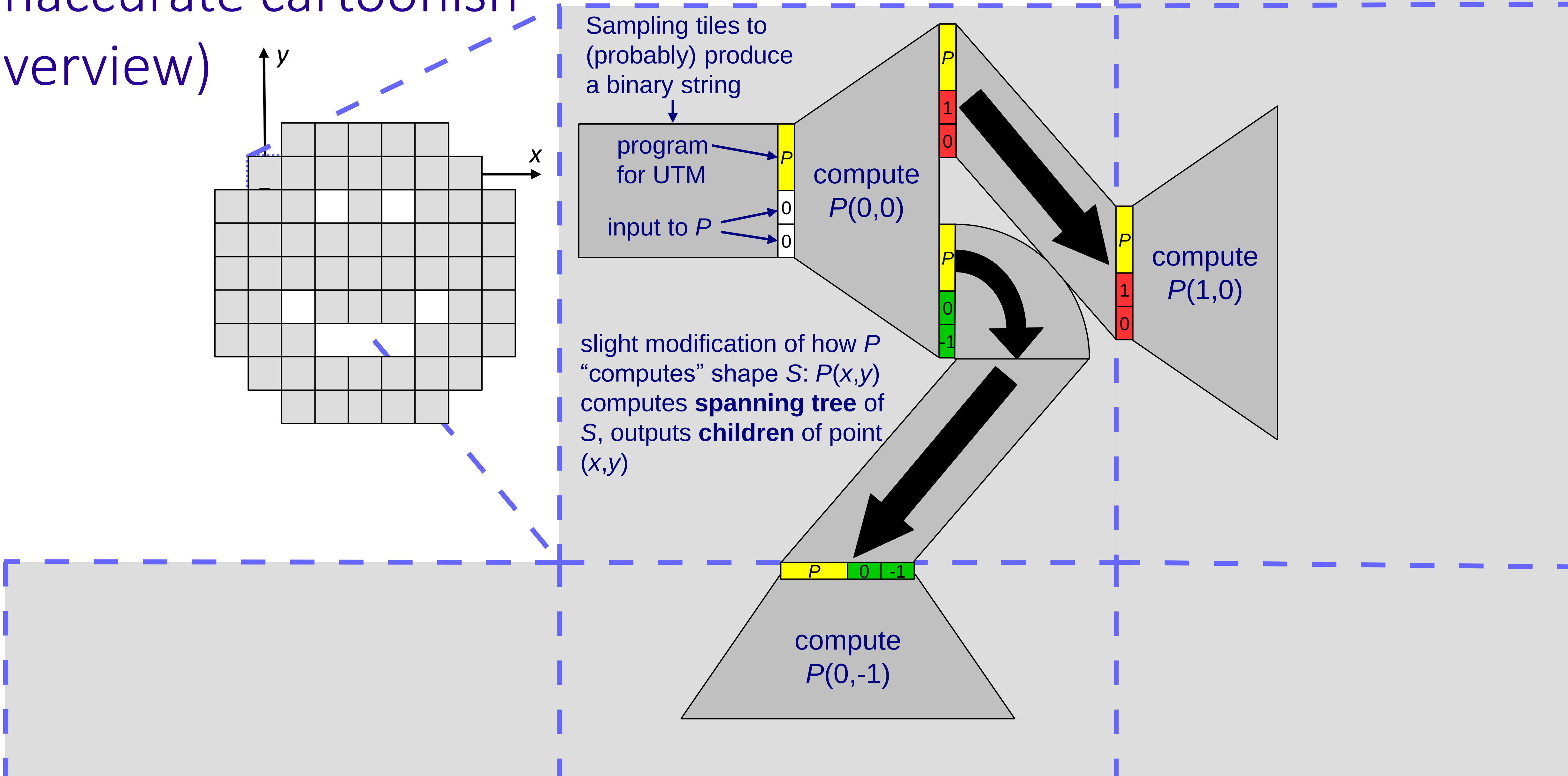
Programming polymer length 2^k precisely



Programming a binary string

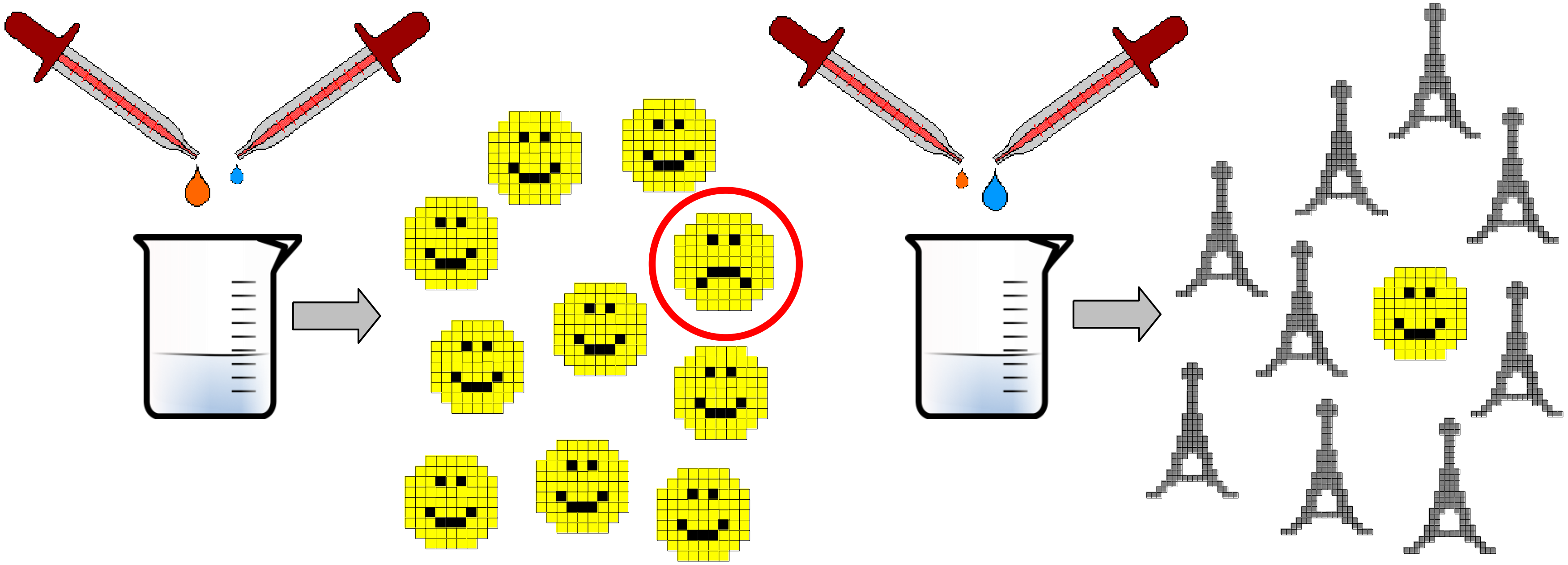


Programming a shape (inaccurate cartoonish overview)



Universal self-assembling molecules

A **fixed** set of tile types can assemble *any* finite (scaled) shape (with high probability) by mixing them in the right concentrations.



[Doty, *Randomized self-assembly for exact shapes*, SICOMP 2010, FOCS 2009]

Other plausible modifications of aTAM model that can reduce tile complexity

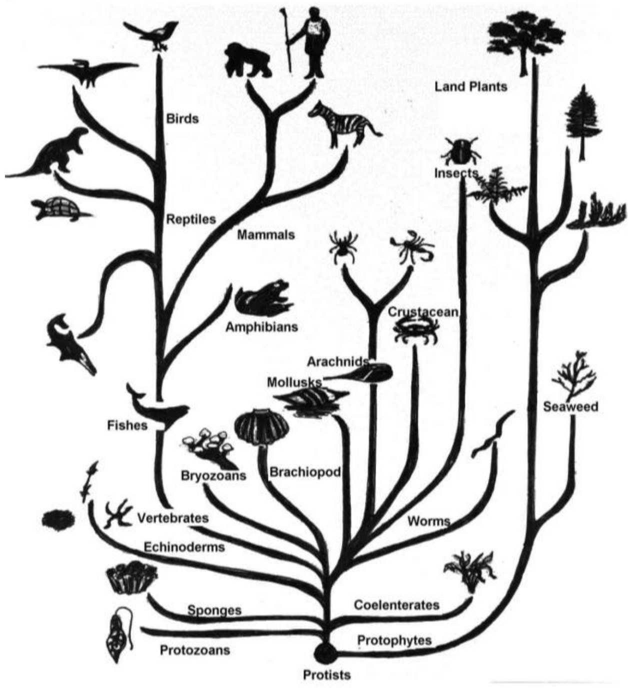
- staged self-assembly:
  - https://doi.org/10.1007/s11047-008-9073-0
- temperature programming:
  - https://dl.acm.org/doi/10.5555/1109557.1109620

The power of nondeterminism in self-assembly

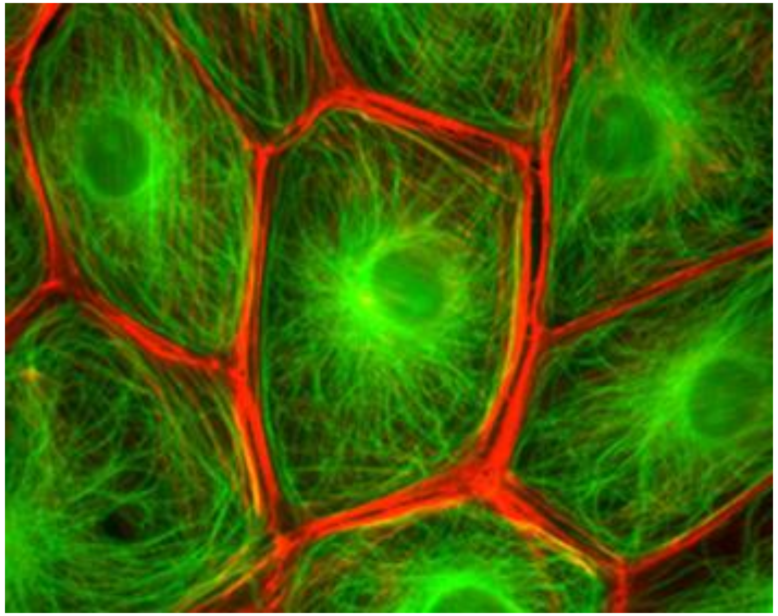
Can nondeterminism help to self-assemble shapes?

Nondeterminism in Biology

Genetic mutation



Cytoskeleton formation



Nondeterminism can allow complex structures to be created from a compact encoding.

Nondeterminism in Computer Science

Algorithm types:

Nondeterministic:
flips coins; magical

Randomized:
flips coins; realistic

Trivially nondeterministic (“pseudodeterministic”):
flips coins, but *final output* independent of flip results

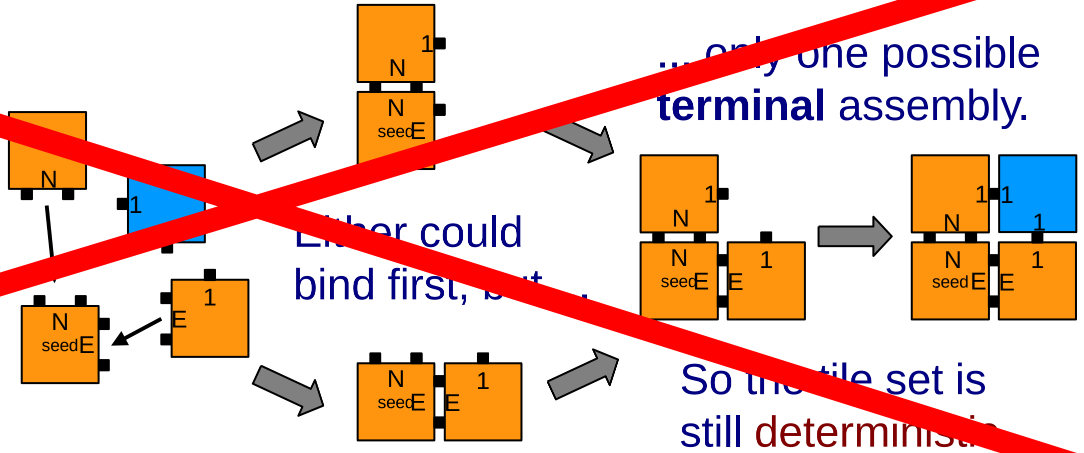
Deterministic: entire computation uniquely determined by input

Power

Nondeterminism in Self-Assembly

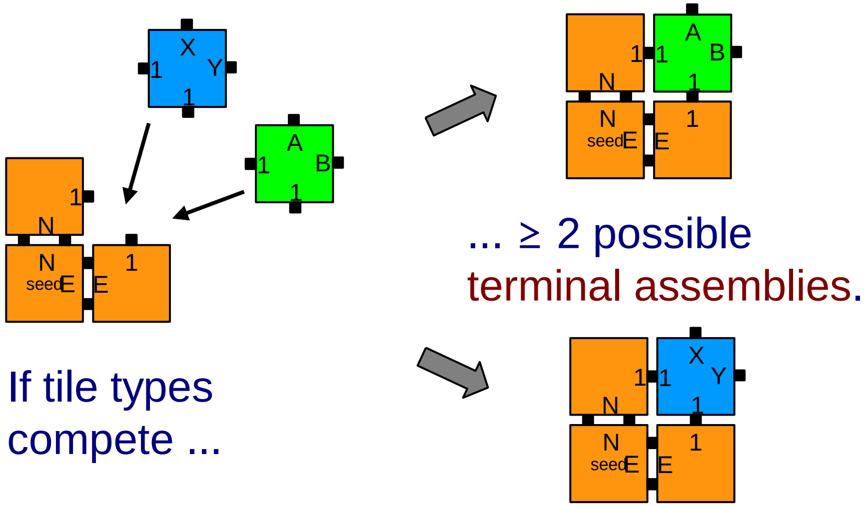
Perhaps:

≥ 2 potential binding sites



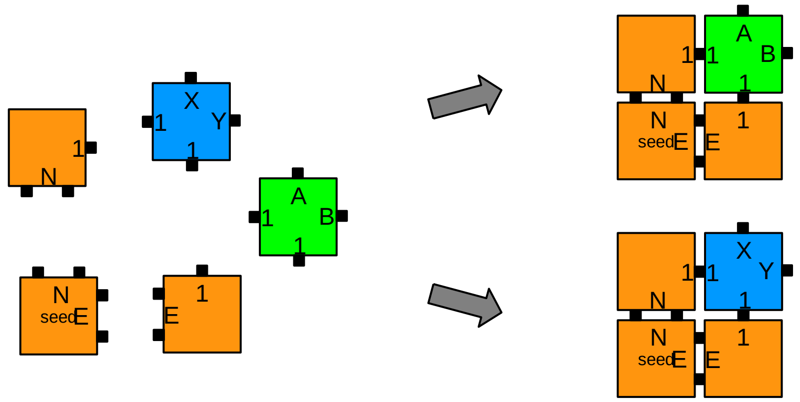
More meaningful:

at a *single* binding site, ≥ 2 tile types attachable



Nondeterminism in Self-Assembly

- A tile set is **deterministic** if it has only one terminal assembly (map of tile types to points).
- This tile set has multiple terminal assemblies, *but* they all have the same shape.

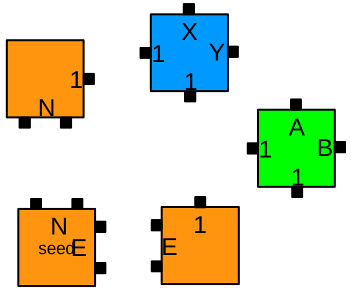


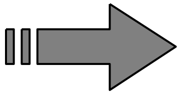
- The tile set **self-assembles** a 2 x 2 square.

Power of Nondeterminism

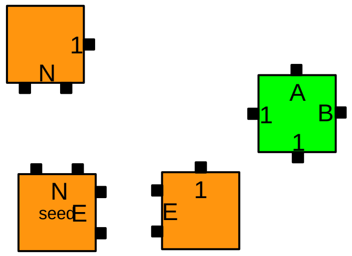
Question: Let *S* be a finite shape self-assembled by some nondeterministic tile set. Does some deterministic tile set also self-assemble *S*?

In this example, we can convert this nondeterministic tile set that self-assembles a 2 x 2 square ...





... to this deterministic tile set that self-assembles the same shape.



In general???

Power of Nondeterminism

Question: Let *S* be a finite shape self-assembled by some nondeterministic tile set. Does some deterministic tile set als

Answer: T

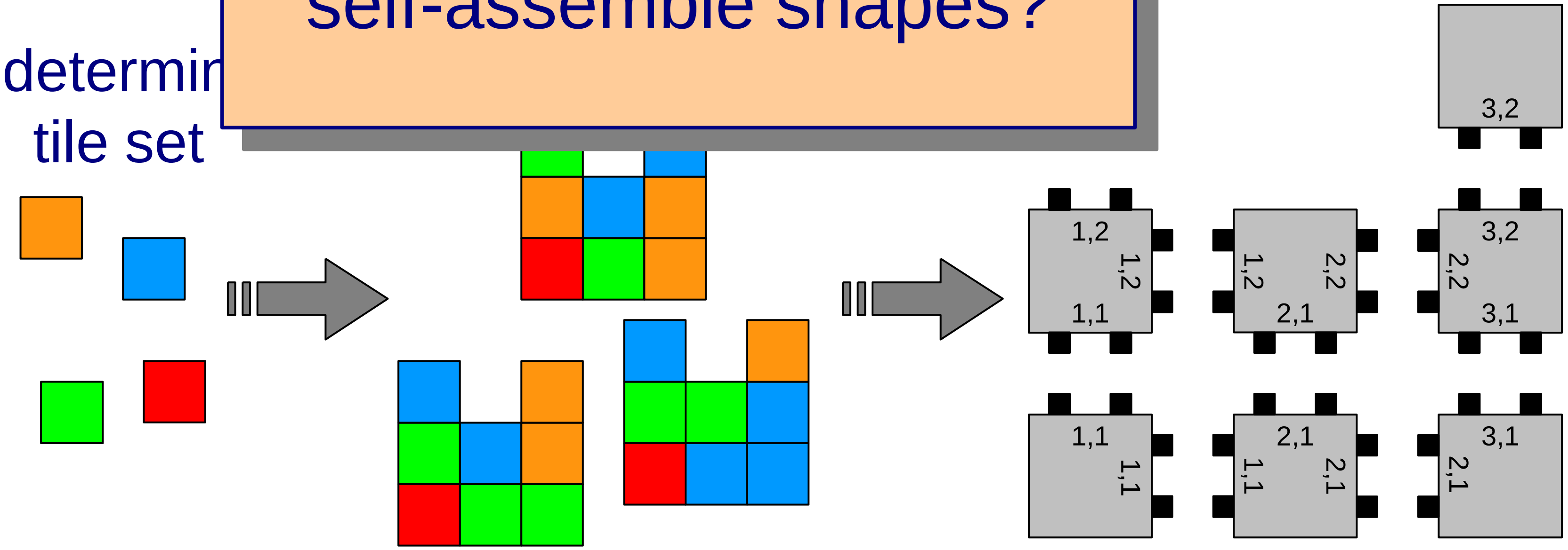
rministic tile set

d-coding *S*)

nondetermi

tile set

Is there some way that nondeterminism helps to self-assemble shapes?



Power of Nondeterminism

Question 1: Let *S* be an <u>infinite</u> shape strictly self-assembled by some nondeterministic tile system. Does some deterministic tile set also self-assemble *S*?
Is *tile computability* unaffected by nondeterminism?
Answer: **No**

There is an infinite shape *S* strictly self-assembled by <u>only</u> nondeterministic tile systems.

Remainder of talk

Question 2: Let *S* be a <u>finite</u> shape strictly self-assembled by some nondeterministic tile system <u>with *k* tile types</u>. Does some deterministic tile system <u>with at most *k* tile types</u> also self-assemble *S*?
Is *tile complexity* unaffected by nondeterminism?
Answer: **No**

There is a finite shape *S* strictly self-assembled with at most *k* tile types by <u>only</u> nondeterministic tile systems.

Optimization Problems

MinTileSet

Given: finite shape *S*

Find: size of smallest tile system that self-assembles *S*

MinDetTileSet

Given: finite shape *S*

Find: size of smallest **deterministic** tile system that self-assembles *S*

False statement: Nondeterminism does not affect tile complexity:
for every nondeterministic tile set of size *k* that self-assembles a shape *S*,
there is a deterministic tile set of size at most *k* that self-assembles *S*.

if true, would imply MinDetTileSet = MinTileSet

Main Result

- We show: MinTileSet is $\mathbf{NP}^{\mathbf{NP}}$-complete.

  a.k.a., Σ_2^P

- MinDetTileSet is **NP**-complete. (Adleman, Cheng, Goel, Huang, Kempe, Moisset de Espanés, Rothemund, *STOC* 2002)

- $\mathbf{NP} \neq \mathbf{NP}^{\mathbf{NP}} \Rightarrow$ MinTileSet $\neq$ MinDetTileSet

Nondeterminism in Algorithms and Self-Assembly

Algorithm that flips coins but always produces same output

- coin flips **useless**

Tile set that flips coins but always produces same shape

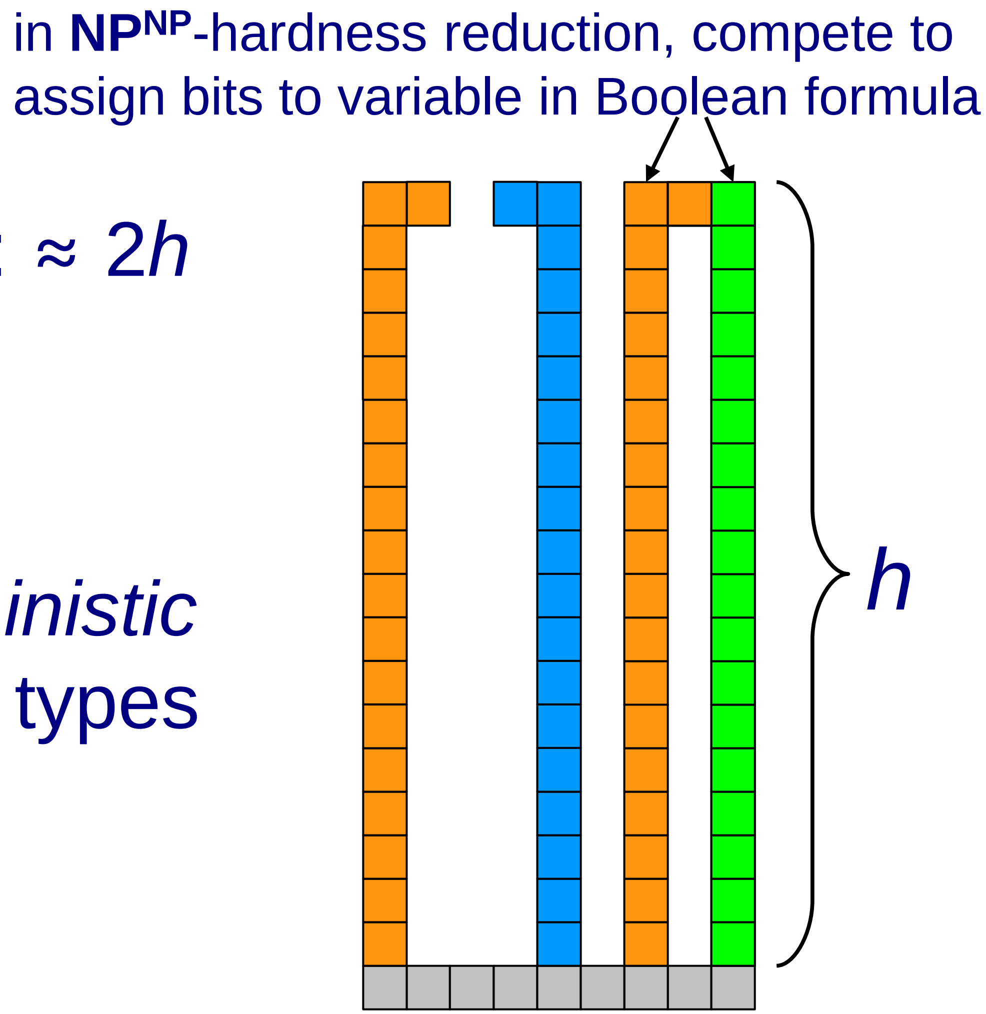
- coin flips **useful**

But … finding smallest tile set is harder if it flips coins.

A Finite Shape for which Nondeterminism Affects Tile Complexity

- Smallest tile set: ≈ 2*h* tile types

- Smallest *deterministic* tile set: ≈ 3*h* tile types



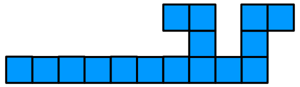
NP^{NP}-hardness Reduction

- **NP^{NP}**-complete problem (Stockmeyer,Wrathall 1976): ∃∀CNF-UNSAT
  - Given: CNF Boolean formula Φ with $k+n$ input bits $x=x_1 \ldots x_k$ and $y=y_1 \ldots y_n$
  - Question: is $(\exists x)(\forall y)\neg\Phi(x,y)$ true?
- **Reduction goal:** Given Φ, output shape S and integer c such that $(\exists x)(\forall y)\neg\Phi(x,y)$ holds if and only if some tile set of size at most c self-assembles S.

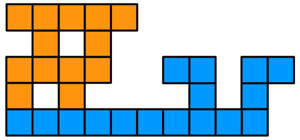
$\mathbf{NP}^{\mathbf{NP}}$-hardness Reduction

<u>Main idea</u> (due to Adleman et al. *STOC* 2002):

- Given a tree shape (no simple cycles), it is possible to compute its minimum tile set in polynomial time.
- Create a tree shape ϒ that "encodes" Φ.


ϒ

- Compute ϒ's minimal tile set *T*. (*c*=*T*)
- Create shape *S* ⊃ ϒ such that


S

  - If $(\exists x)(\forall y)\neg\Phi(x,y)$, tiles from *T* can be altered to assemble *S*.
  - Otherwise, tiles from *T* cannot be altered to assemble *S*.
  - "Since ϒ ⊆ *S*," every tile set that assembles *S* contains *T*, so if tiles from *T* cannot be altered to assemble *S* then additional tiles are needed; i.e., *S* requires more than $c = |T|$ tile types.

Evaluation of Formula

- Order variables $w = w_1 \dots w_n$ (both ∃ and ∀ variables) and clauses $C_1 \dots C_m$ arbitrarily.
- Fix an assignment to variables.
- For each clause C_j and variable w_i, let a_{ij} be the pair (U/S, T/F) representing whether C_j is satisfied by w_k for $k \le i$, and whether w_k is true or false.
- The matrix $A = (a_{ij})$ looks like

$w = 0011$

$\Phi = (w_1 \lor w_3) \land (w_1 \lor w_2 \lor w_4) \land (\neg w_1 \lor w_2)$

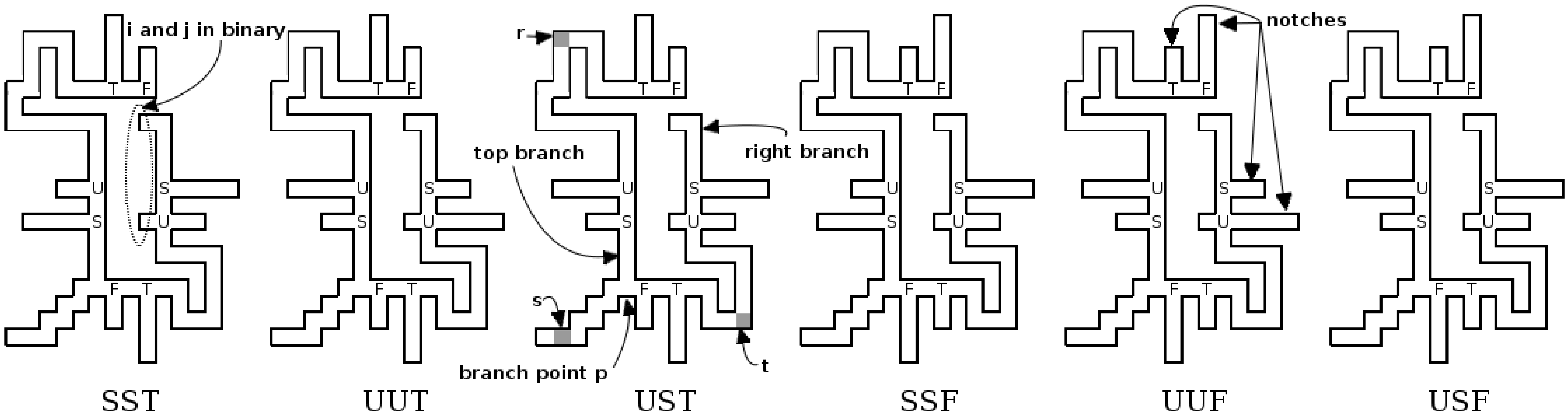
C_3	SF	SF	ST	ST
C_2	UF	UF	UT	ST
C_1	UF	UF	ST	ST
	w_1	w_2	w_3	w_4

→

highlighting when C_j goes from unsatisfied (U) to satisfied (S)

C_3	USF	SSF	SST	SST
C_2	UUF	UUF	UUT	UST
C_1	UUF	UUF	UST	SST
	w_1	w_2	w_3	w_4

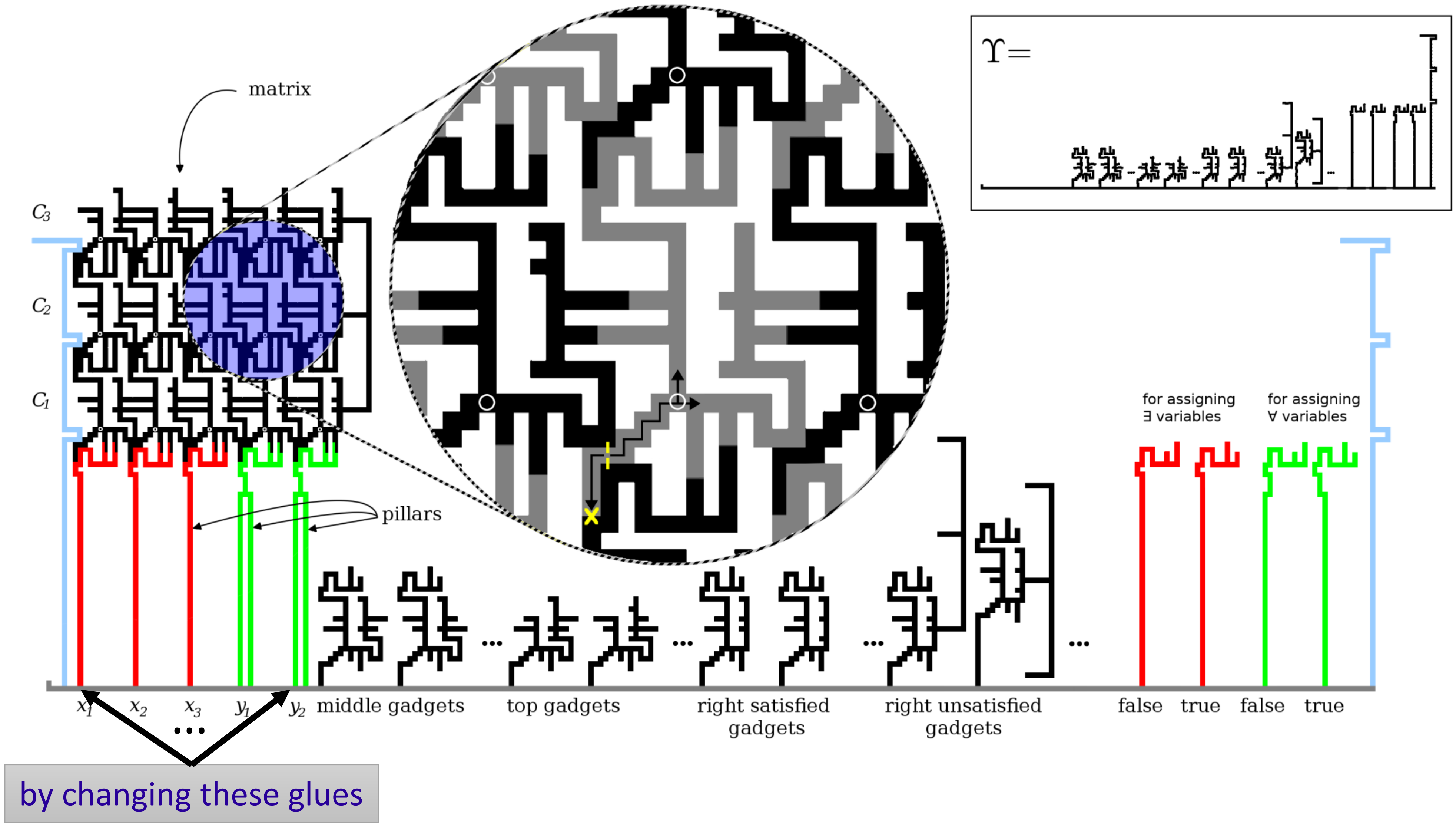
Gadgets (Adleman et al. 2002)



For each variable w_i and clause C_j, value of w_i = T/F and

SS_{ij} – C_j satisfied by a previous variable (w_k for $k < i$)
US_{ij} – C_j unsatisfied by previous variables but is satisfied by w_i
UU_{ij} – C_j unsatisfied by previous variables and by w_i

Shape S



T_Υ = tile types to self-assemble ϒ; size $c = |T_\Upsilon|$

$(\exists x)(\forall y)\neg\Phi(x,y)$ is true ⇔ tiles in T_Υ can be modified to self-assemble S

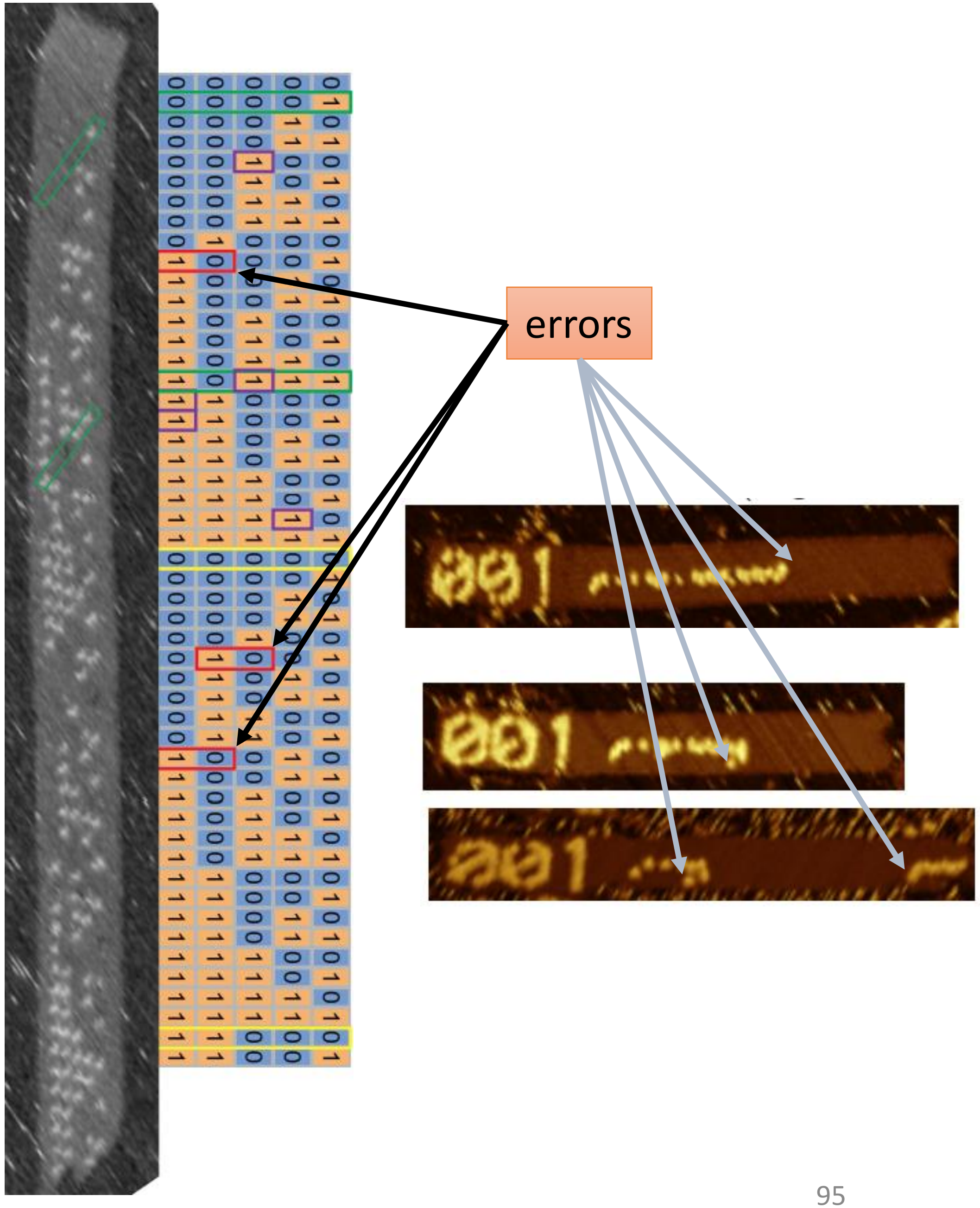
Open Questions

- How large is the gap between deterministic tile complexity and unrestricted tile complexity? our example has ratio 3/2; Schweller (unpublished) improved to quadratic gap: https://faculty.utrgv.edu/robert.schweller/papers/TheGap.pdf
- Hardness of approximation of minimum tile set problem
- Minimum tile set problem when shape is a square
  - deterministic case in **P**; likely not **NP**-hard by Mahaney's theorem (no sparse set is **NP**-hard unless **P**=**NP**)
- *Weak self-assembly* (pattern painting): paint some tile types "black", and say "pattern assembled" is set of points with a black tile
  - *Minimum tile set problem:* uncomputable! (**NP**-complete with some restrictions: https://arxiv.org/abs/1404.0967)
  - *Power of nondeterminism:* is it possible to uniquely paint a pattern, but only by assembling more than one shape on which the pattern is painted?

Errors in algorithmic self-assembly

Errors in self-assembly

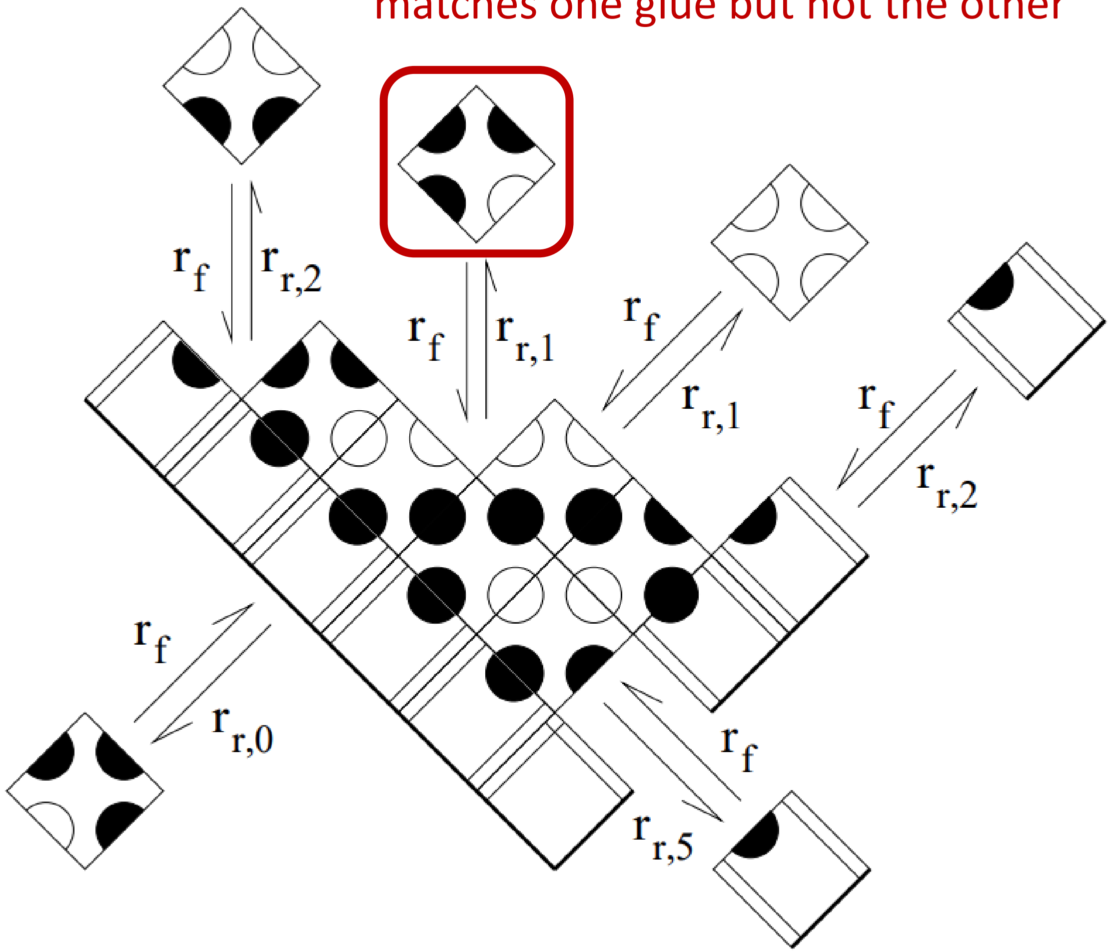
- abstract Tile Assembly Model (aTAM, the model we've used so far):
  - tiles attach but never detach
  - tiles bind only with strength 2 or higher
- unrealistic… what's a better model?
- kinetic Tile Assembly Model (kTAM); essential differences with aTAM:
  - tiles can detach
  - tiles can bind with strength 1



Modeling errors: kinetic Tile Assembly Model

- All tiles attach with rate r_f (no matter how many glues match)
- Tiles detach with rate $r_{r,b}$, if they are attached by total glue strength *b*
- "rate" = time until it occurs is exponential random variable with that rate; expected time 1/rate
  - a.k.a., continuous time Markov process
- Take home message: tiles bound with fewer glues (potential errors) fall off faster, but could get locked in by subsequent neighboring attachment

main cause of algorithmic errors: tile matches one glue but not the other



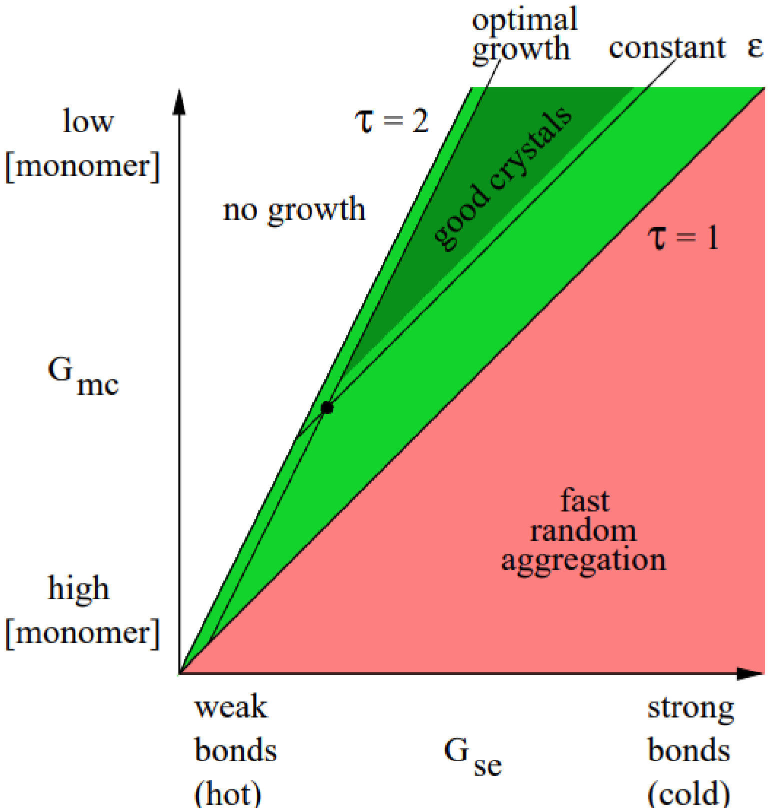
kTAM simulators

- ISU TAS (developed by Matt Patitz) also does kTAM simulation:
  - http://self-assembly.net/wiki/index.php?title=ISU_TAS
  - http://self-assembly.net/wiki/index.php?title=ISU_TAS_Tutorials
- xgrow (developed by Erik Winfree)
  - https://www.dna.caltech.edu/Xgrow/
  - older and a bit less intuitive

Tradeoff between assembly speed and errors

- attach rate r_f can be controlled through concentrations
  - "energy" of attachment is called G_{mc} (<u>m</u>onomer <u>c</u>oncentration): $r_f \propto e^{-G_{mc}}$
- detach rate $r_{r,b}$ can be controlled through temperature
  - "energy" of detachment is called G_{se} (<u>s</u>ticky <u>e</u>nd): $r_{r,b} \propto e^{-b \cdot G_{se}}$
- Intuitively, setting $r_f \approx r_{r,2}$ is like "temperature $\tau = 2$" assembly
  - … but with net zero growth rate
  - make r_f a little larger, and growth is faster, but error rates go up

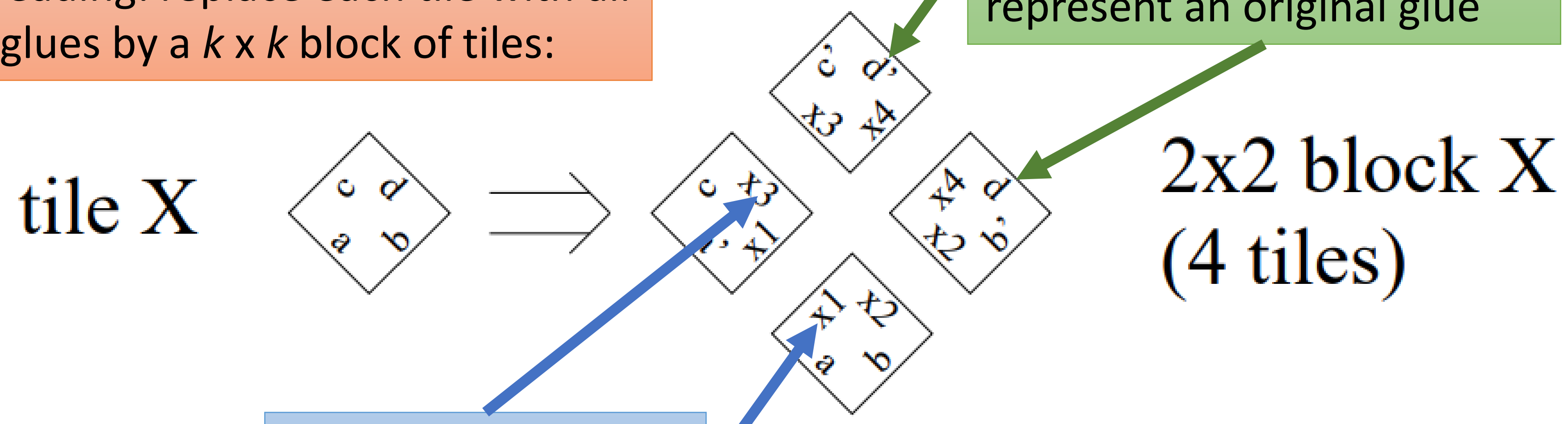
Theorem [Winfree, 1998]: To have total error rate ε, for fastest assembly speed, set $G_{se} = \ln(4/\varepsilon)$ and $G_{mc} = \ln(8/\varepsilon^2)$, i.e., $G_{mc} = 2G_{se} - \ln 2$, i.e., $r_f/r_{r,2} = 2$



Proofreading: Algorithmic error correction

k x *k* proofreading: replace each tile with all strength-1 glues by a *k* x *k* block of tiles:

glues external to the block come in *k* versions that each represent an original glue



glues internal to the block all unique

Proposition: No tiling of the *k* x *k* region with "consistent external glues" (all represent the same glue in original tile set) has *m* mismatches, where $0 < m < k$, i.e., if any mismatch occurs, then at least *k* mismatches occur before the *k* x *k* block can be completed to represent the wrong external glue.

Theorem(ish): If the error rate of the original tile system is ε, the error rate of the *k* x *k* proofreading tile system is $O(\varepsilon^k)$, e.g., if $\varepsilon = 0.01$, then 2 x 2 proofreading gets error rate about $\varepsilon^2 = 0.0001$.

Experimental algorithmic self-assembly

Crystals that think
about how they’re growing

joint work with Damien Woods, Erik Winfree, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin

slides for ECS 232: Theory of Molecular Computation

Caltech
Inria Paris
UC Davis
Harvard

Acknowledgements

Caltech

Inria Paris

UC Davis

Harvard

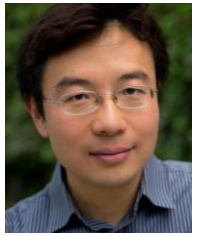
Damien Woods
(co-first author)

Erik Winfree

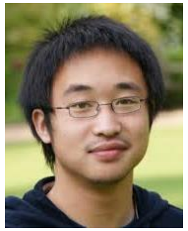
co-authors

Cameron Myhrvold

Peng Yin



Felix Zhou



Joy Hui

lab/science help

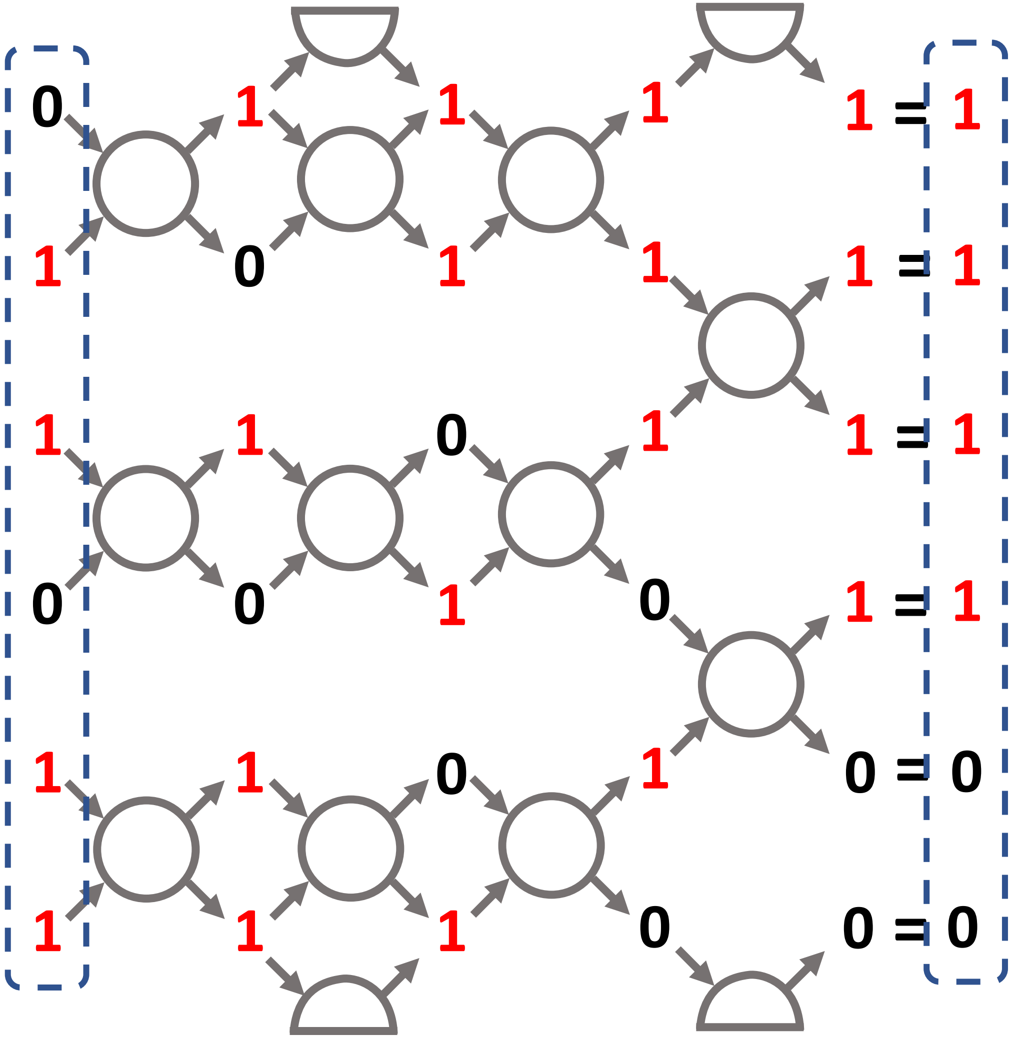
Sungwook Woo
Constantine Evans
Sarina Mohanty
Niranjan Srinivas
Deborah Fygenson
Yannick Rondolez
Mingjie Dai
Nikhil Gopalkrishnan
Chris Thachuk
Nadine Dabby
Jongmin Kim
Paul Rothemund
Bryan Wei
Cody Geary
Ashwin Gopinath

Diverse and robust molecular algorithms using reprogrammable DNA self-assembly.
Damien Woods†, David Doty†, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, Erik Winfree.
Nature 2019. †*These authors contributed equally*.

Hierarchy of abstractions

Bits:	**Boolean circuits compute**
Tiles:	Tile growth implements circuits
DNA:	DNA strands implement tiles

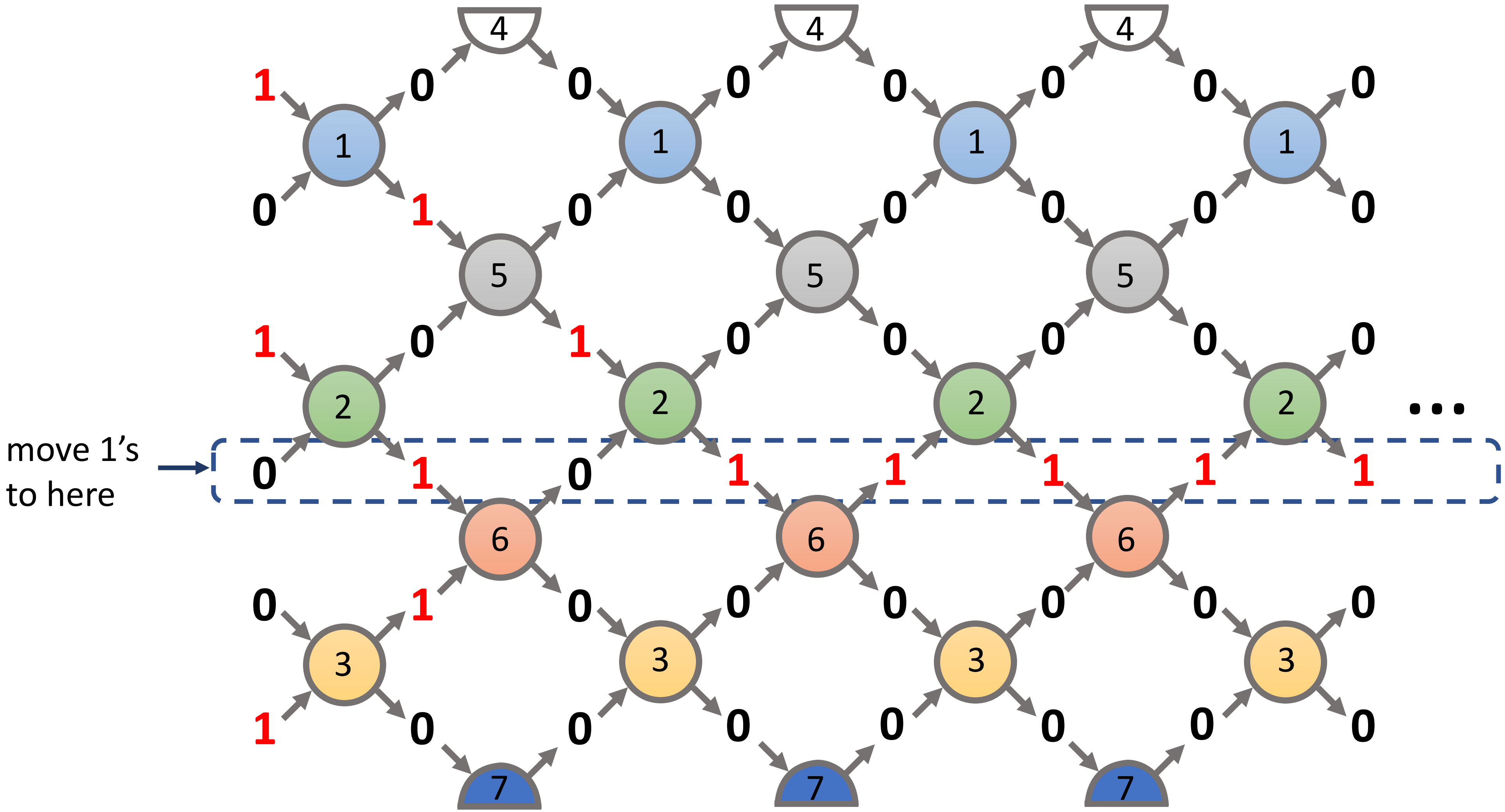
Harmonious arrangement



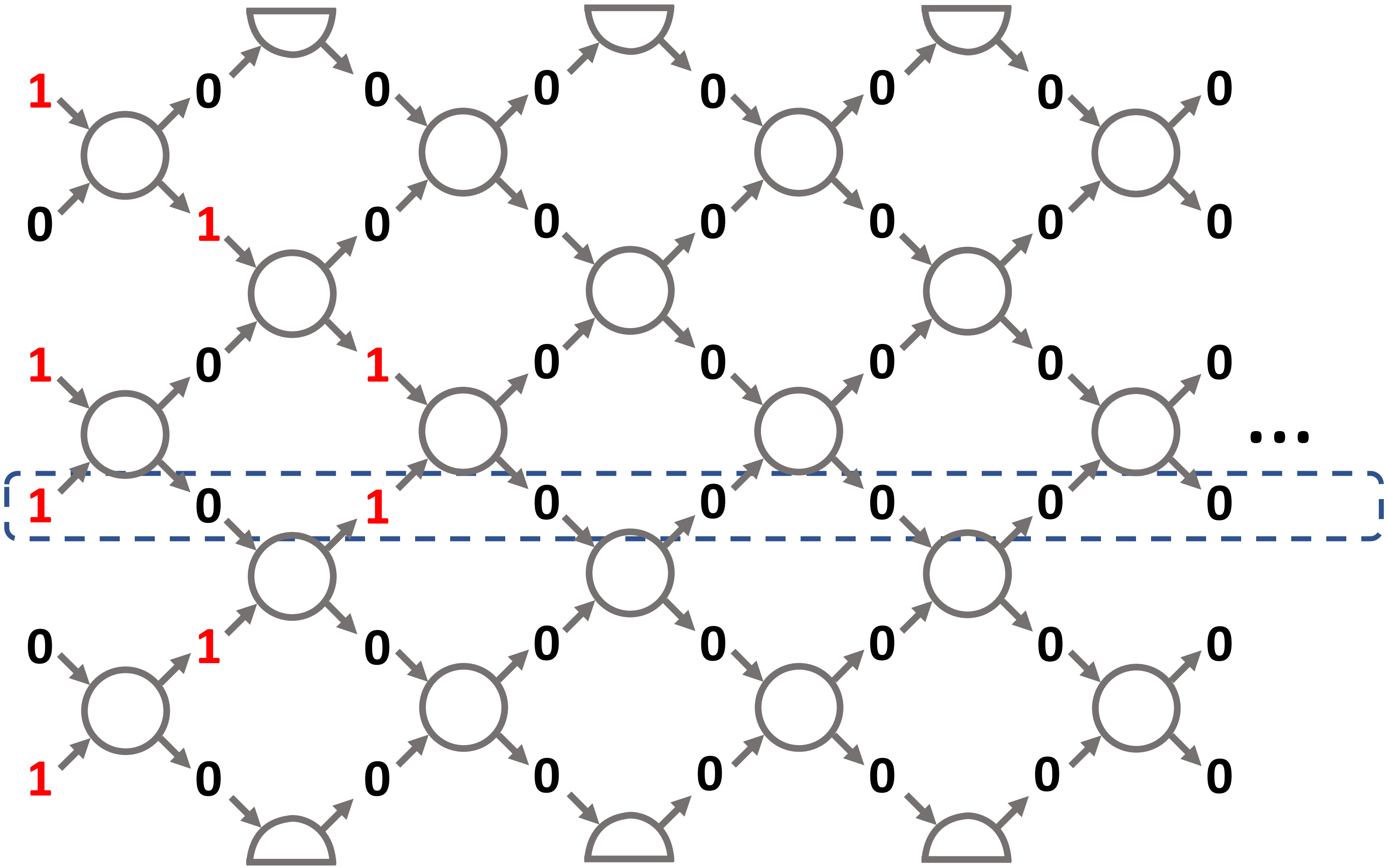
a.k.a. sorting

Odd bits

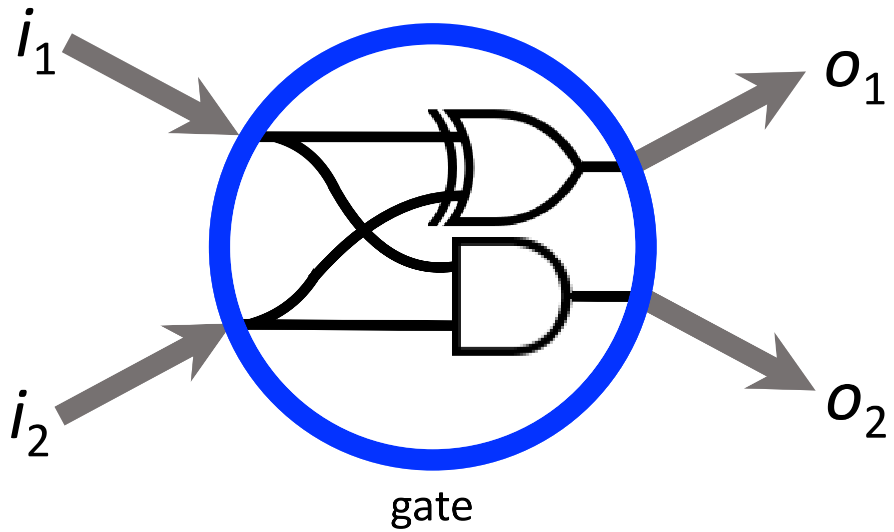
a.k.a. parity



Parity



Circuit model

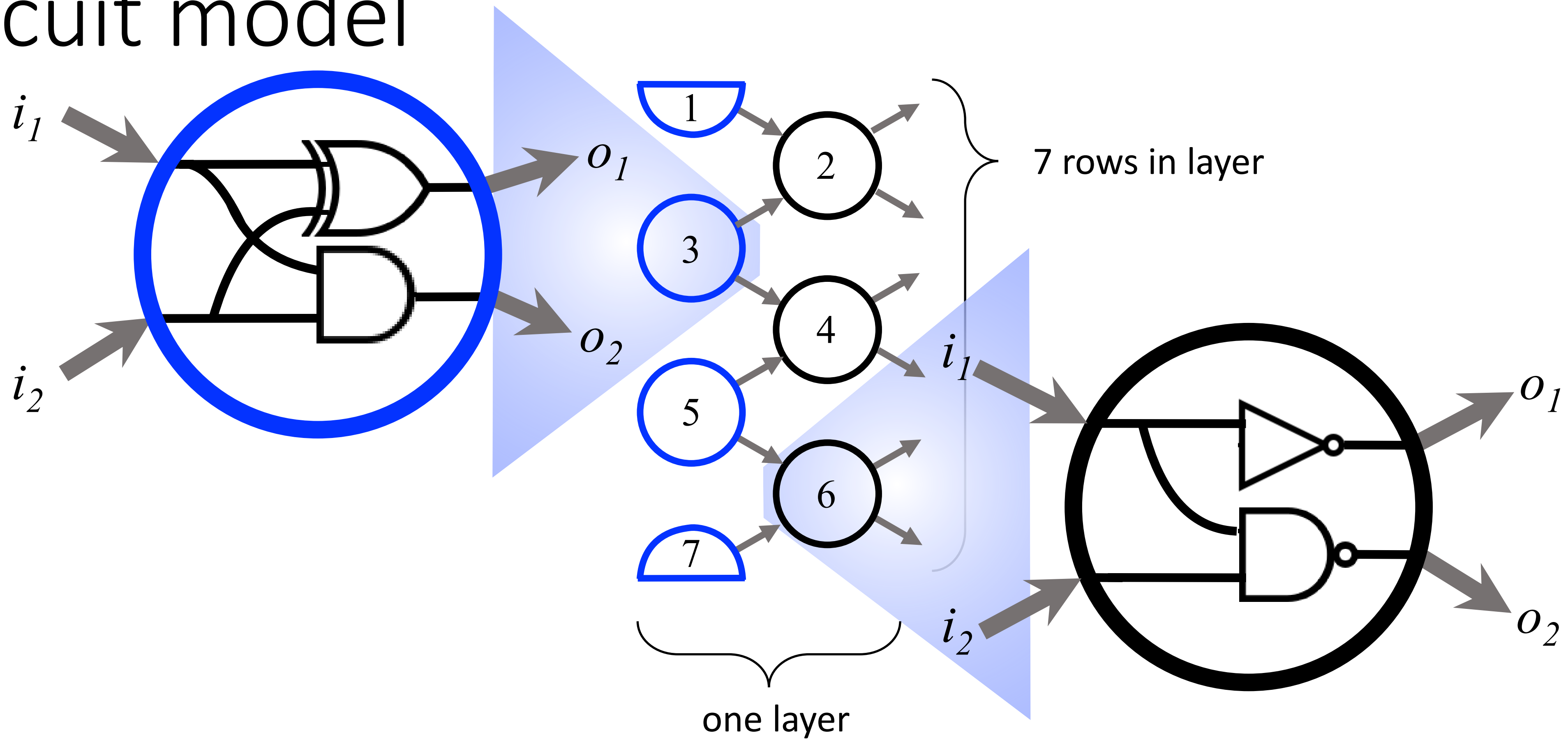


truth table

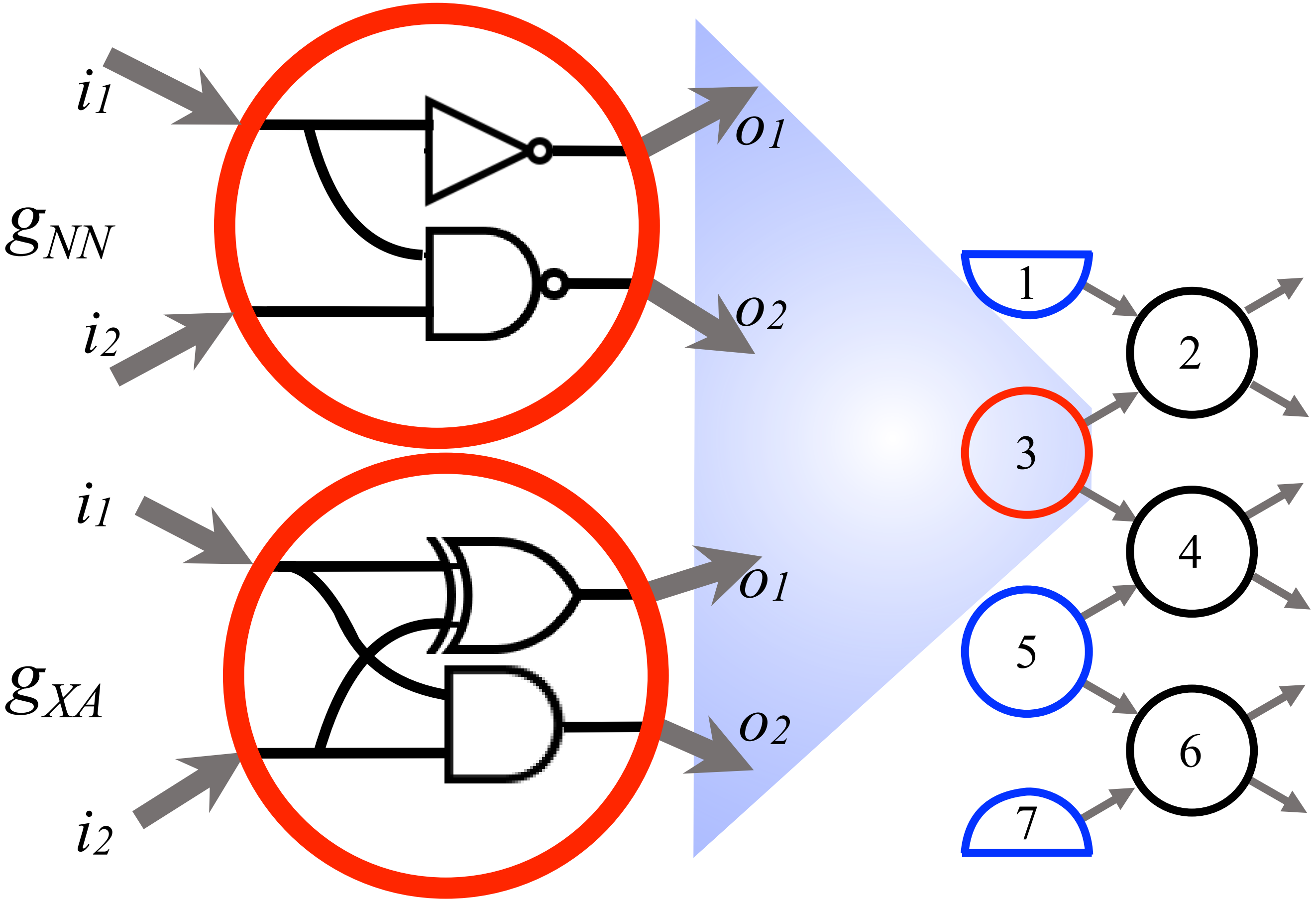
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

gate: function with two input bits i_1,i_2
and two output bits o_1,o_2

Circuit model



Circuit model

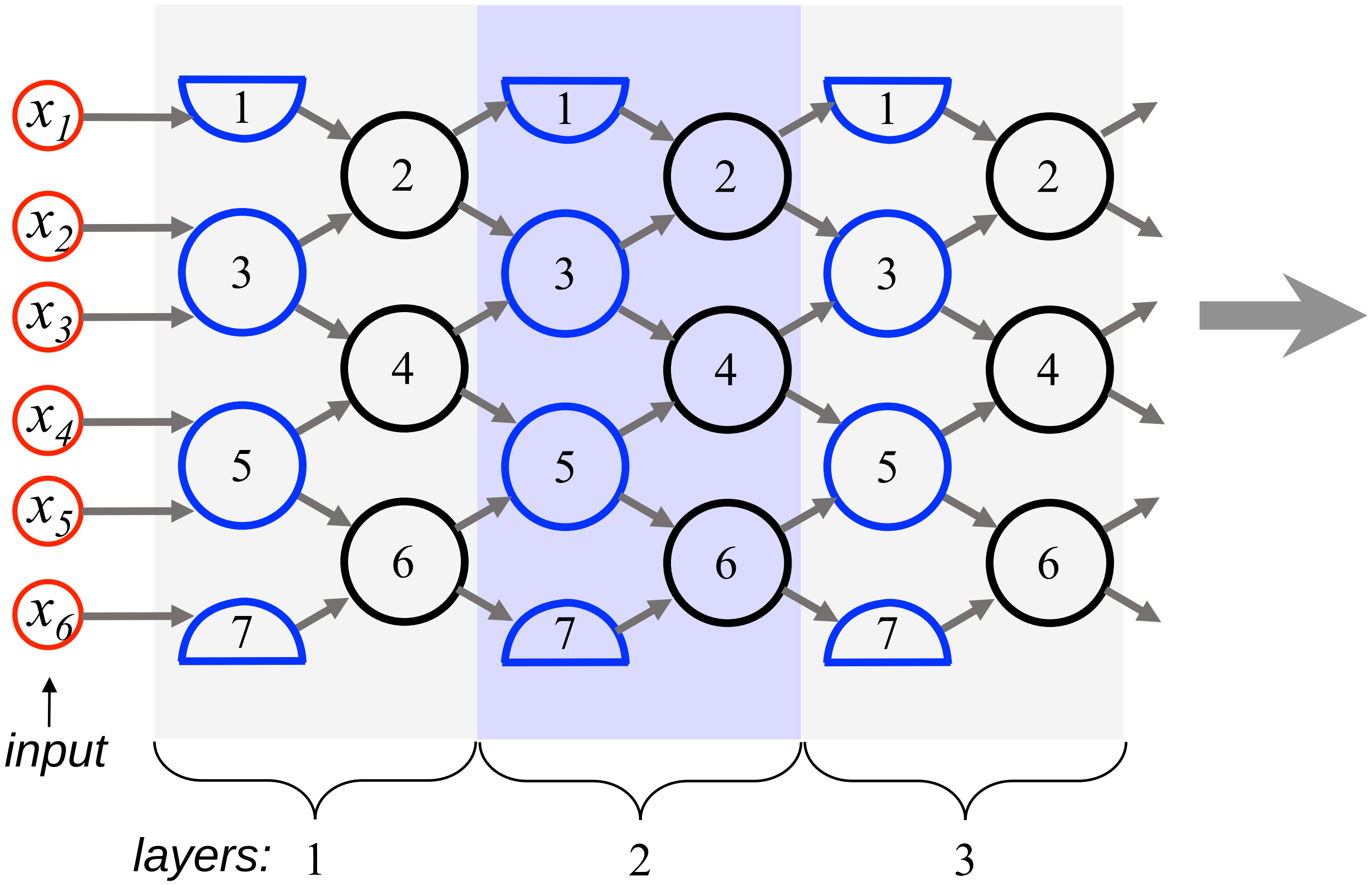


Randomization: Each row may be assigned ≥ 2 gates, with associated probabilities, e.g., $\Pr[g_{NN}] = \Pr[g_{XA}] = ½$

Circuit model

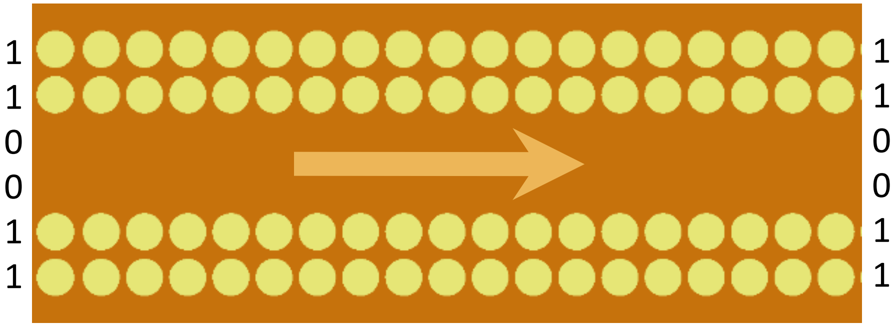
Programmer specifies layer: gates to go in each row

User gives n input bits

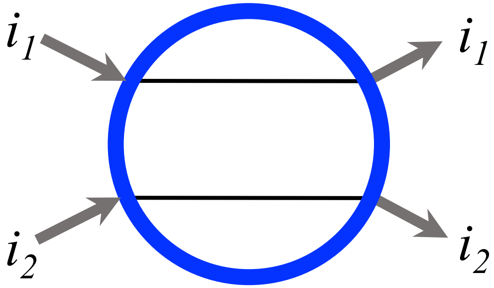


Example circuits with same gate in every row

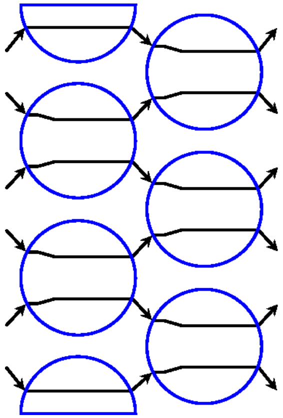
COPY



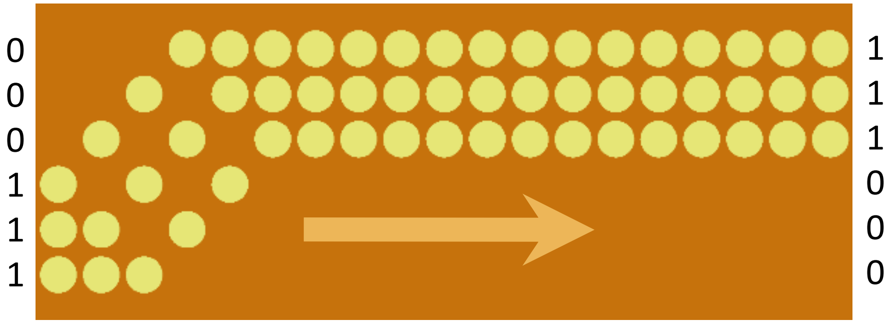
COPY gates



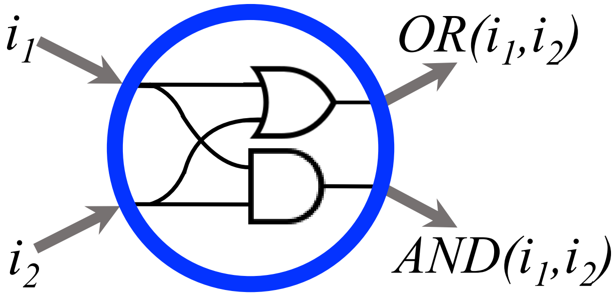
i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1



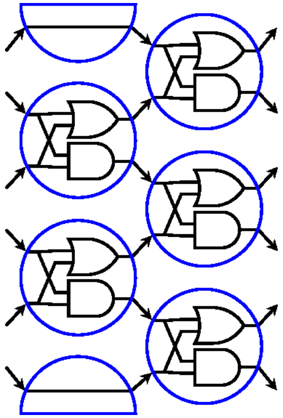
SORTING



SORTING gates

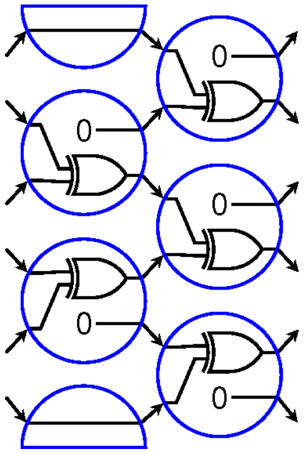


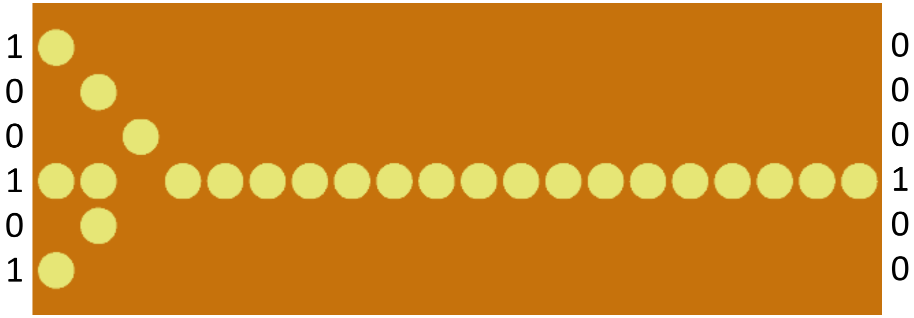
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1



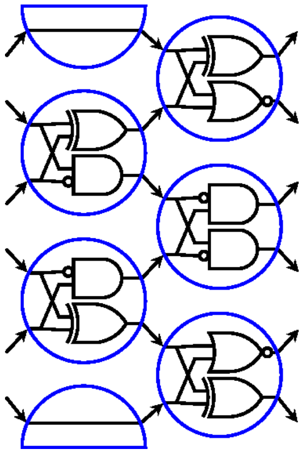
Example circuits with different gates in each row

PARITY

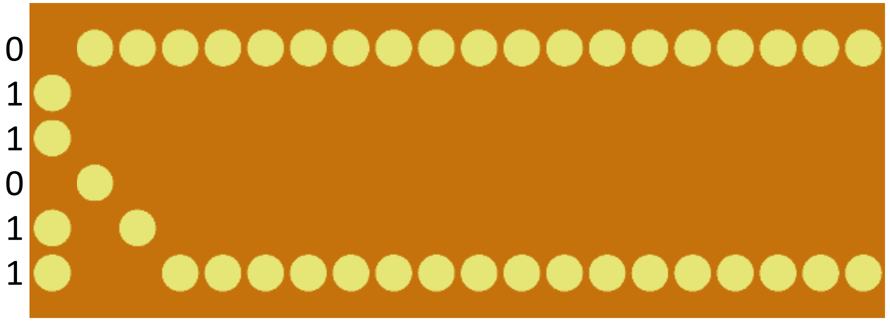




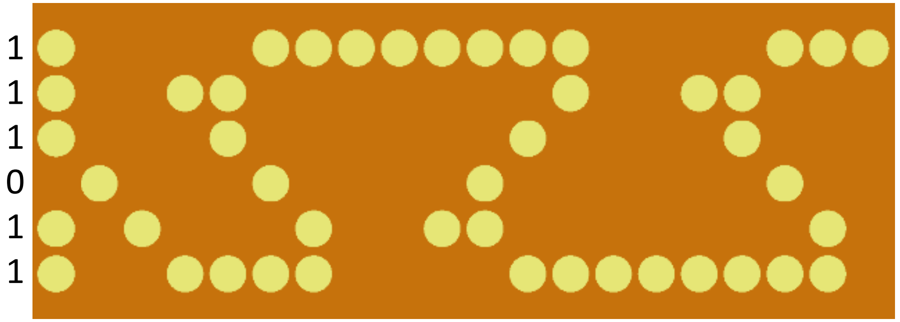
MULTIPLEOF3



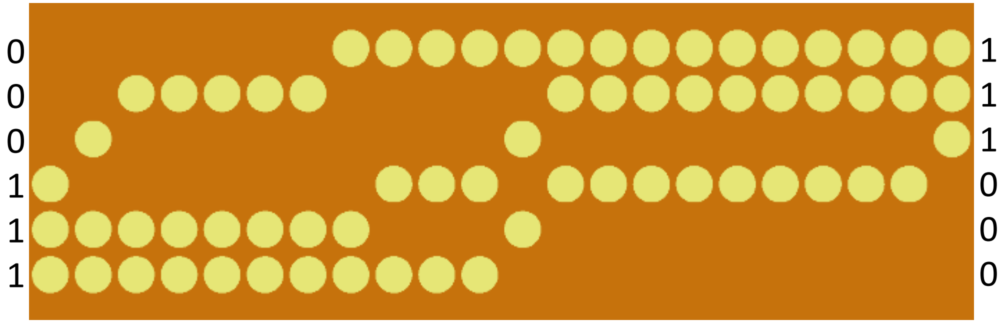
$011011_2 = 27_{10} = 3 \cdot 9$



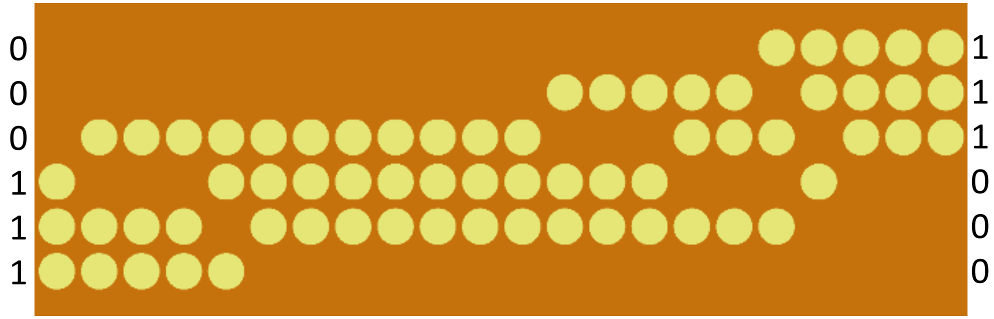
$111011_2 = 59_{10} = 3 \cdot 19 + 2$

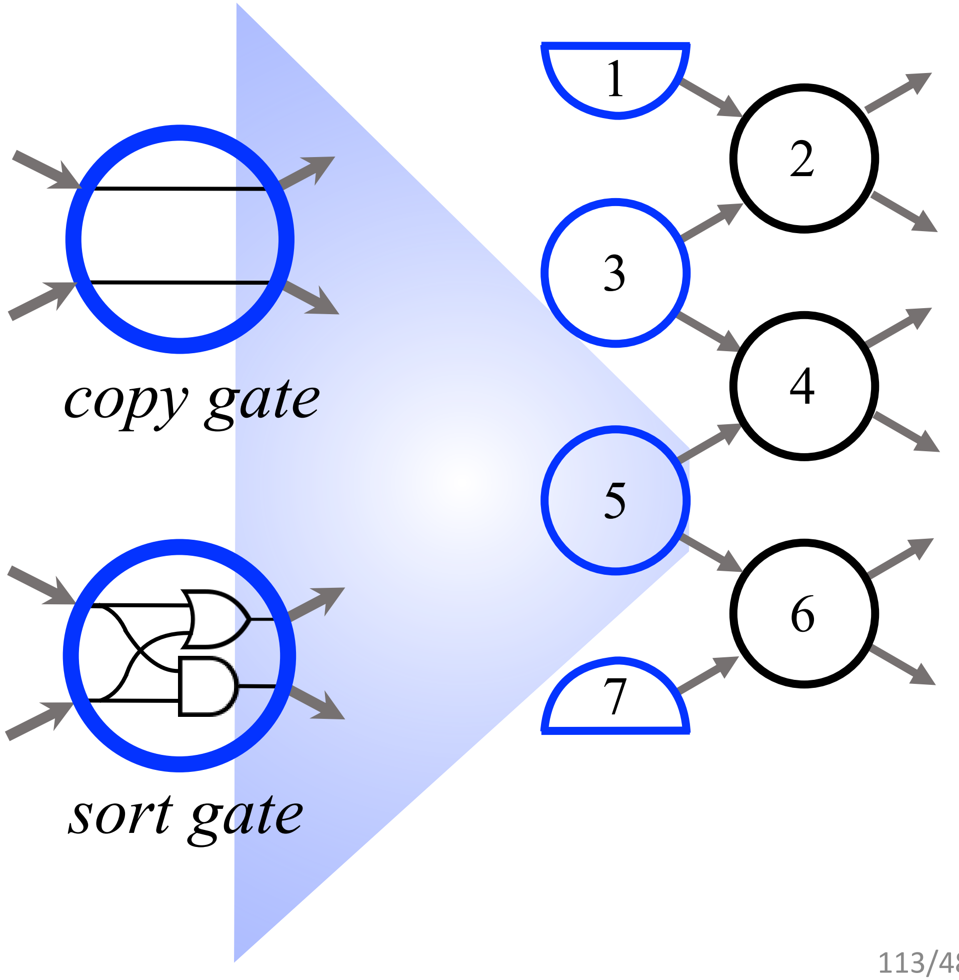


Randomization: "Lazy" sorting



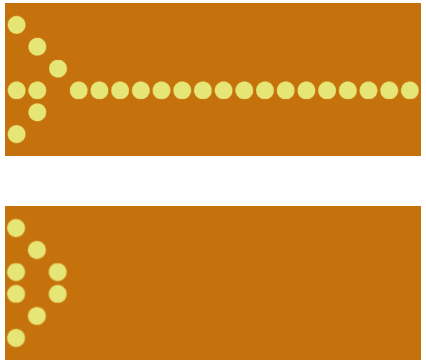
If 1 and 0 out of order, flip a coin to decide whether to swap them.



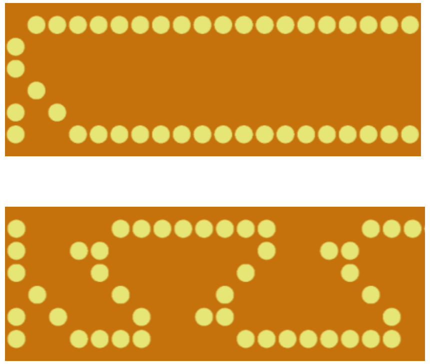


Deterministic circuits

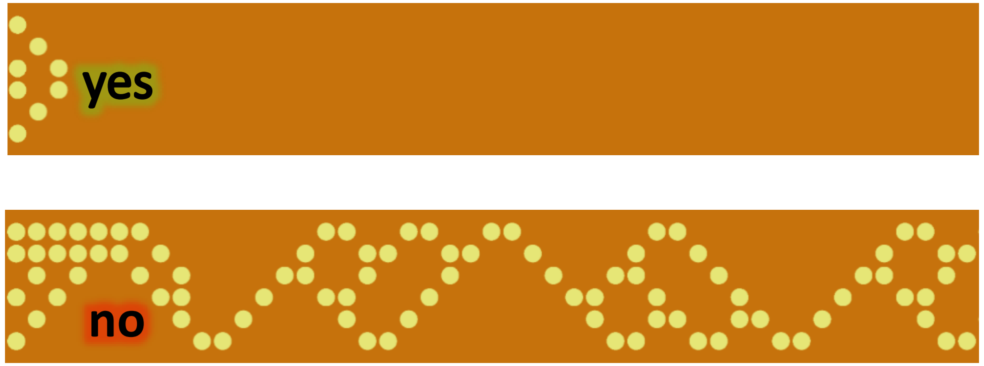
PARITY



MULTIPLEOF3



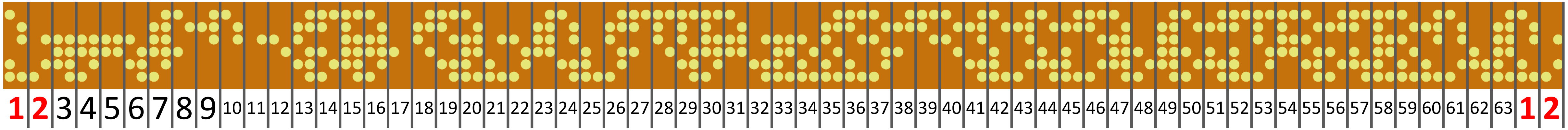
PALINDROME



answer yes/no question

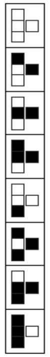
CYCLE63

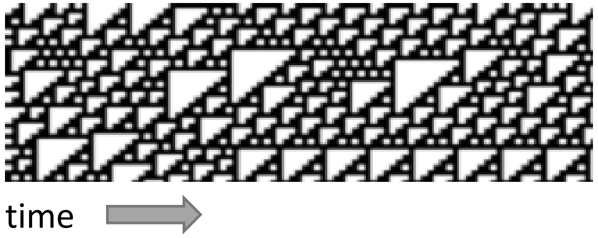
"count" as high as possible

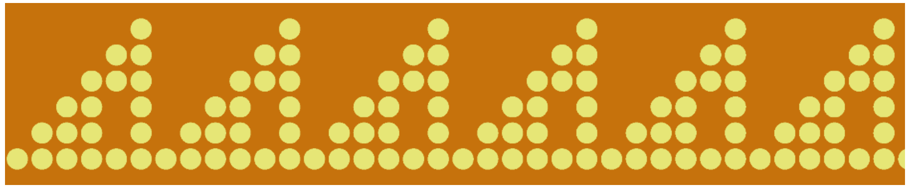


RULE110

simulate cellular automata







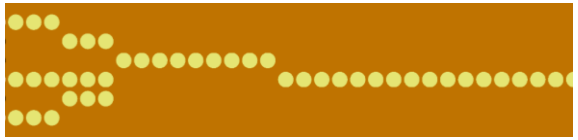
Theorem: Rule 110 can efficiently execute <u>any</u> algorithm.

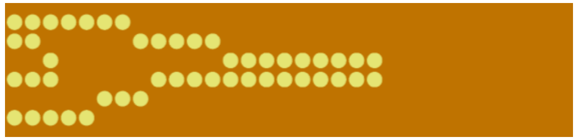
[Cook, <u>Complex Systems</u> 2004]

[Neary, Woods, <u>ICALP</u> 2006]

Randomized circuits

LAZYPARITY





RANDOMWALKINGBIT

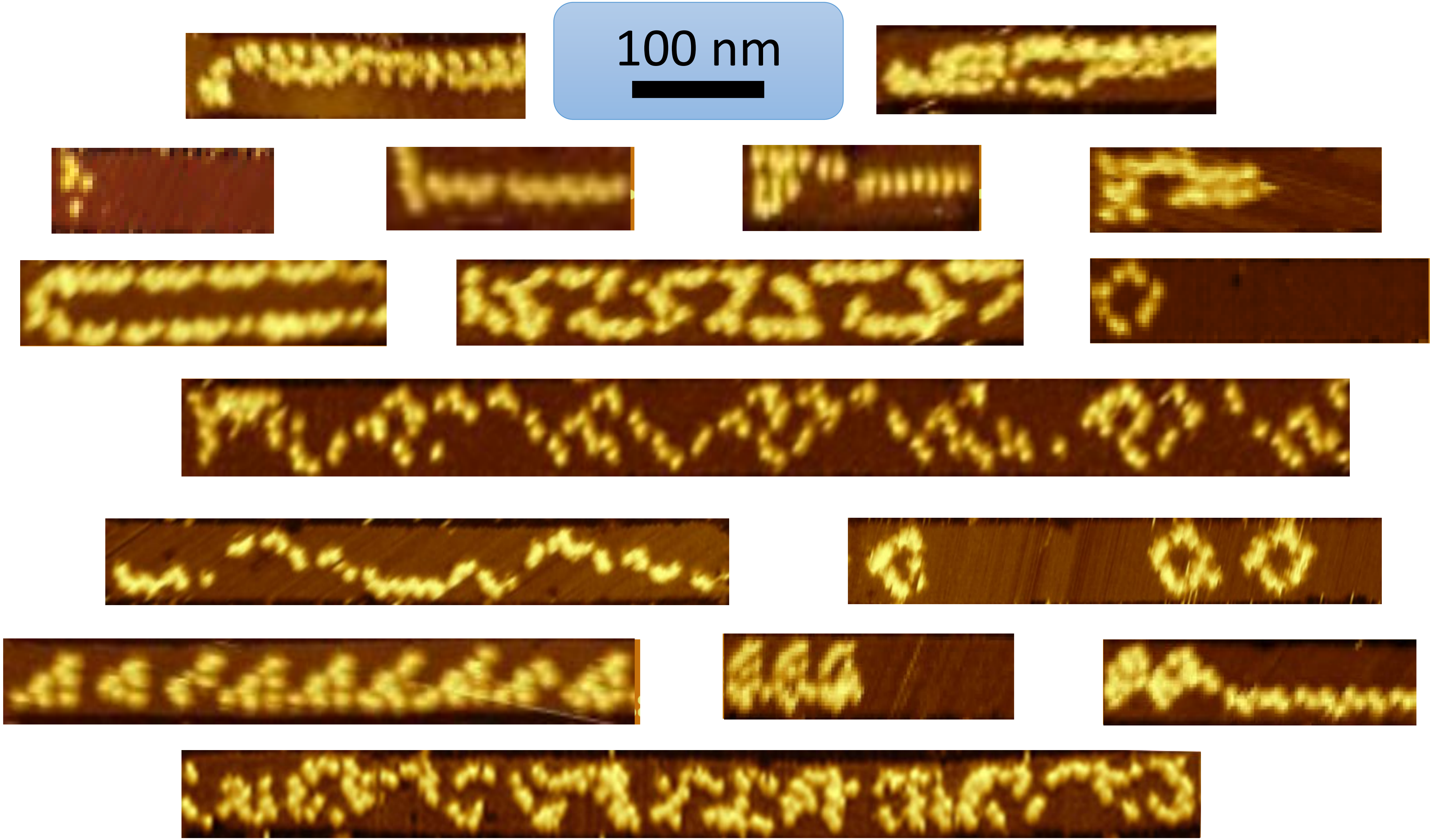
DIAMONDSAREFOREVER

FAIRCOIN

use biased coin to
simulate unbiased coin

Pr [] = Pr [] = ½

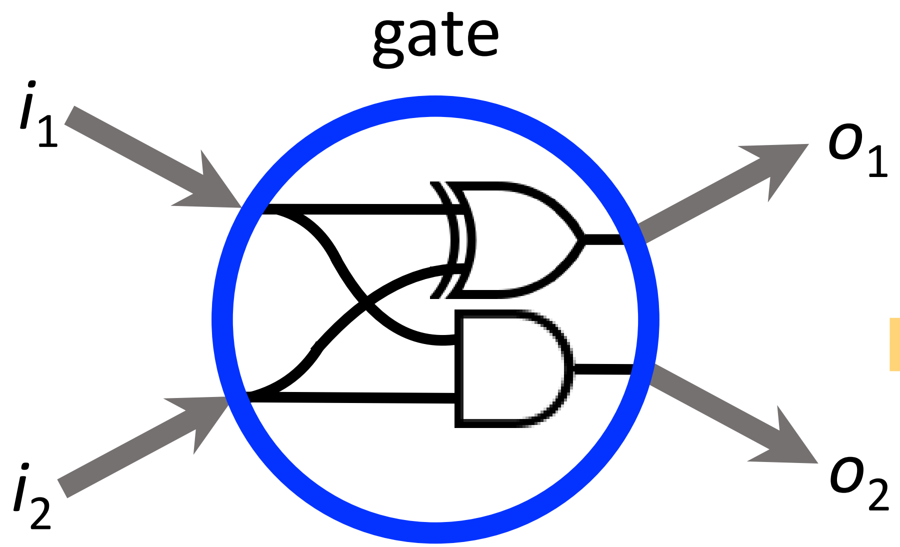
for **any** (positive) probabilities for the randomized gate


100 nm

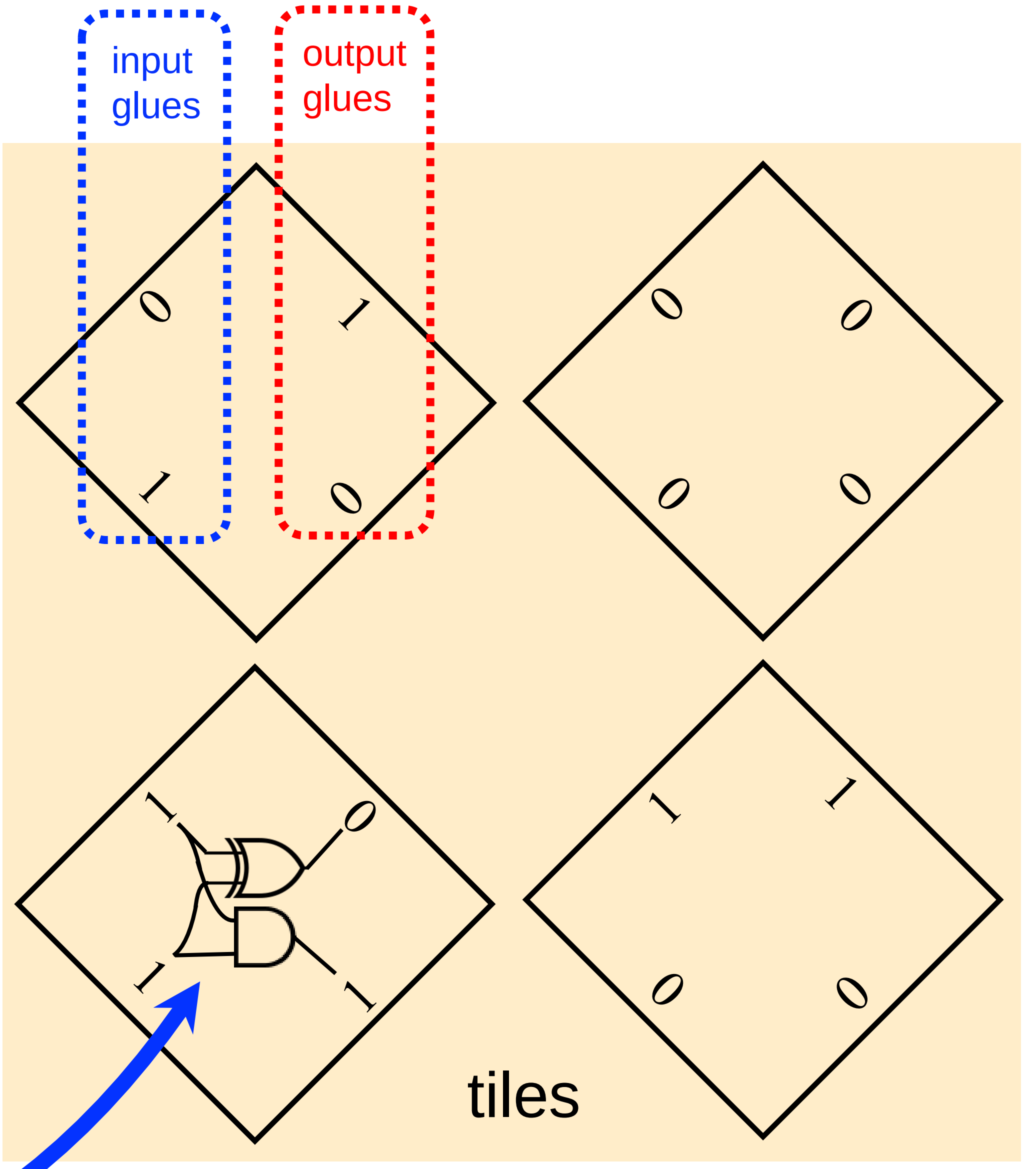
Hierarchy of abstractions

Bits:	Boolean circuits compute
➡ **Tiles:**	**Tile growth implements circuits**
DNA:	DNA strands implement tiles

Gates → Tiles

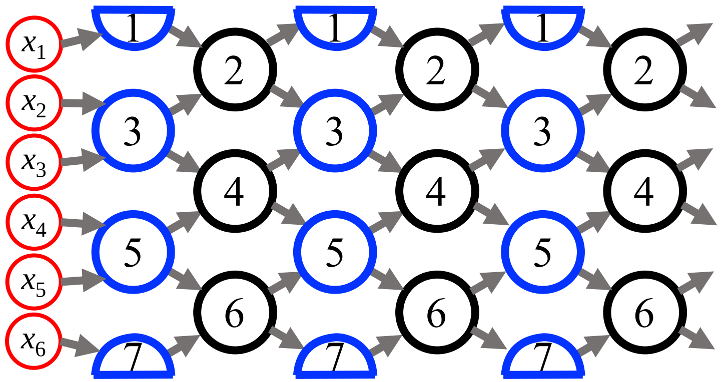


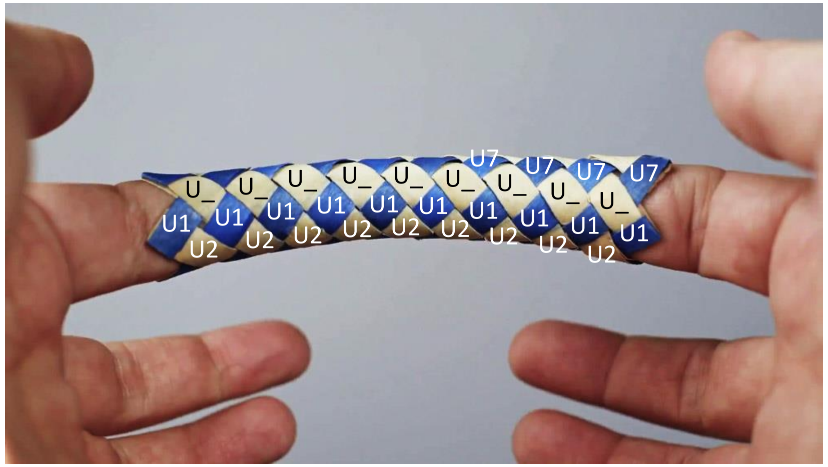
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

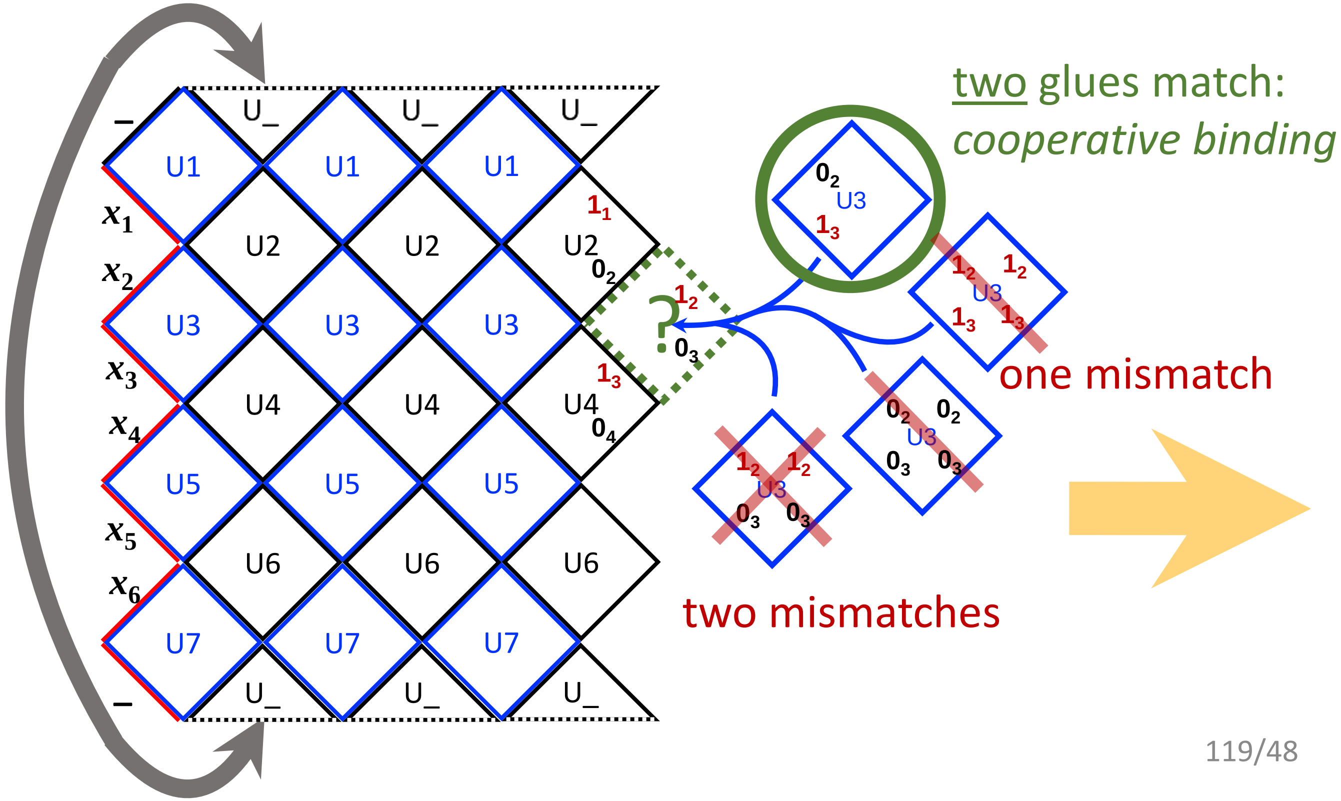


truth table row is encoded by a **tile** with 4 **glues** encoding bits

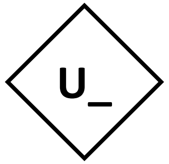
How tiles compute while growing (algorithmic self-assembly)







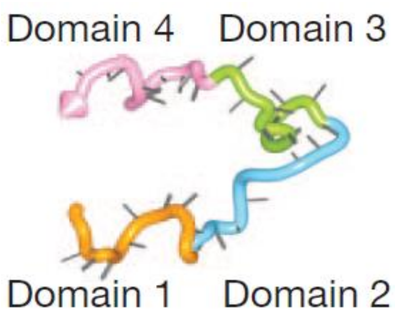
"data-free" tile wraps top to bottom to form a tube

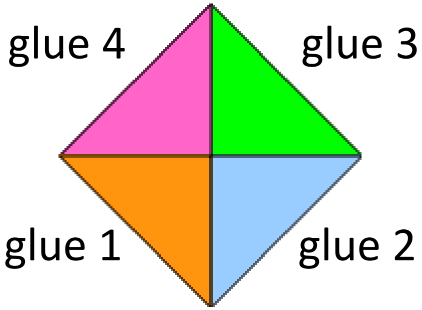


Hierarchy of abstractions

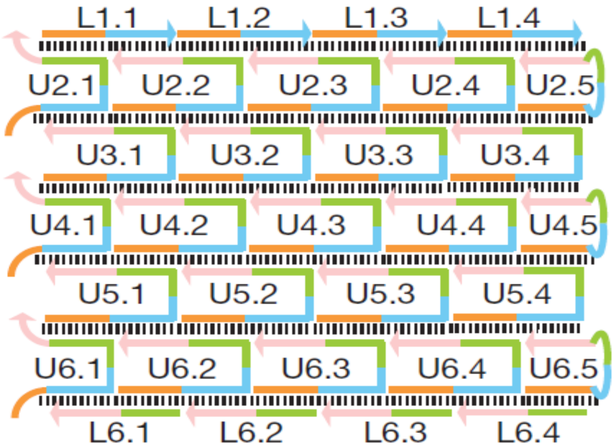
Bits:	Boolean circuits compute
Tiles:	Tile growth implements circuits
➡ **DNA:**	**DNA strands implement tiles**

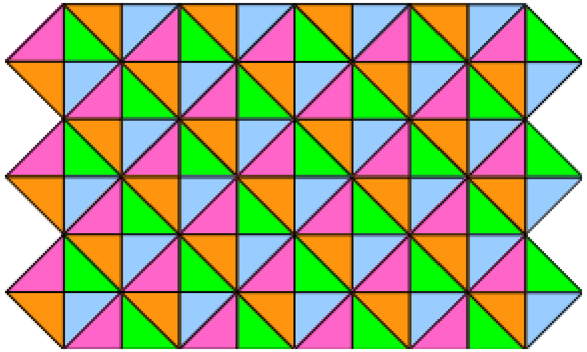
DNA single-stranded tiles





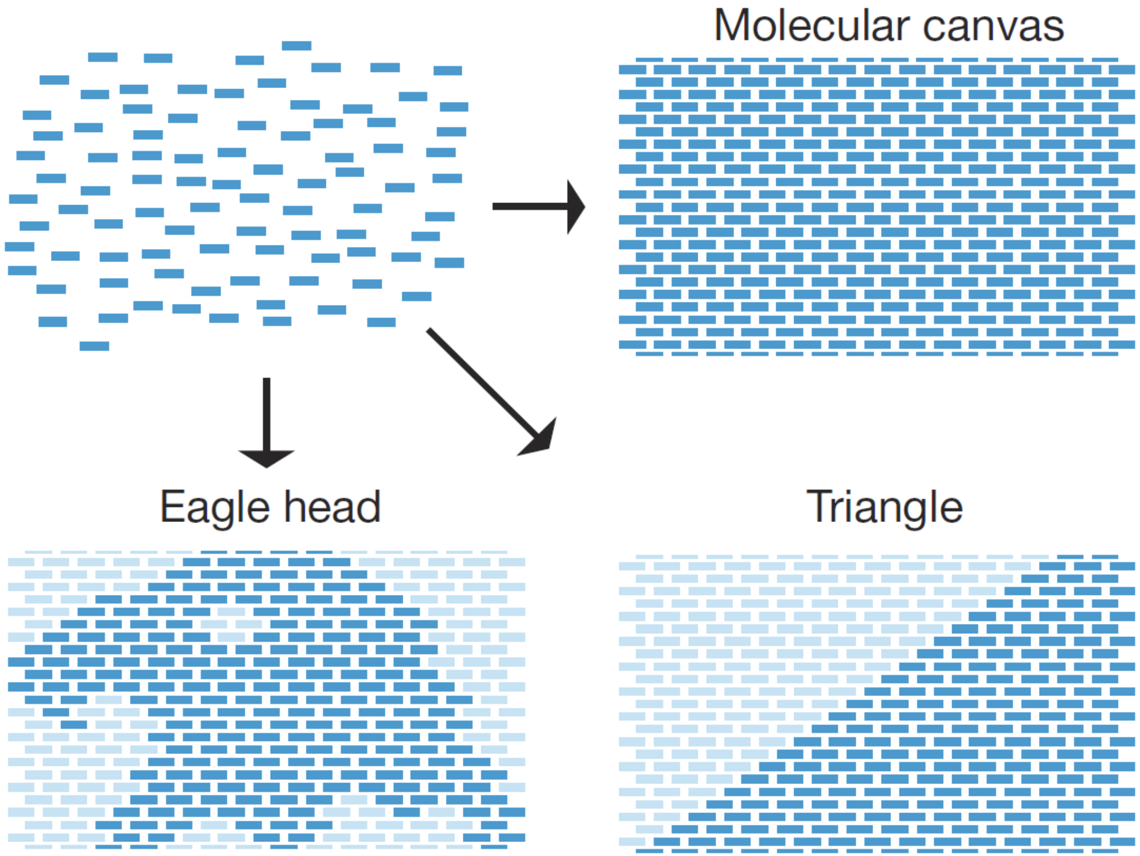
assembly





Yin, Hariadi, Sahu, Choi, Park, LaBean, and Reif.
Programming DNA tube circumferences.
<u>Science</u> 2008

Single-stranded tiles for making any shape



Bryan Wei, Mingjie Dai, and Peng Yin.
Complex shapes self-assembled from single-stranded DNA tiles.
Nature 2012.

Uniquely addressed self-assembly versus algorithmic

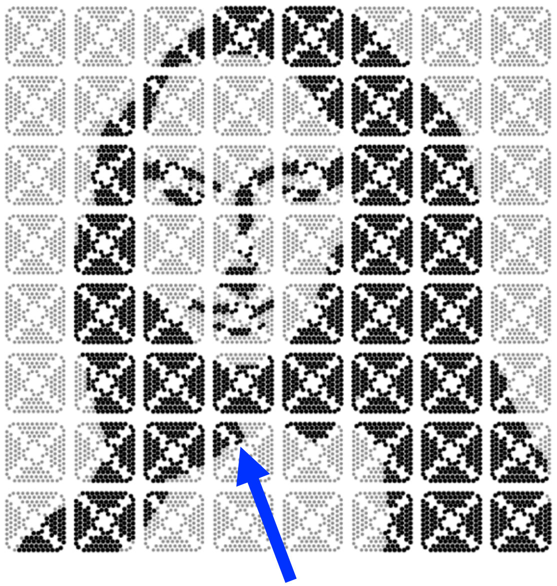
Unique addressing: each DNA "monomer" appears **exactly once** in final structure.

Algorithmic: DNA tiles are **reused** throughout the structure.

single DNA origami

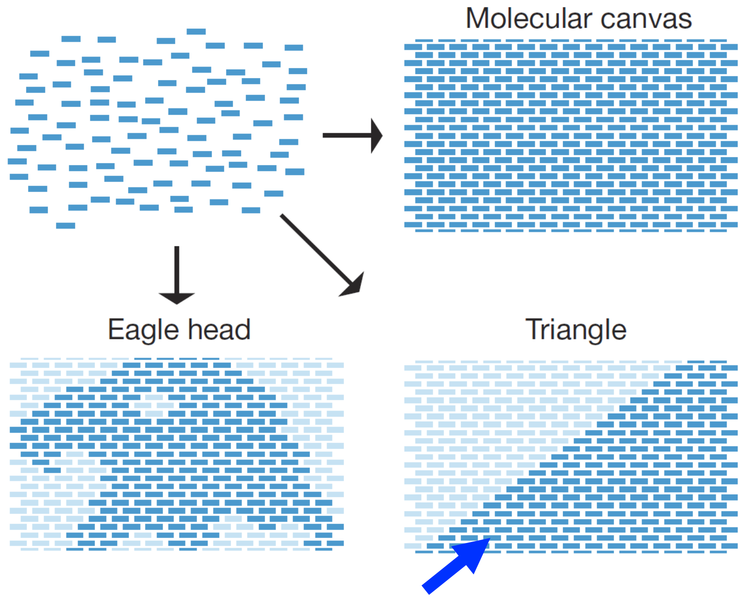
staple strand for position (4,2)

array of many DNA origamis



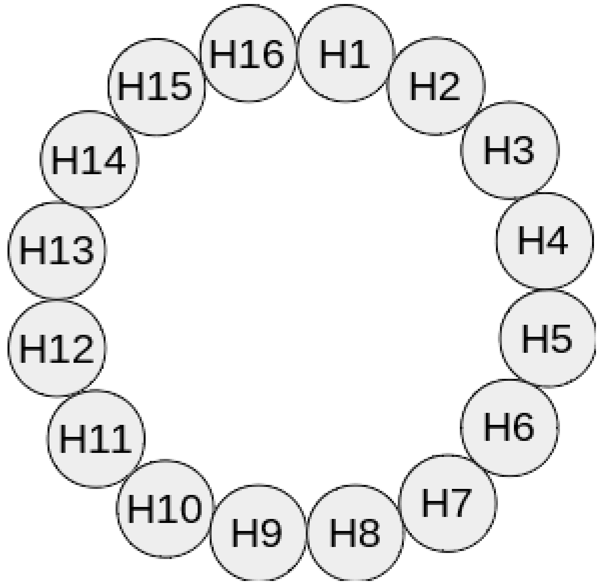
origami for position (4,2)

uniquely-addressed tiles

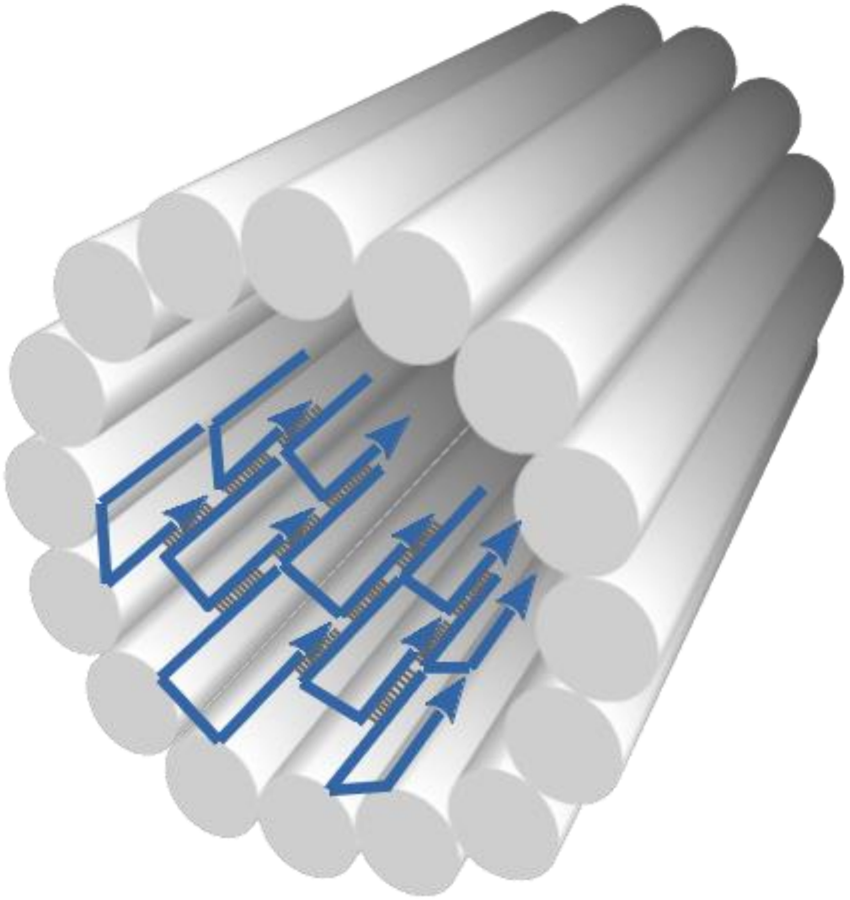


tile for position (4,2)

Single-stranded tile tubes

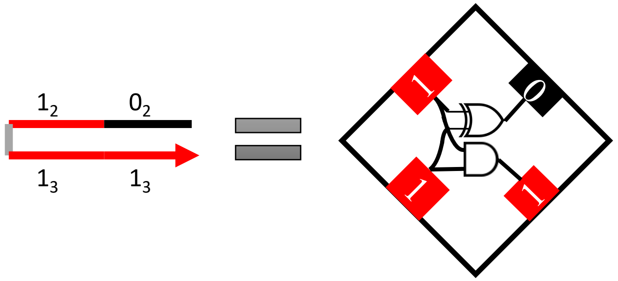


DNA-level diagram of 20-helix tube

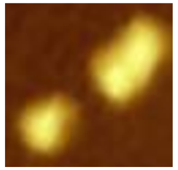


Yin, Hariadi, Sahu, Choi, Park, LaBean, and Reif. *Programming DNA tube circumferences*, Science 2008.

Seeded growth

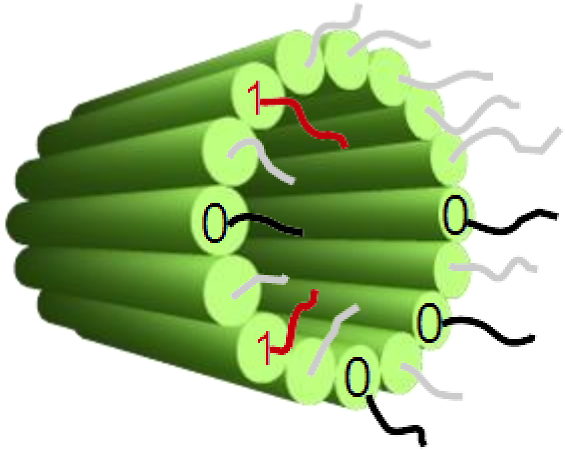


can later add streptavidin (5 nm wide protein) to bind biotins and visualize where the 1's are

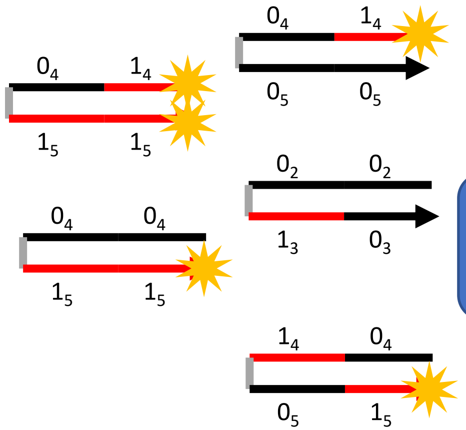


DNA origami seed

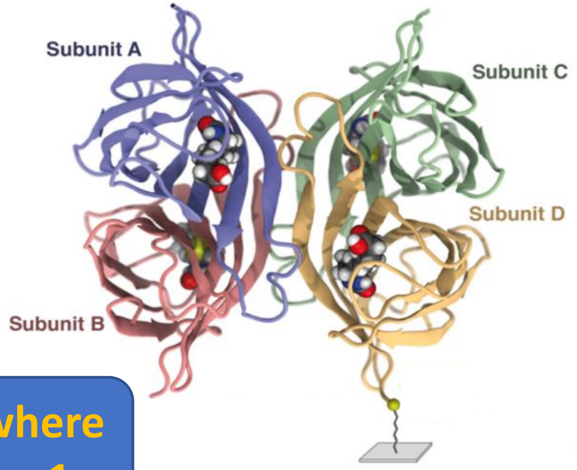
single-stranded "input-adapter" extensions encoding 6 input bits



single-stranded tiles implementing circuit gates

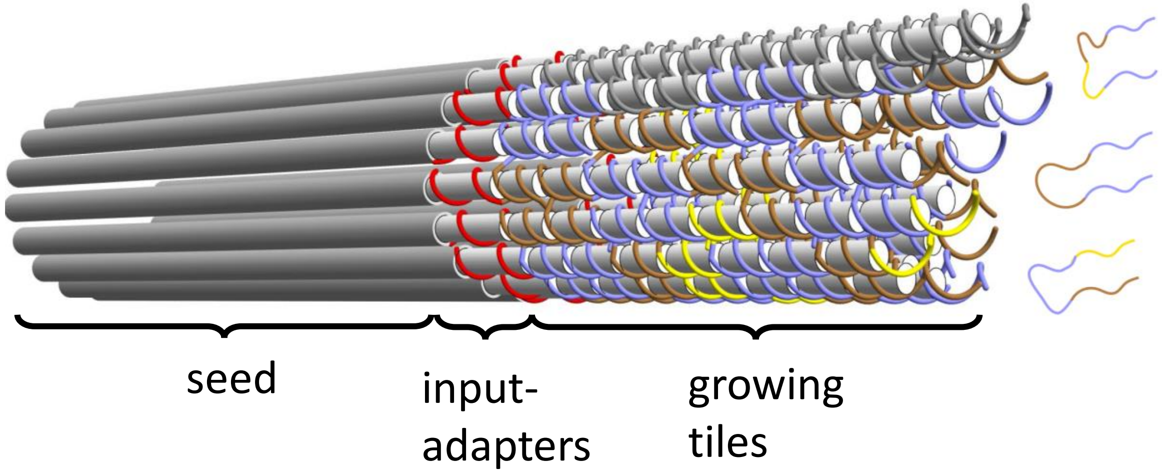


biotins where output = 1



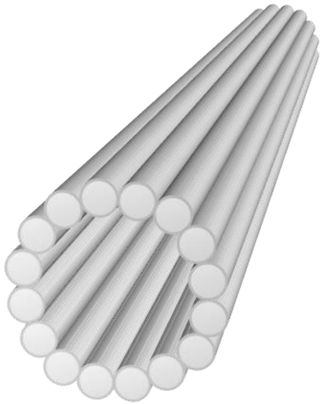
need barrier to nucleation (tile growth without seed);
[tile]=100 nM;
temperature=50.9° C

hold 8-48 hours

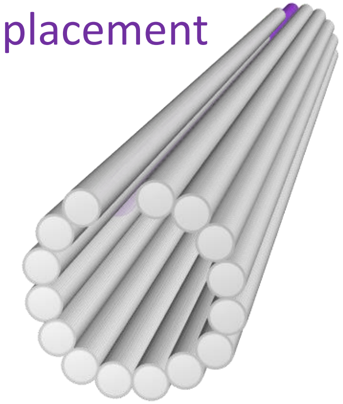


Tubes to ribbons

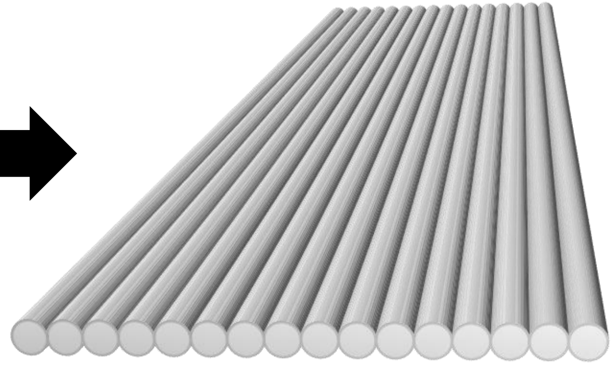
tube



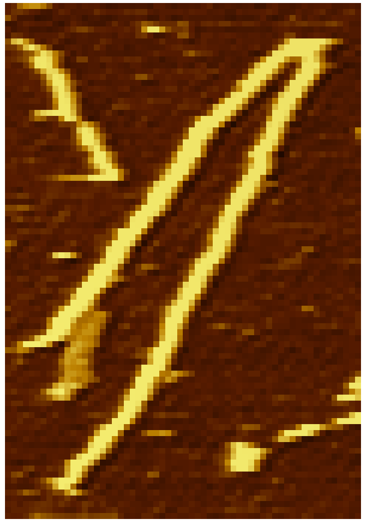
remove “seam” by
strand displacement



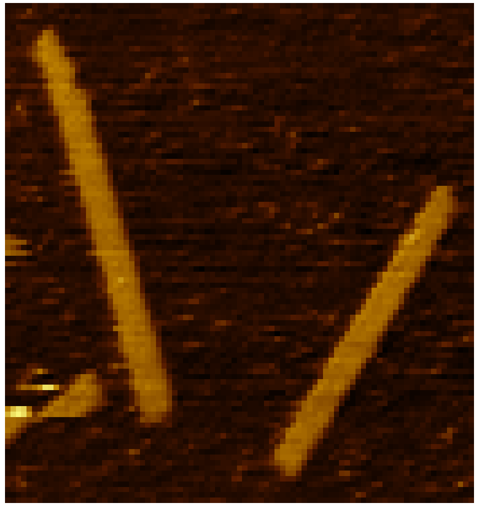
ribbon



AFM
image

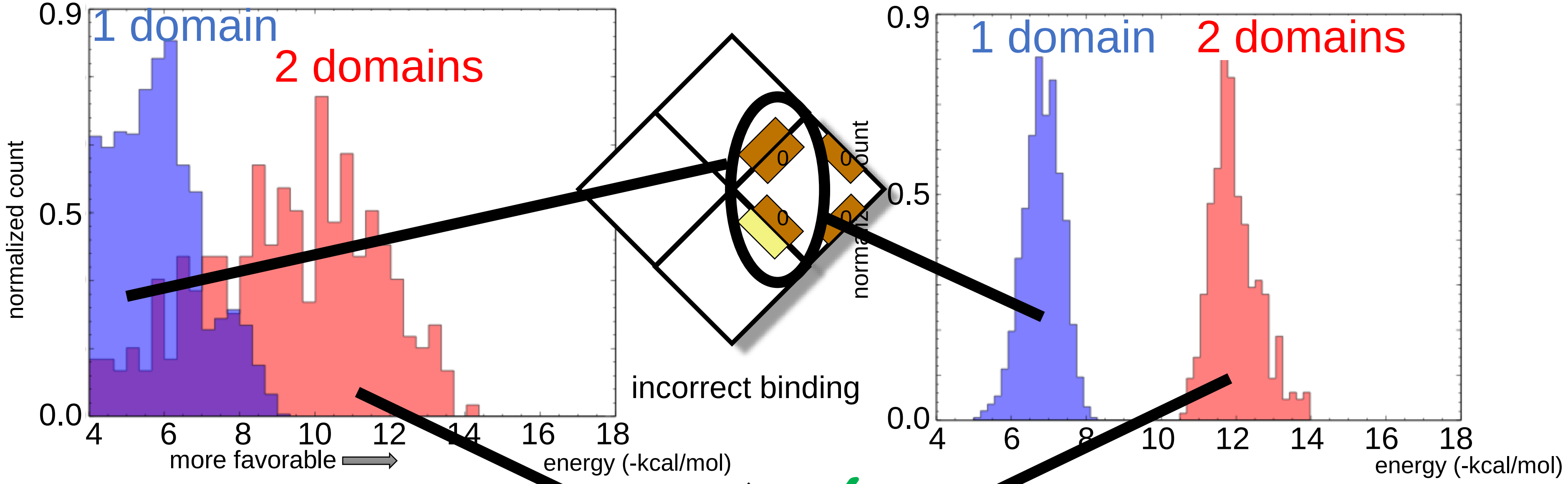


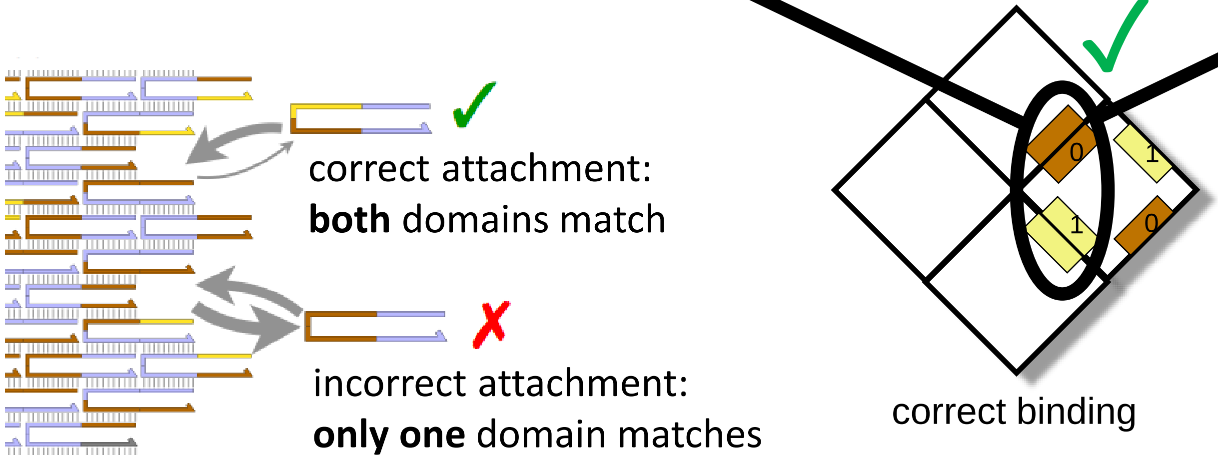
500 nm

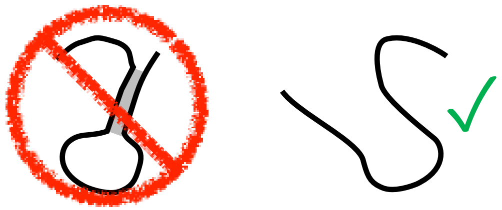


DNA sequence design

Random sequences vs designed sequences



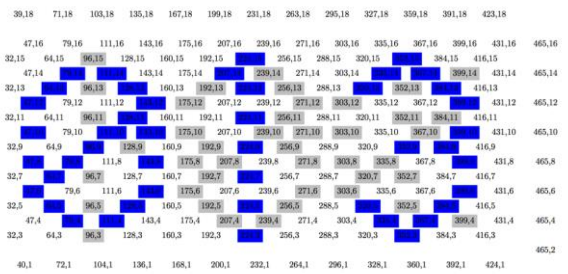




Other goals:

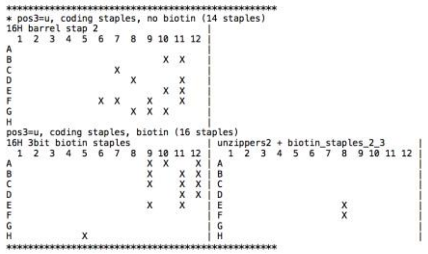
- low strand secondary structure
- low interaction between strands

Bar-coding origami seed for imaging multiple samples at once



some staples of origami seed have version with a biotin

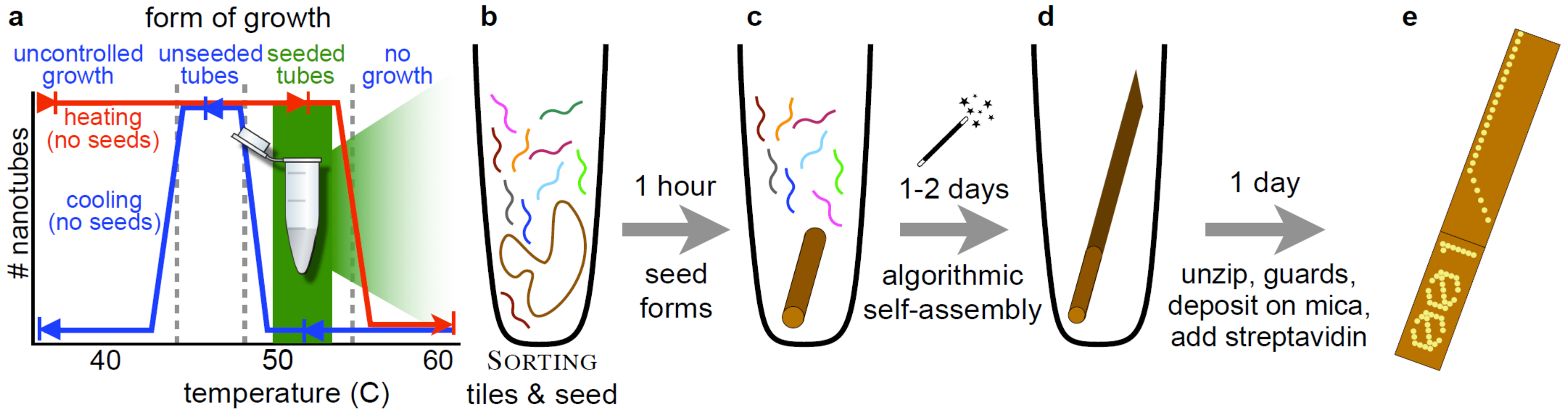
Generate plate map



represents some combination of circuit and input, e.g.,
013 = *“parity circuit, input=011010”*

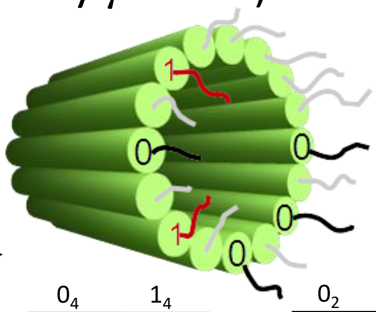
label with streptavidin

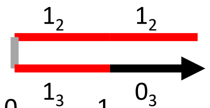
Experimental protocol

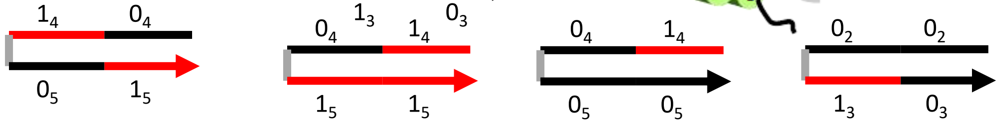


To execute circuit γ on input $x \in \{0,1\}^*$:

- Mix
  - origami seed (bar-coded to identify γ and x)
  - "adapter" strands encoding x
  - tiles computing γ



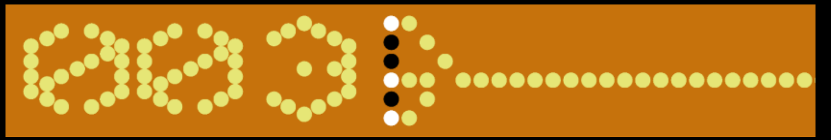




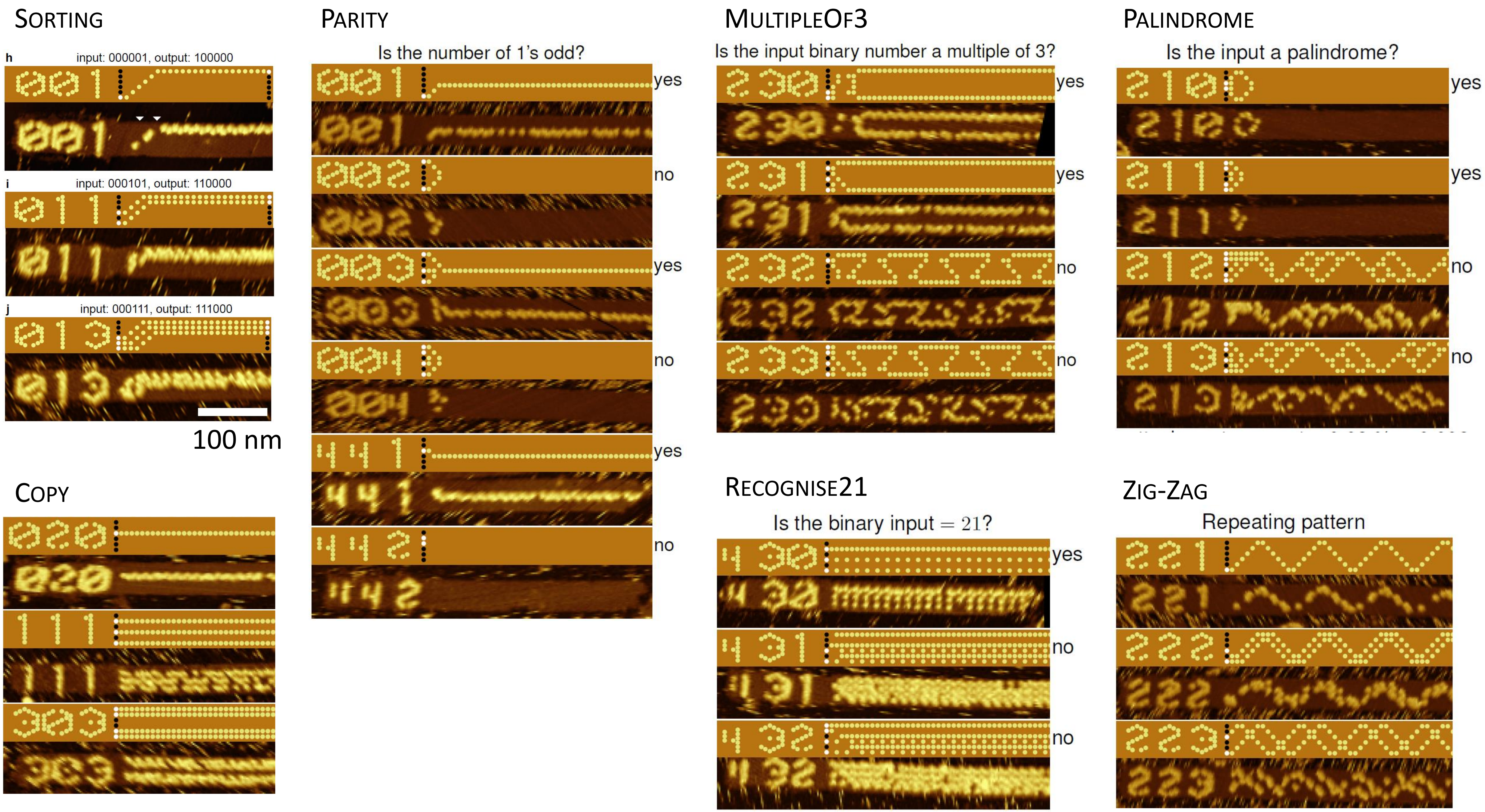
- Anneal 90° C to 50.9° C in 1 hour (*origami seeds form*)
- Hold at 50.9° C for 1-2 days (*tiles grow tubes from seed*)
- Add "unzipper" strands (remove seam to convert tube to ribbon)
- Add "guard" strands (complements of output sticky ends, to deactivate tiles)
- Deposit on mica, buffer wash, add streptavidin, AFM

Results

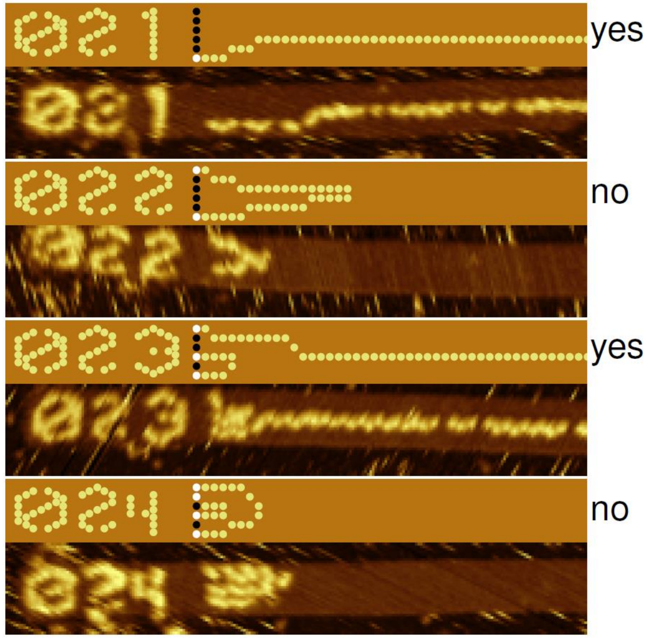
```
def test_parity():
    actual = parity('100101')

    expected =
```



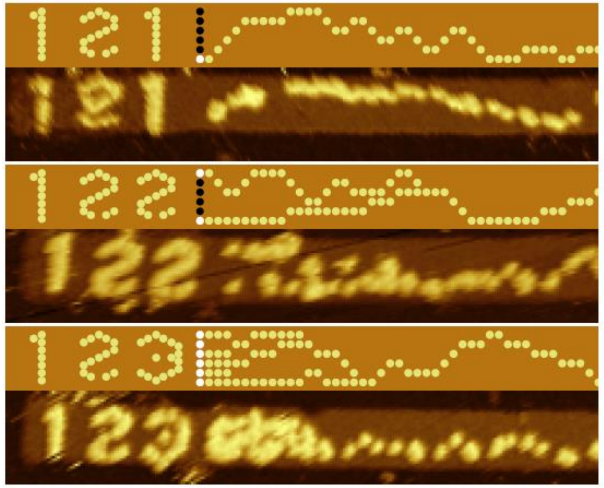
```
    assertEquals(expected, actual)
```


SORTING
h input: 000001, output: 100000
i input: 000101, output: 110000
j input: 000111, output: 111000
100 nm
COPY
PARITY
Is the number of 1's odd?
yes
no
yes
no
yes
no
MULTIPLEOF3
Is the input binary number a multiple of 3?
yes
yes
no
no
PALINDROME
Is the input a palindrome?
yes
yes
no
no
RECOGNISE21
Is the binary input = 21?
yes
no
no
ZIG-ZAG
Repeating pattern

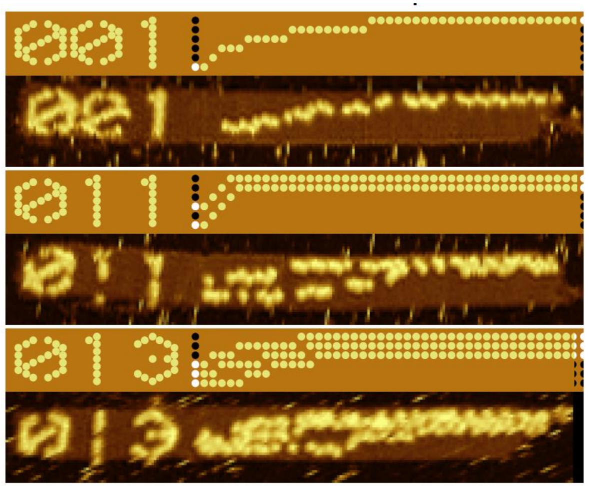
LazyParity



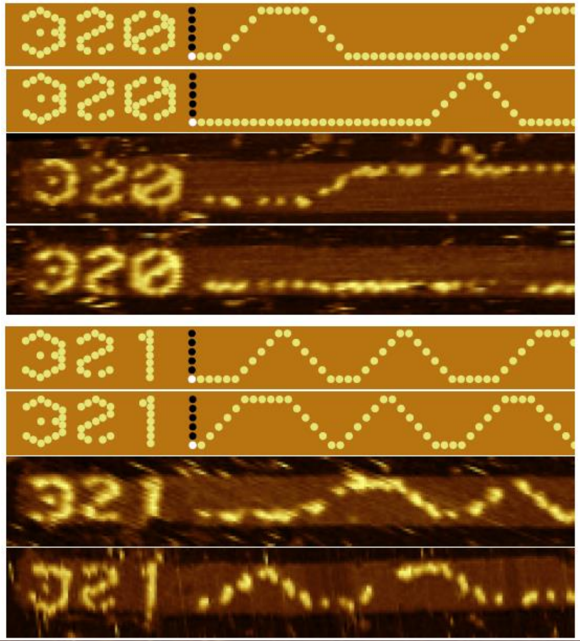
LeaderElection



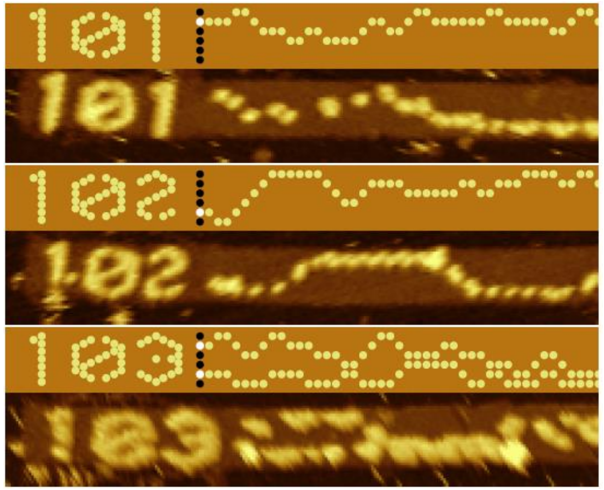
LazySorting



Waves

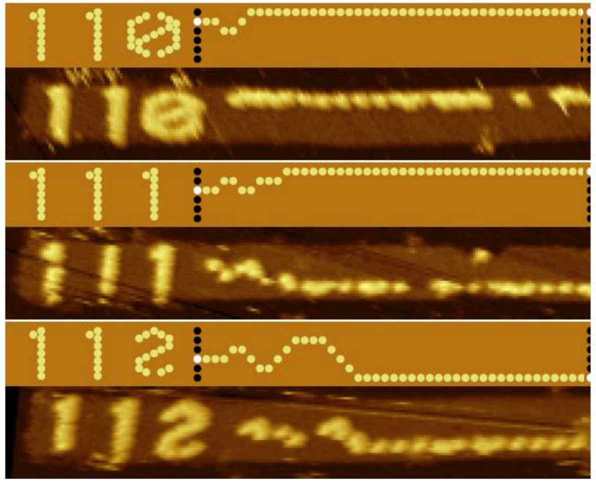


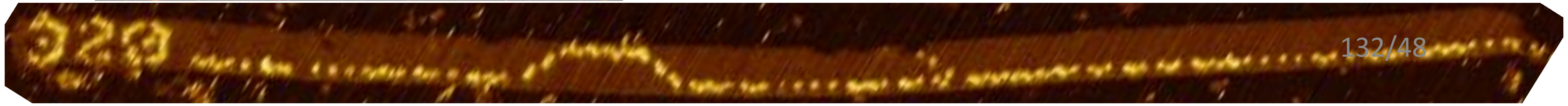
RandomWalkingBit



AbsorbingRandomWalkingBit

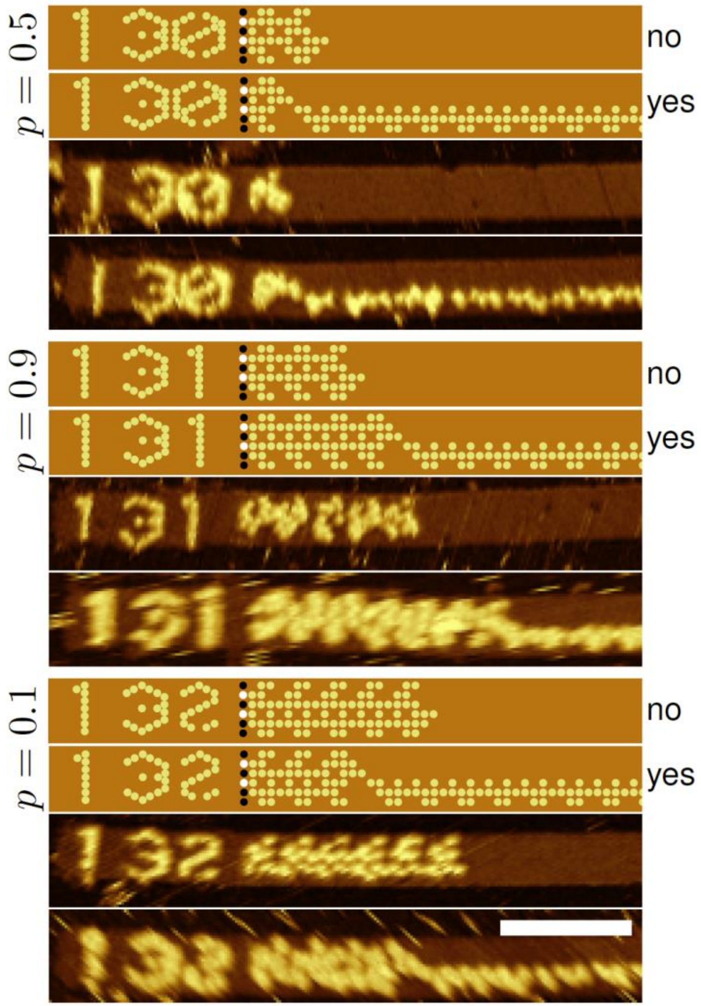
Random walker absorbs to top/bottom

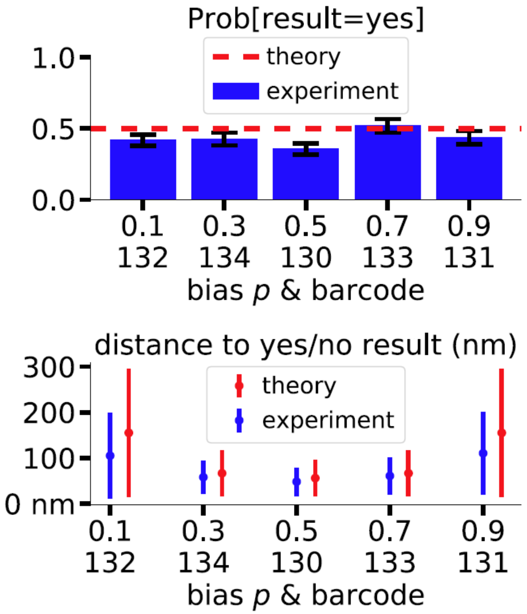




FAIRCOIN

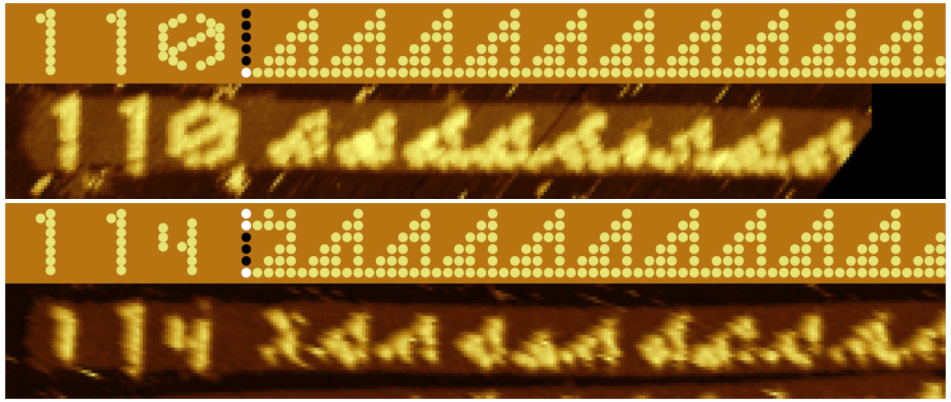
Unbiasing a biased coin





RULE110

Simulation of a cellular automaton



Counting to 63

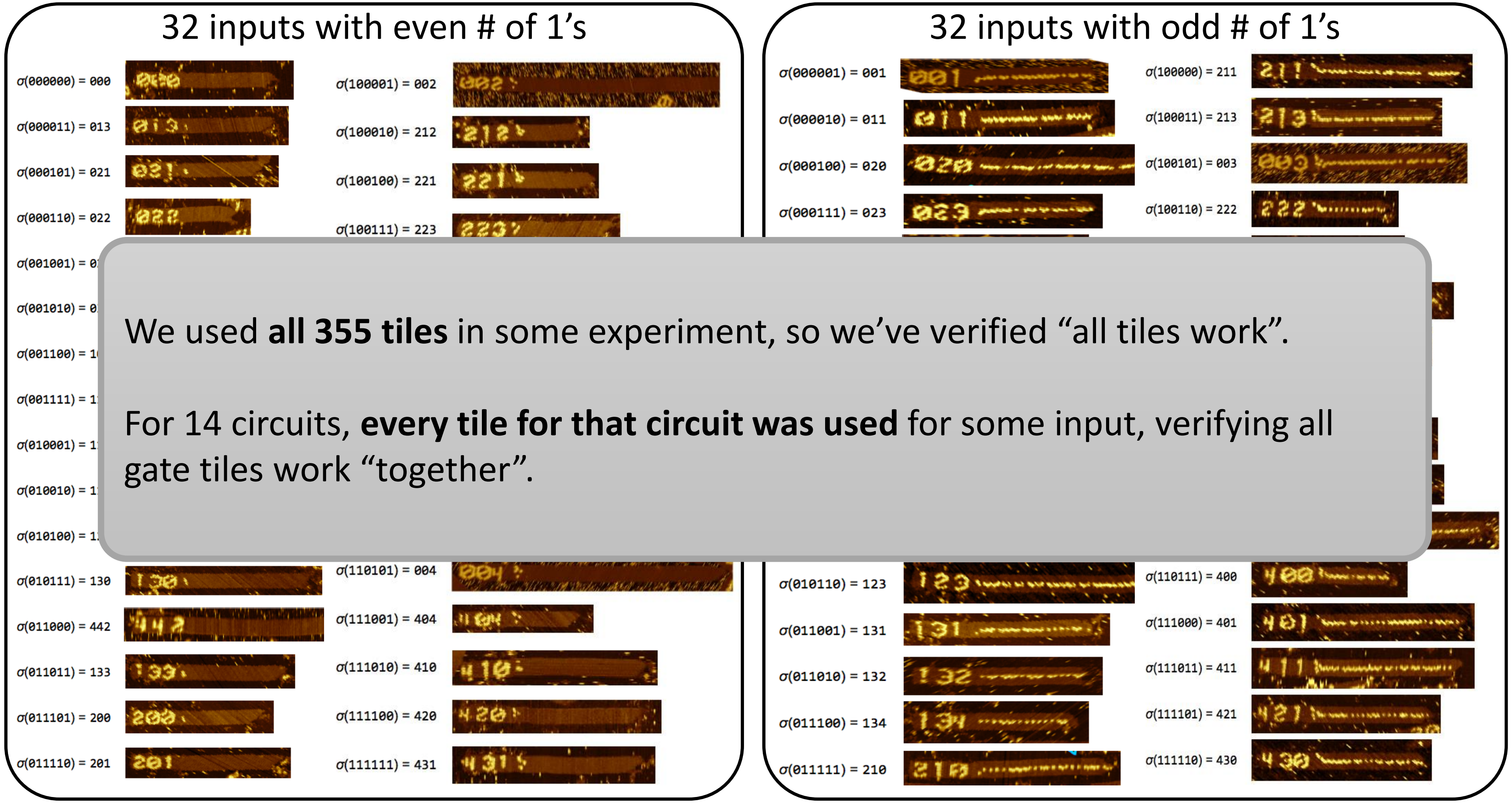
Circuit with 63 distinct strings

Is there a 64-counter?

No!
Proof by Tristan Stérin, Maynooth University
Consequence of following theorem:
No Boolean function computes an odd permutation if some output bit does not depend on all input bits.

Parity tested on all inputs

2^6 = 64 inputs with 6 bits



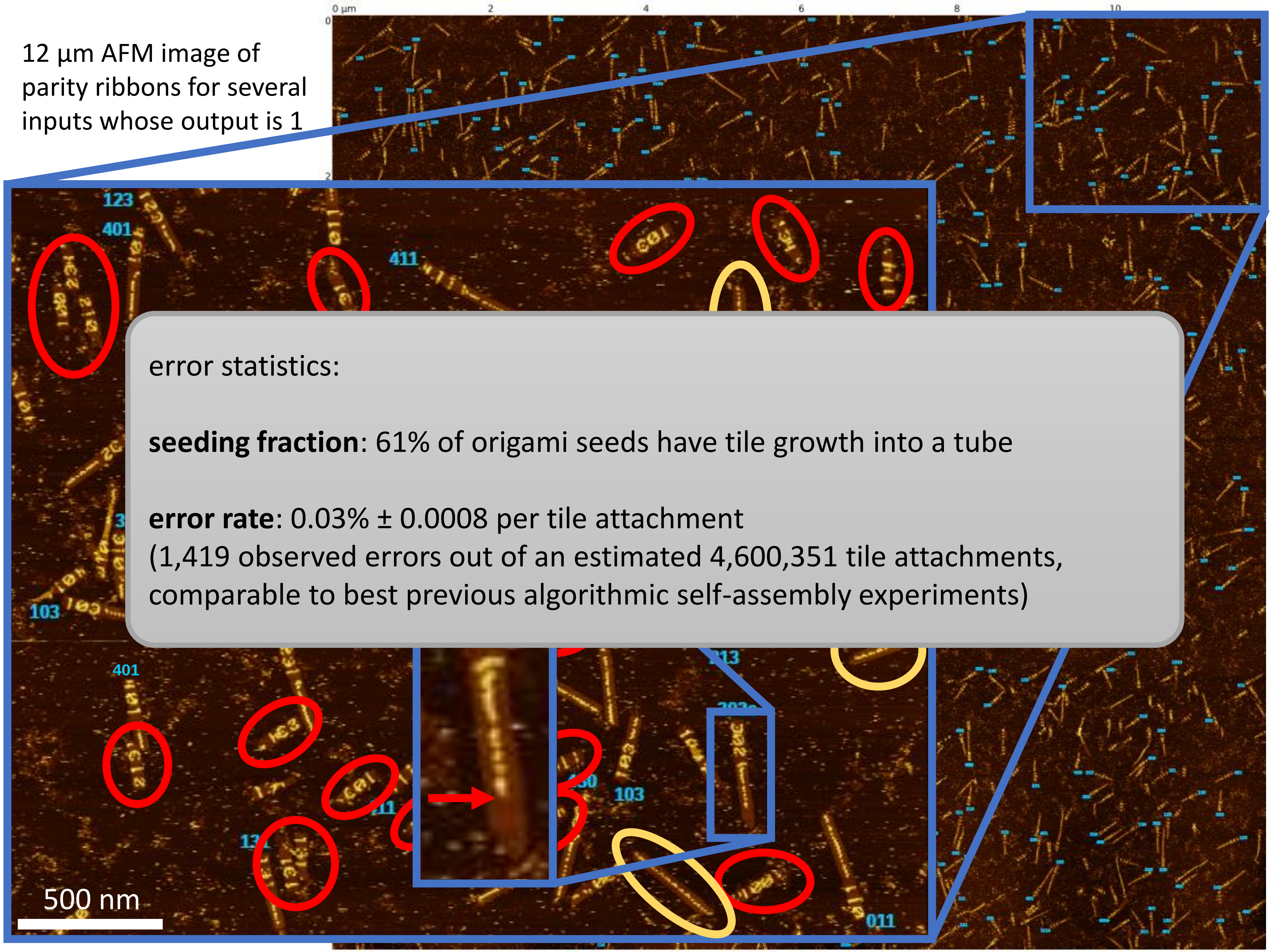
We used **all 355 tiles** in some experiment, so we've verified "all tiles work".

For 14 circuits, **every tile for that circuit was used** for some input, verifying all gate tiles work "together".

σ(6-bit input) = 3-digit barcode representing that input

150 nm

12 µm AFM image of parity ribbons for several inputs whose output is 1



error statistics:

seeding fraction: 61% of origami seeds have tile growth into a tube

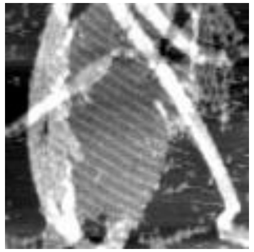
error rate: 0.03% ± 0.0008 per tile attachment
(1,419 observed errors out of an estimated 4,600,351 tile attachments, comparable to best previous algorithmic self-assembly experiments)

What did we learn?

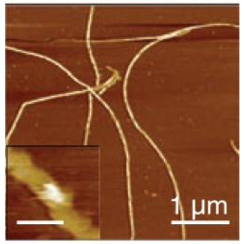
A small(ish) library of molecules can be reprogrammed to self-assemble reliably into many complex patterns, by processing information as they grow.

Contrasting with other self-assembly work:

more algorithmic control than **periodic** self-assembly

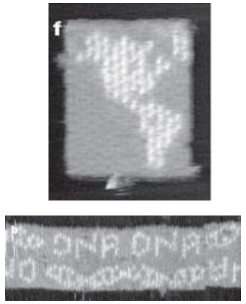


2D tile lattices (Winfree et al., *Nature* 1998)



1D tile tubes (Yin et al., *Science* 2008)

fewer types of DNA strands required than **uniquely-addressed** self-assembly

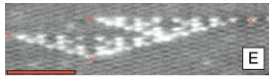


DNA origami (Rothemund, *Nature* 2006)

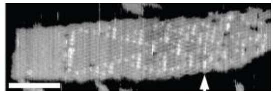
hard-coded tile lattice (Wei et al., *Nature* 2012)

order of magnitude more tile types available than previous **algorithmic** self-assembly

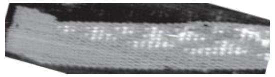
double-crossover tile lattices



(Rothemund et al., *PLoS Bio* 2004)



(Fujibayashi et al., *Nano Letters* 2008)



(Barish et al., *PNAS* 2009)

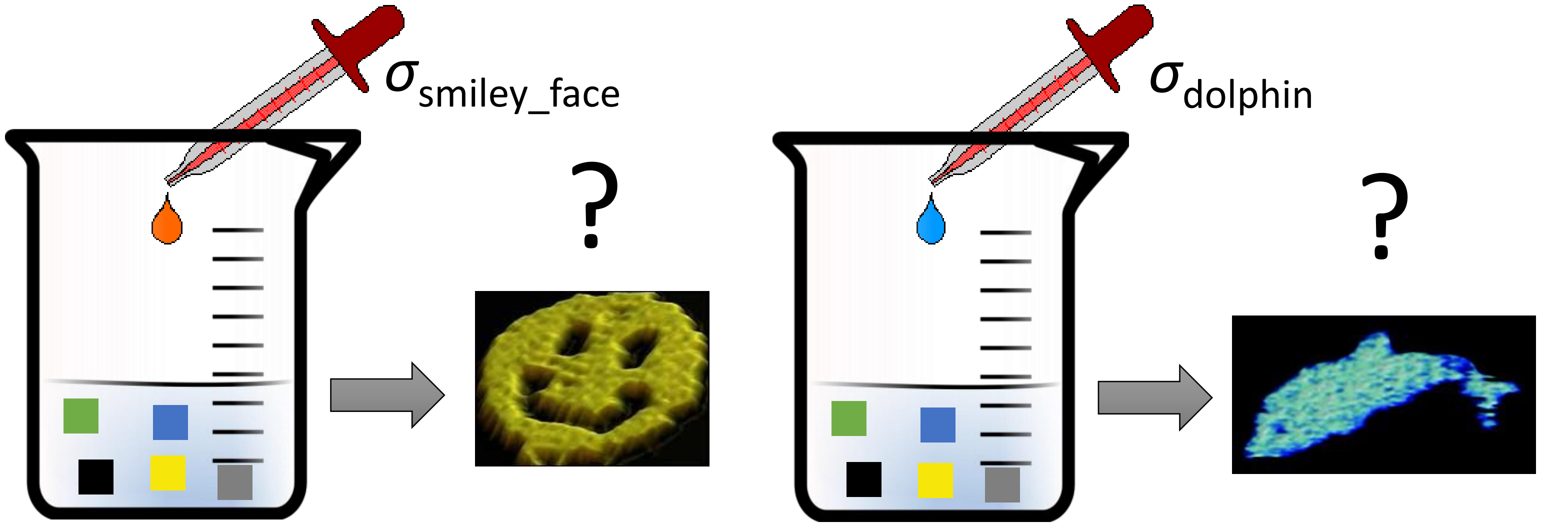
(Evans, *Ph.D. thesis* 2014)

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we run algorithms to **grow interesting shapes**?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.



These tiles are universally programmable for building any shape.

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]