Thermodynamic binding networks

How to self-assemble molecules reliably without knowing too much about them (substrate-independent self-assembly)

David Doty (UC-Davis)

Joint work with **Trent Rogers** (UArk), **David Soloveichik** (UT-Austin), **Chris Thachuk** (Caltech), **Damien Woods** (Inria)

Self-Assembly, Geometry, and Computation Workshop at UCNC

IUT Sénart-Fontainebleau, June 2018

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

IUT
SENART
FONTAINEBLEAU

Acknowledgements

co-authors

Trent Rogers

David Soloveichik

Chris Thachuk

Damien Woods

paper in proceedings of **DNA 23**:
23rd International Meeting on DNA Computing and Molecular Programming

Outline

Part 1: Computation via molecular binding (kinetically)

Part 2: Computation via molecular binding (thermodynamically)

Part 3: Thermodynamic self-assembly

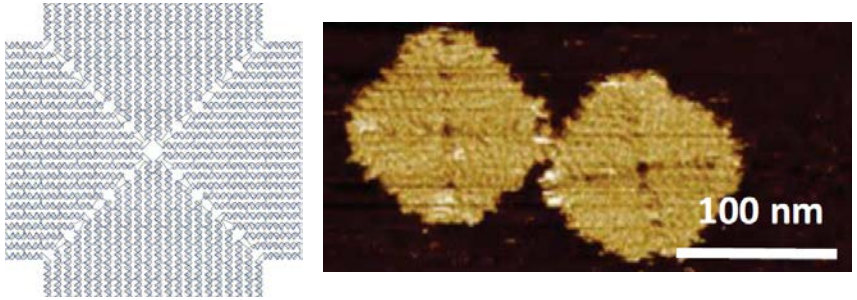
Computation via molecular binding: Two models

Computation via molecular binding: Two models

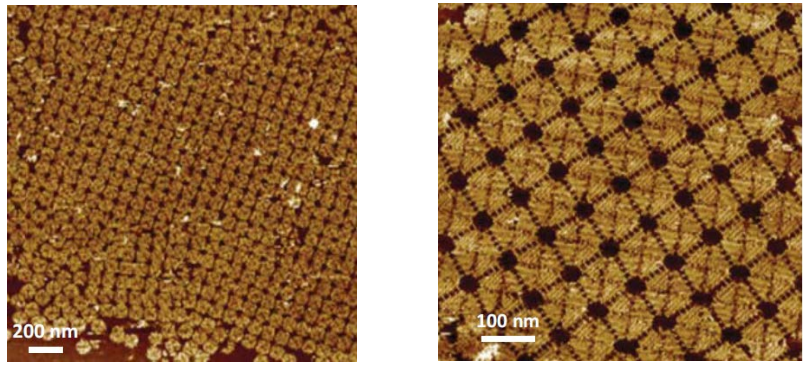
DNA tile self-assembly

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

tile



lattice

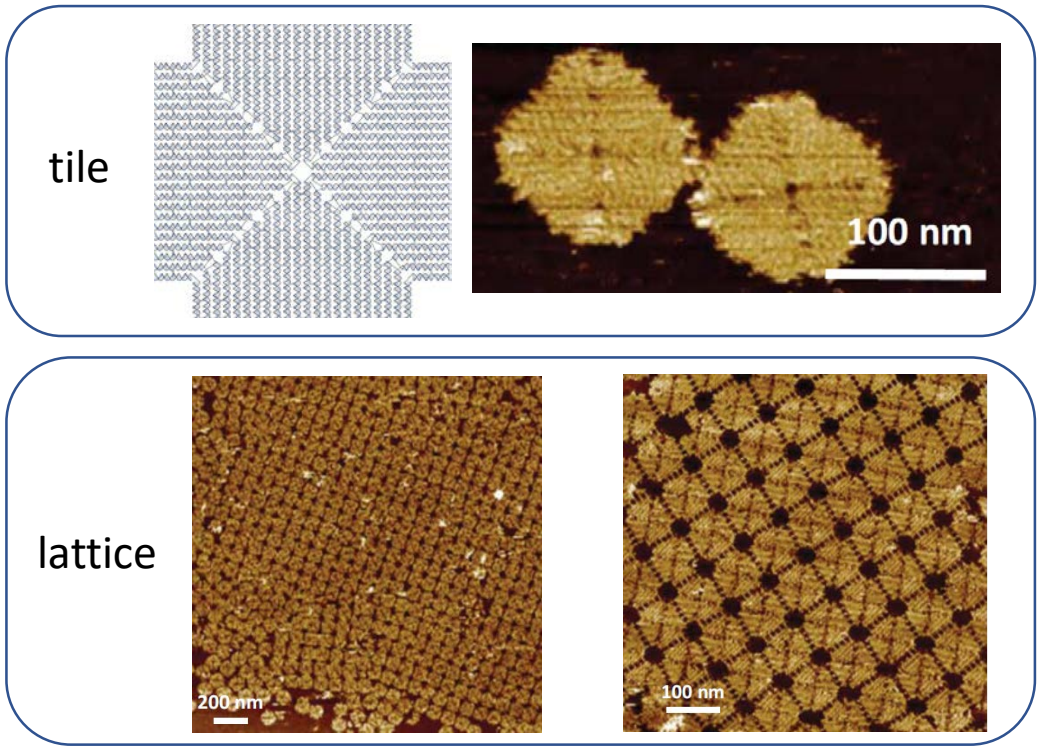


Source: *Programmable disorder in random DNA tilings*. Tikhomirov, Petersen, Qian, Nature Nanotechnology 2017

Computation via molecular binding: Two models

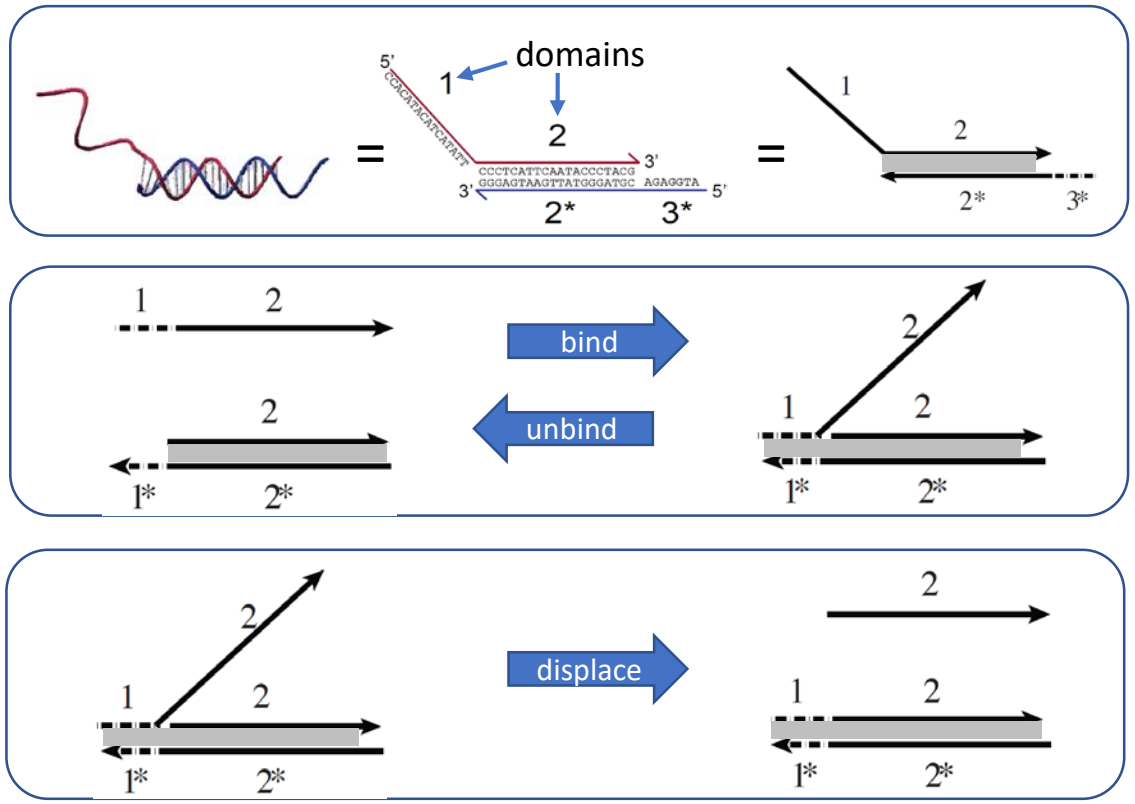
DNA tile self-assembly

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



DNA strand displacement

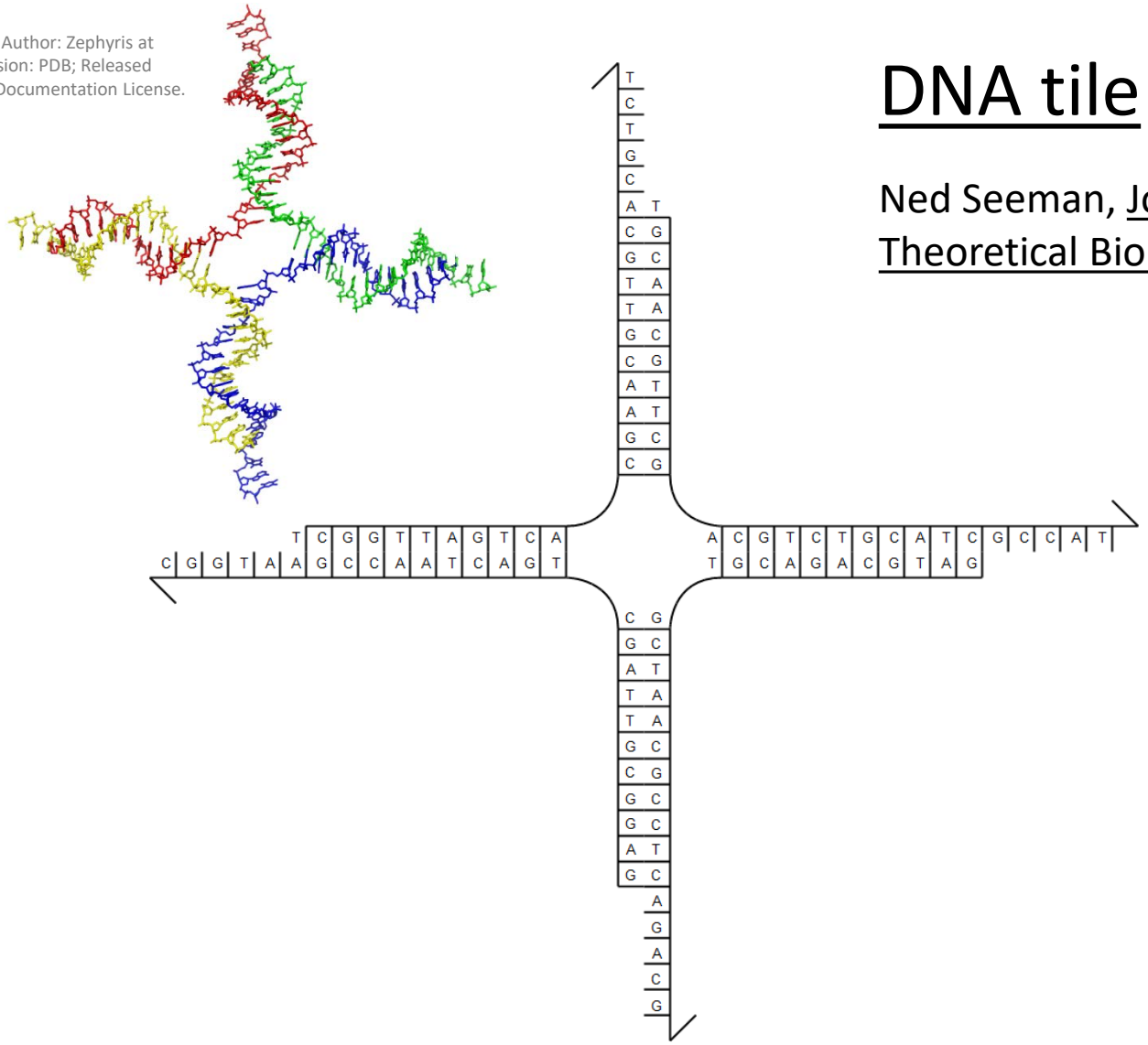
DNA complexes exchange strands with each other by displacing binding domains



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.

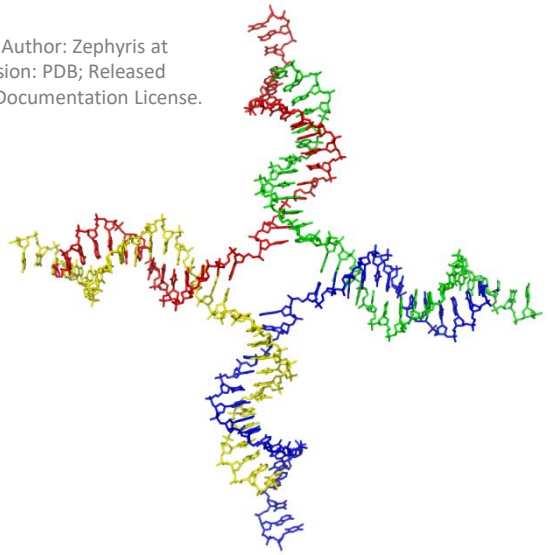


DNA tile

Ned Seeman, Journal of Theoretical Biology 1982

Practice of DNA tile self-assembly

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

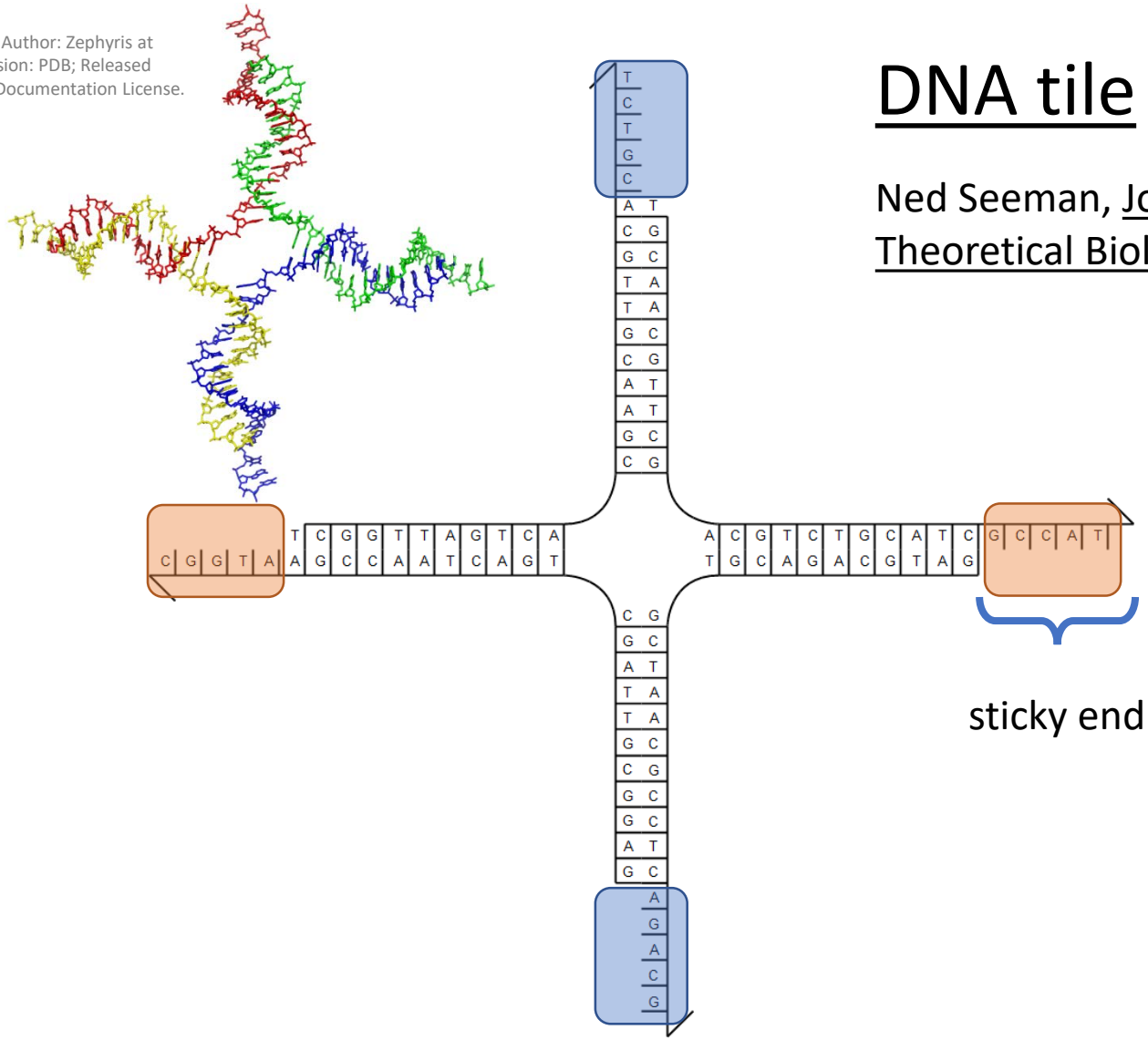


DNA tile

Ned Seeman, Journal of Theoretical Biology 1982

Practice of DNA tile self-assembly

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



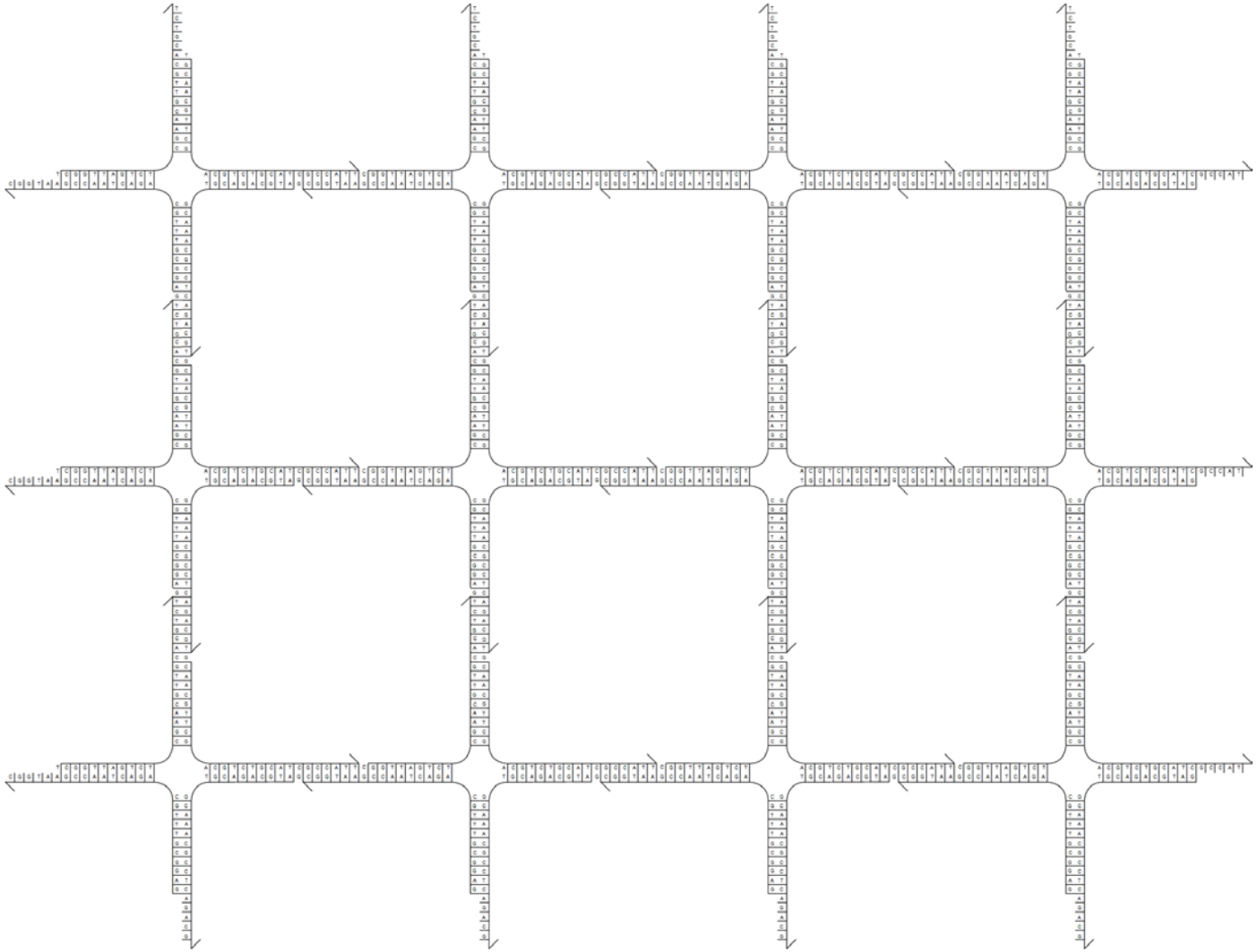
DNA tile

Ned Seeman, Journal of Theoretical Biology 1982

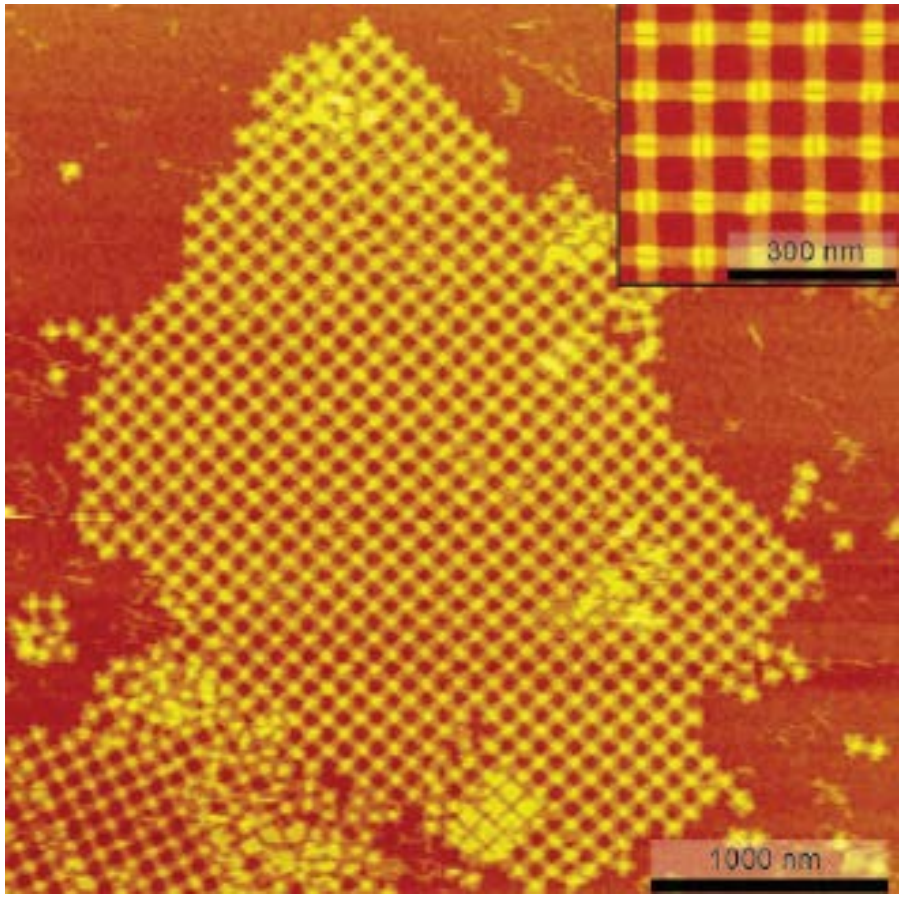
sticky end

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

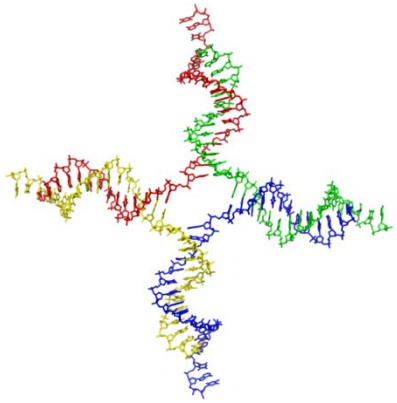
Theory of *algorithmic* self-assembly

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

Erik Winfree

Abstract Tile Assembly Model

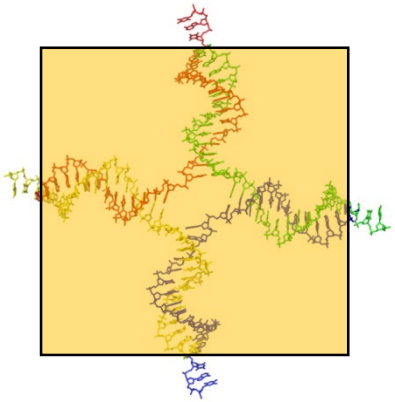
Erik Winfree,
Ph.D. thesis,
Caltech 1998



Abstract Tile Assembly Model

Erik Winfree,
Ph.D. thesis,
Caltech 1998

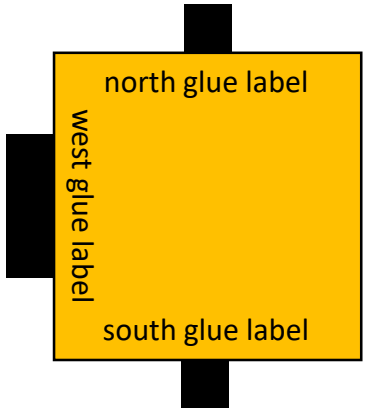
- **tile type** = unit square



Abstract Tile Assembly Model

Erik Winfree,
Ph.D. thesis,
Caltech 1998

- **tile type** = unit square
- each side has a **glue** with a **label** and **strength** (0, 1, or 2)



strength 0

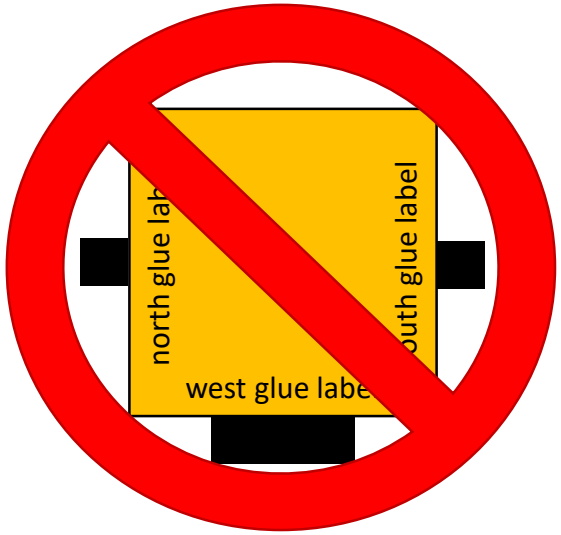
strength 1 (weak)

strength 2 (strong)

Abstract Tile Assembly Model

Erik Winfree,
Ph.D. thesis,
Caltech 1998

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



strength 0

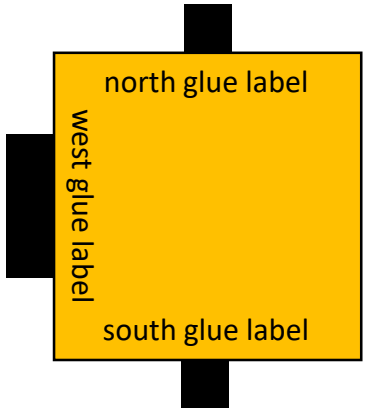
strength 1 (weak)

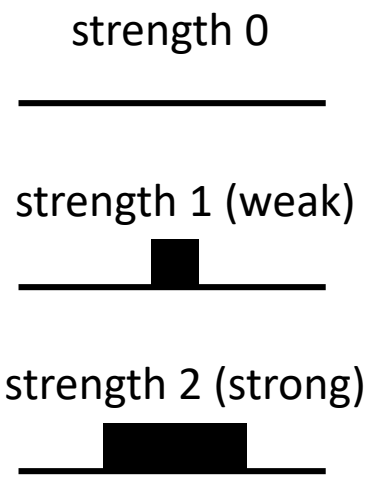
strength 2 (strong)

Abstract Tile Assembly Model

Erik Winfree,
Ph.D. thesis,
Caltech 1998

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



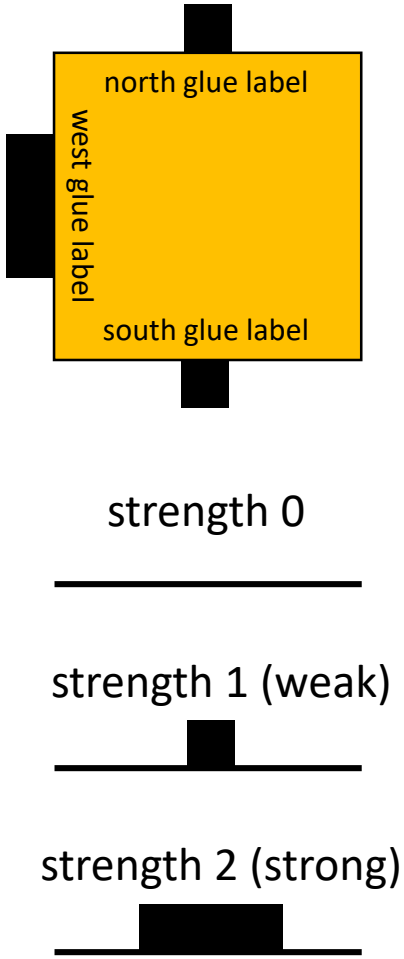


- finitely many tile **types**
- infinitely many **tiles**: copies of each type

Abstract Tile Assembly Model

Erik Winfree,
Ph.D. thesis,
Caltech 1998

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

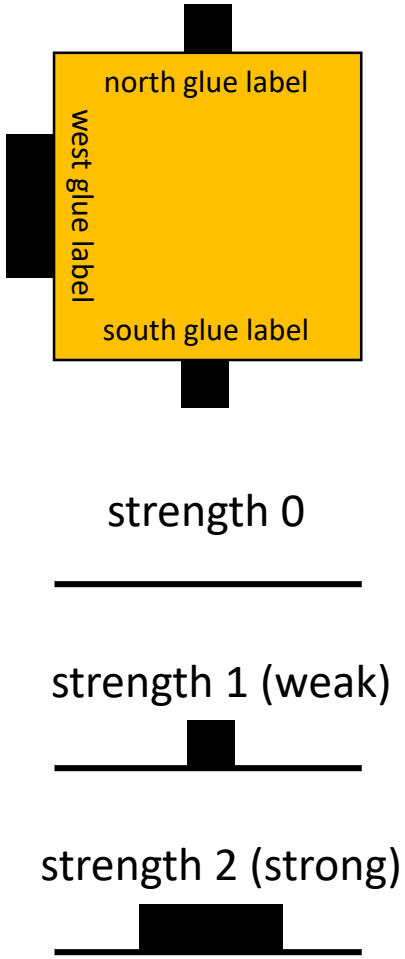


- finitely many tile **types**
- infinitely many **tiles**: copies of each type
- assembly starts as a single copy of a special **seed** tile

Abstract Tile Assembly Model

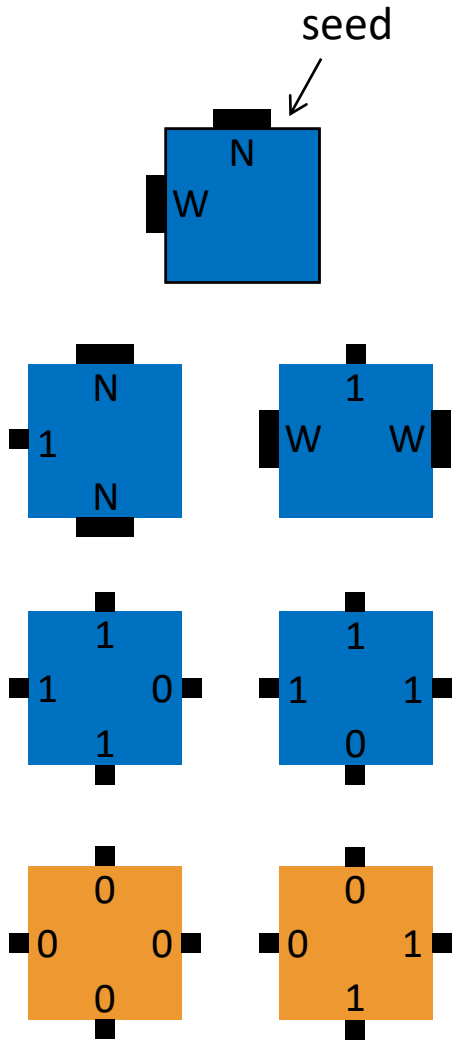
Erik Winfree,
Ph.D. thesis,
Caltech 1998

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

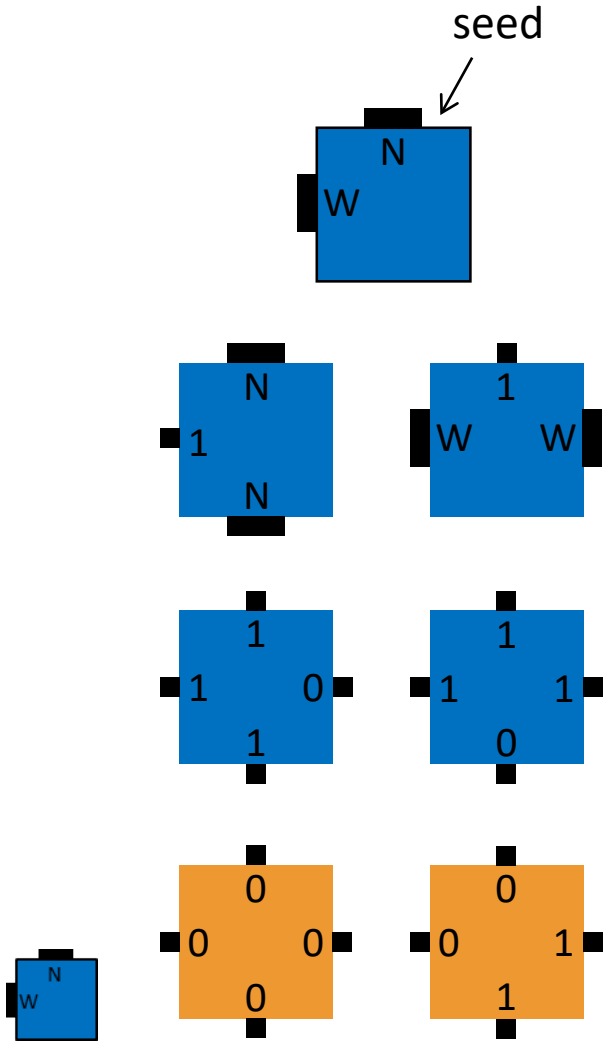


- finitely many tile **types**
- infinitely many **tiles**: copies of each type
- assembly starts as a single copy of a special **seed** tile
- tile can bind to the assembly if total binding strength ≥ 2 (two weak glues or one strong glue)

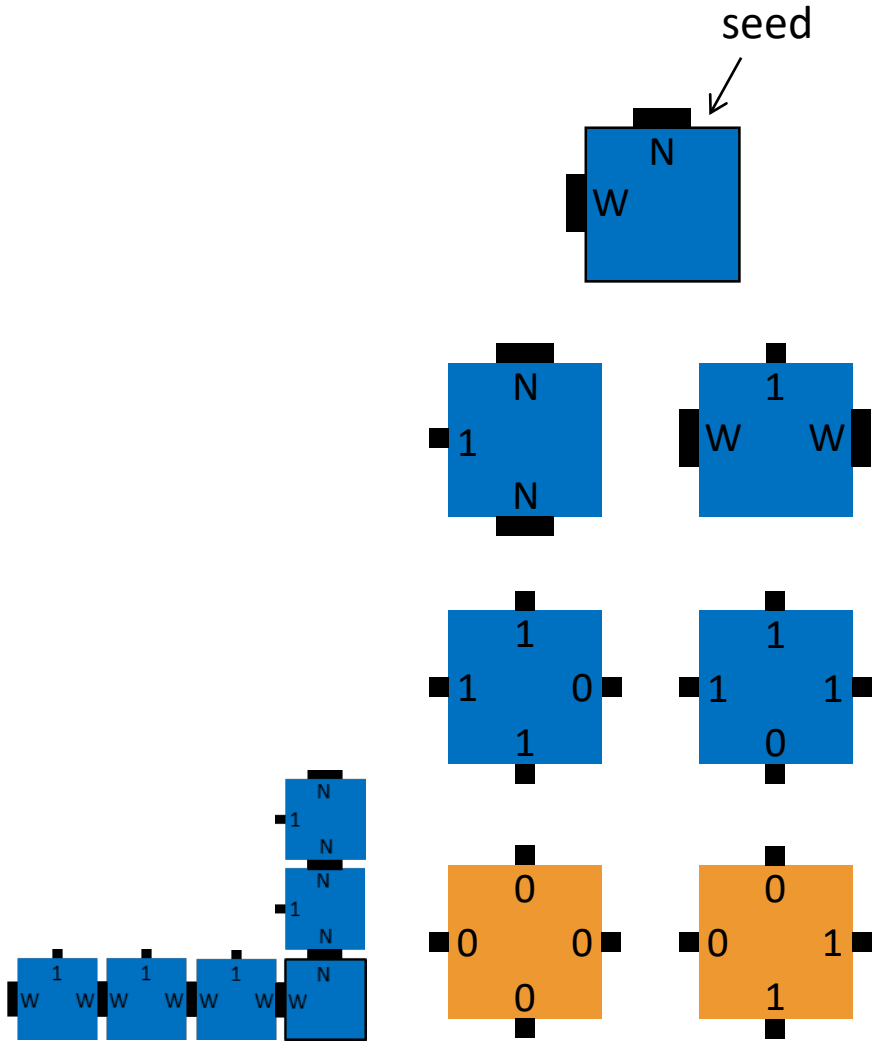
Example tile set



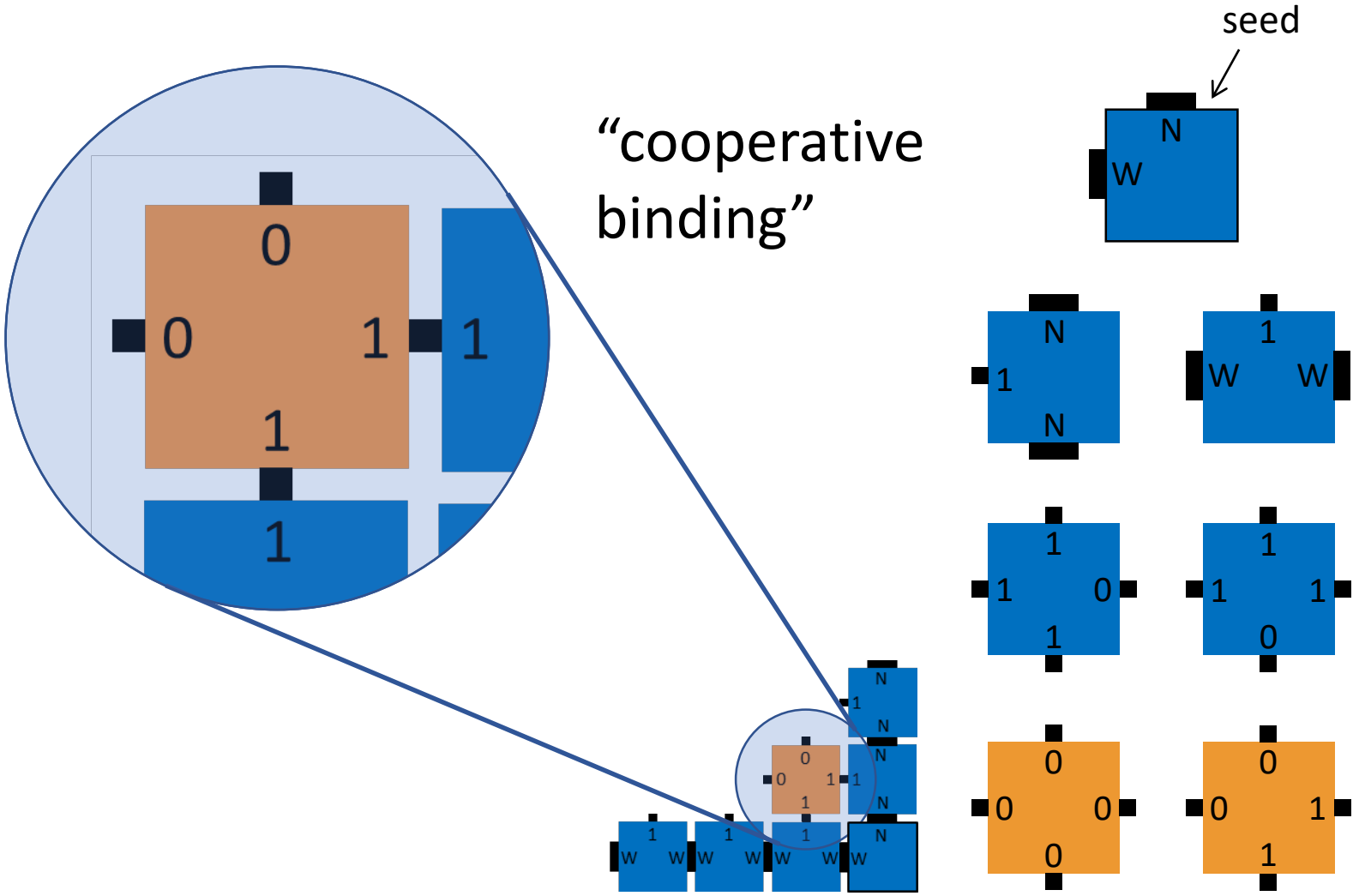
Example tile set



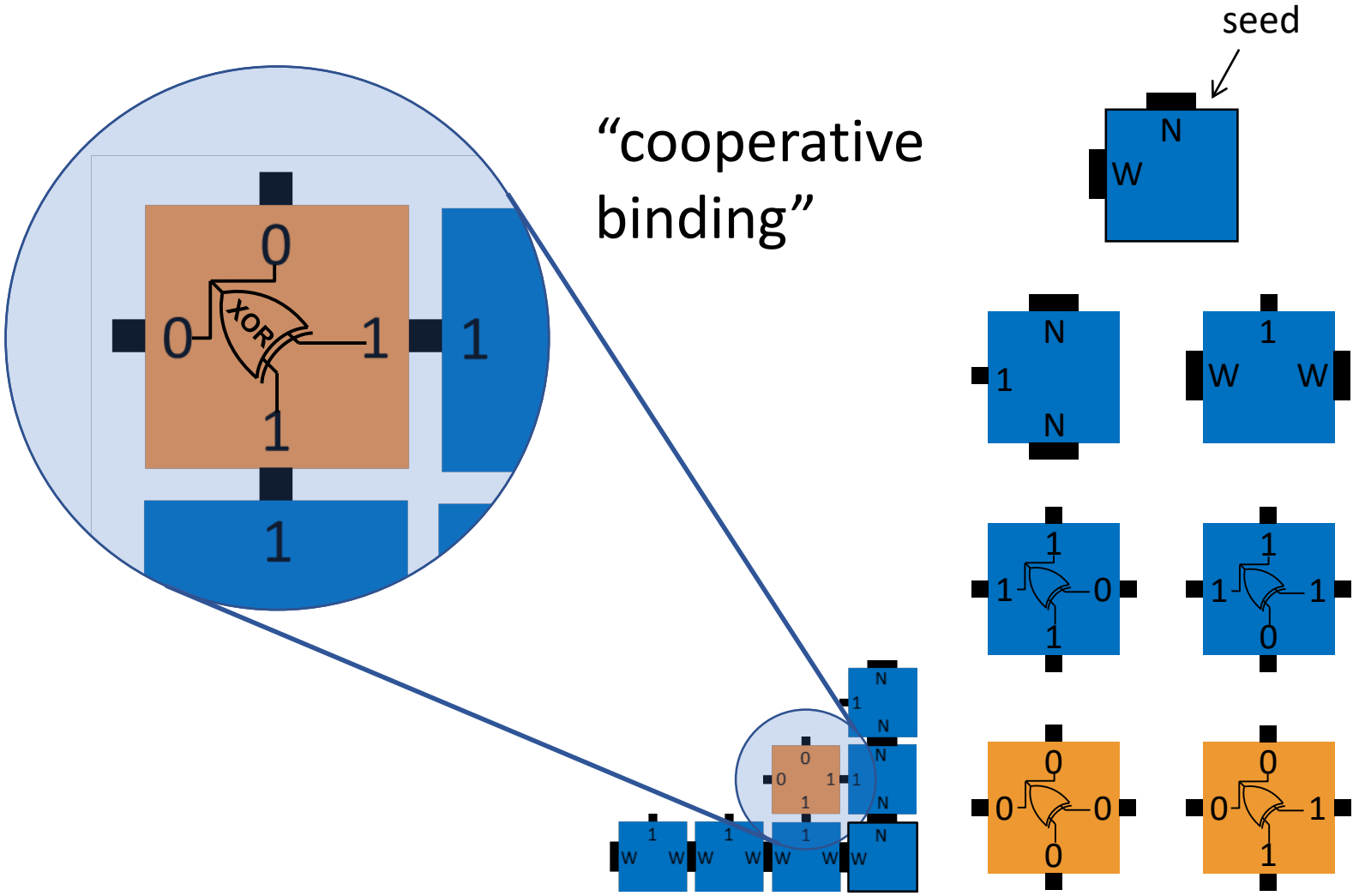
Example tile set



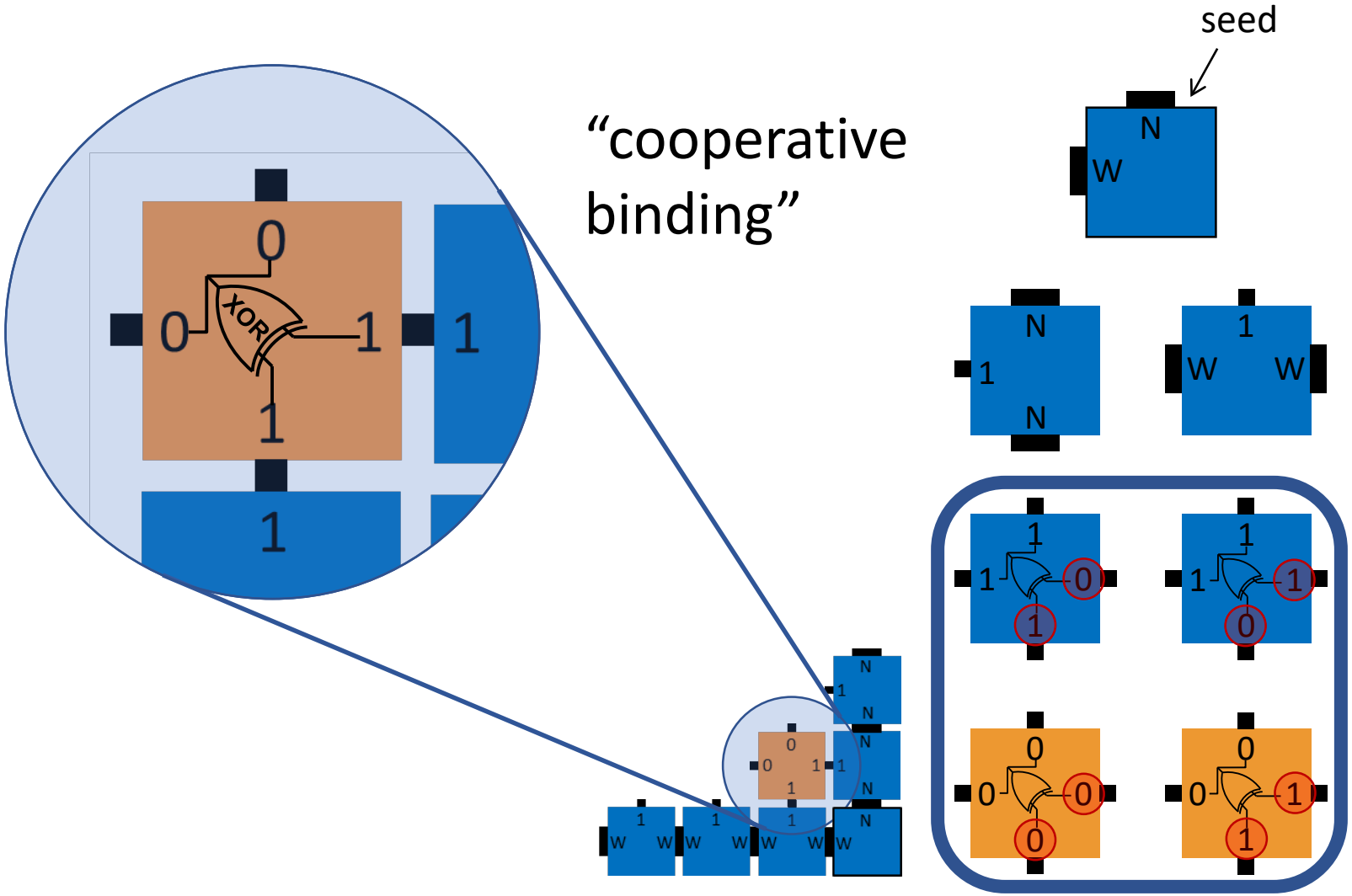
Example tile set



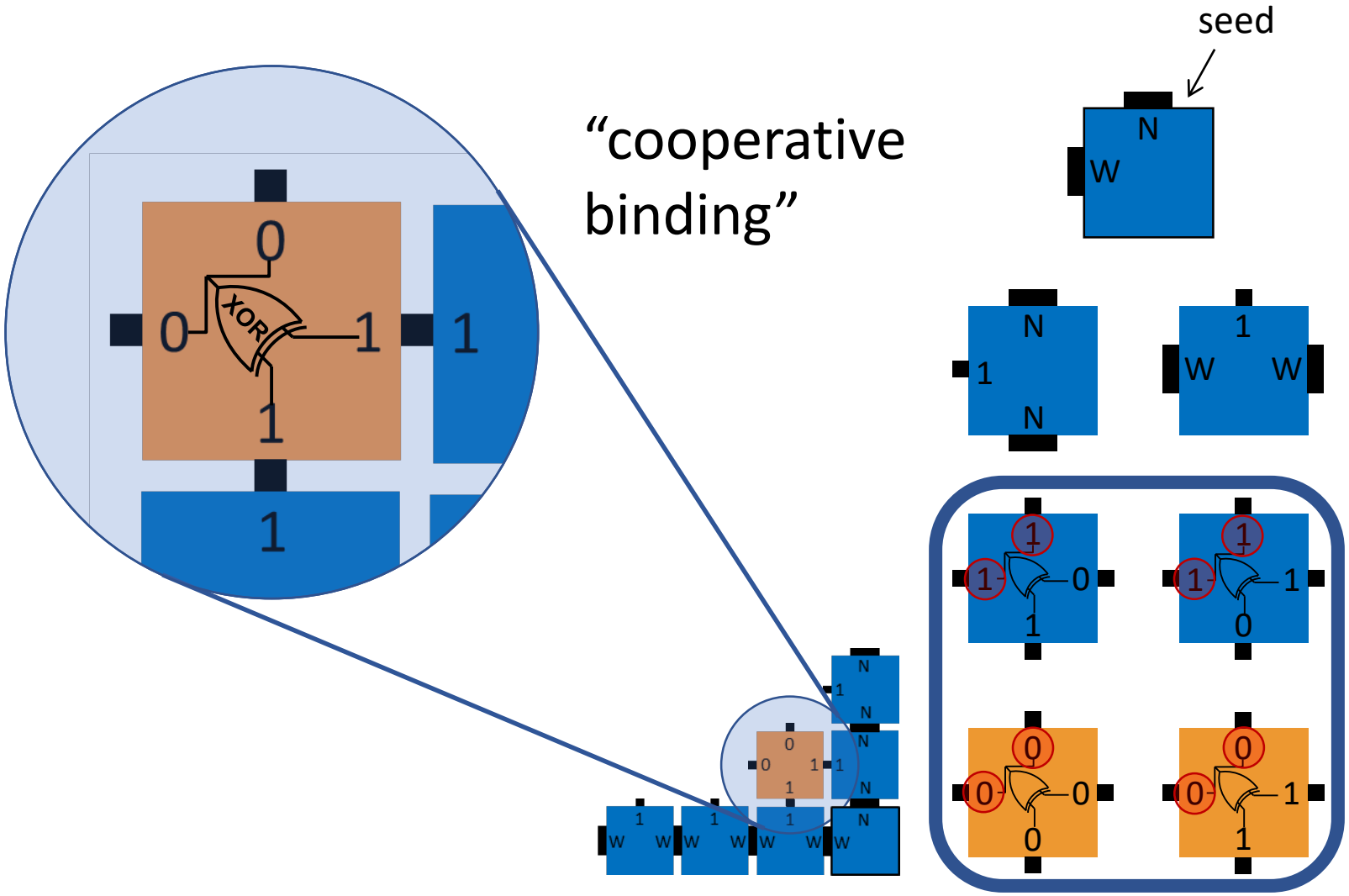
Example tile set



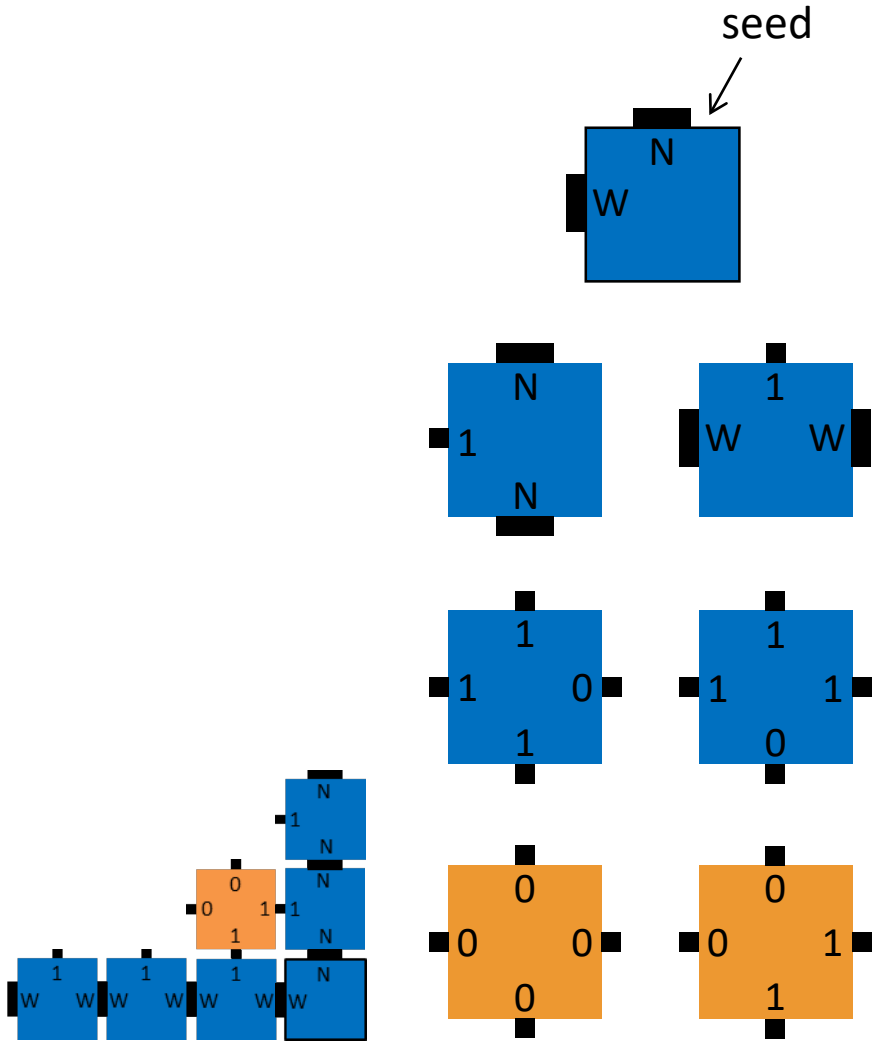
Example tile set

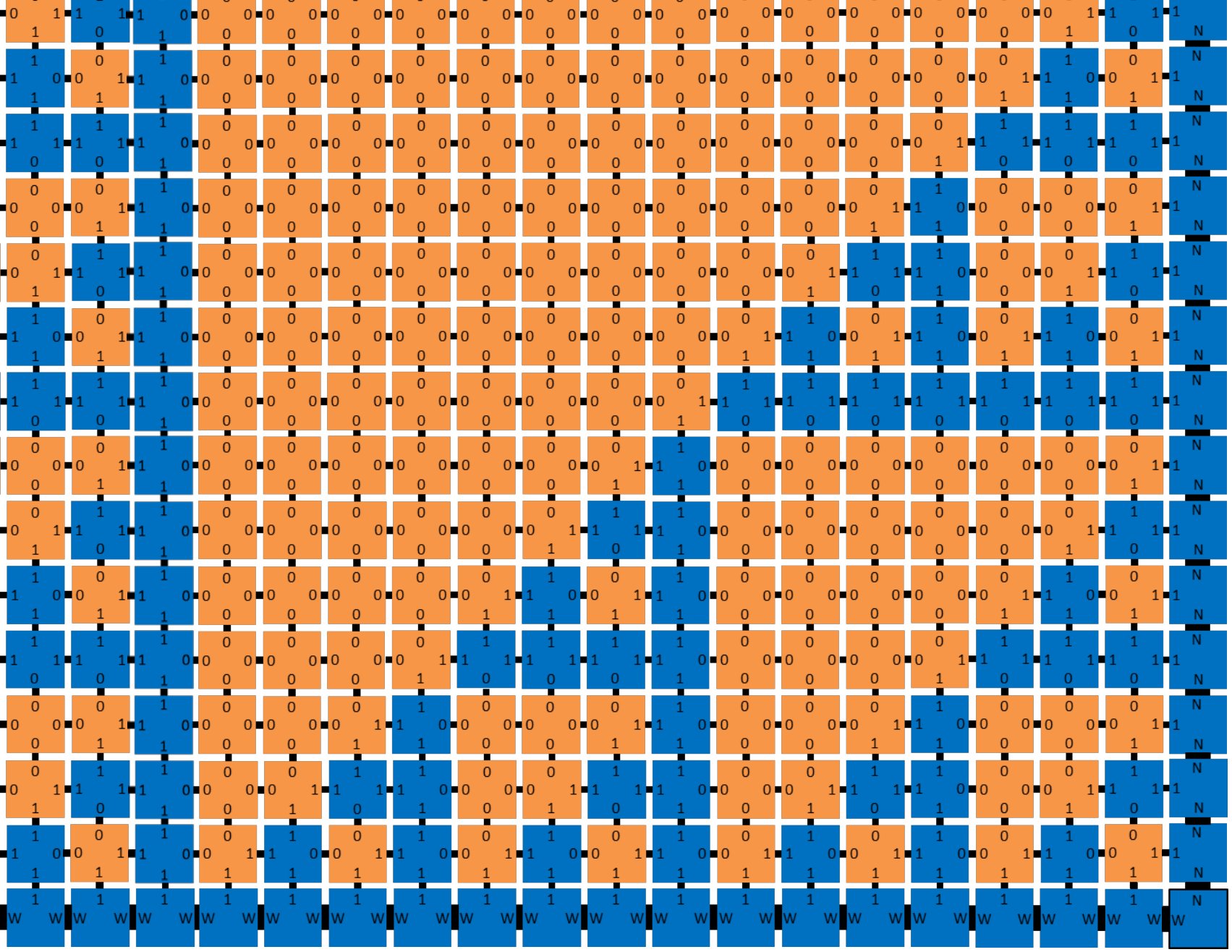


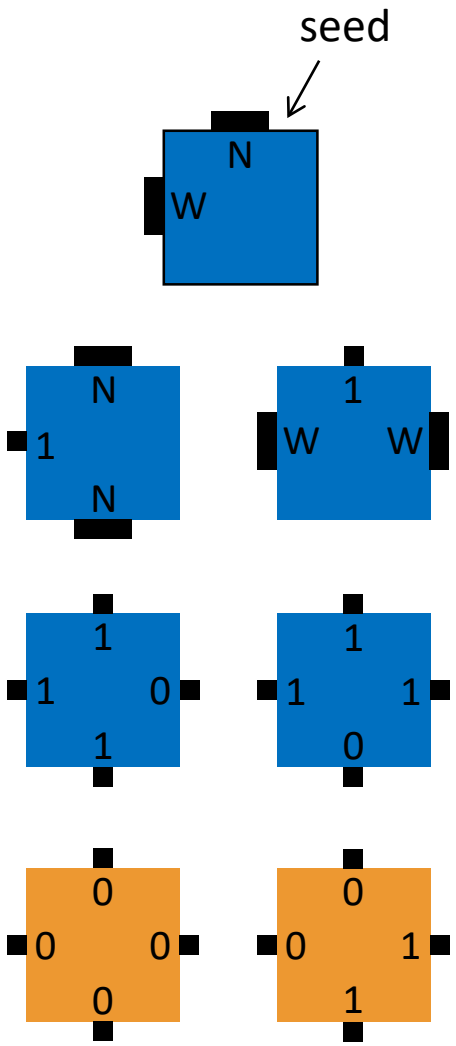
Example tile set

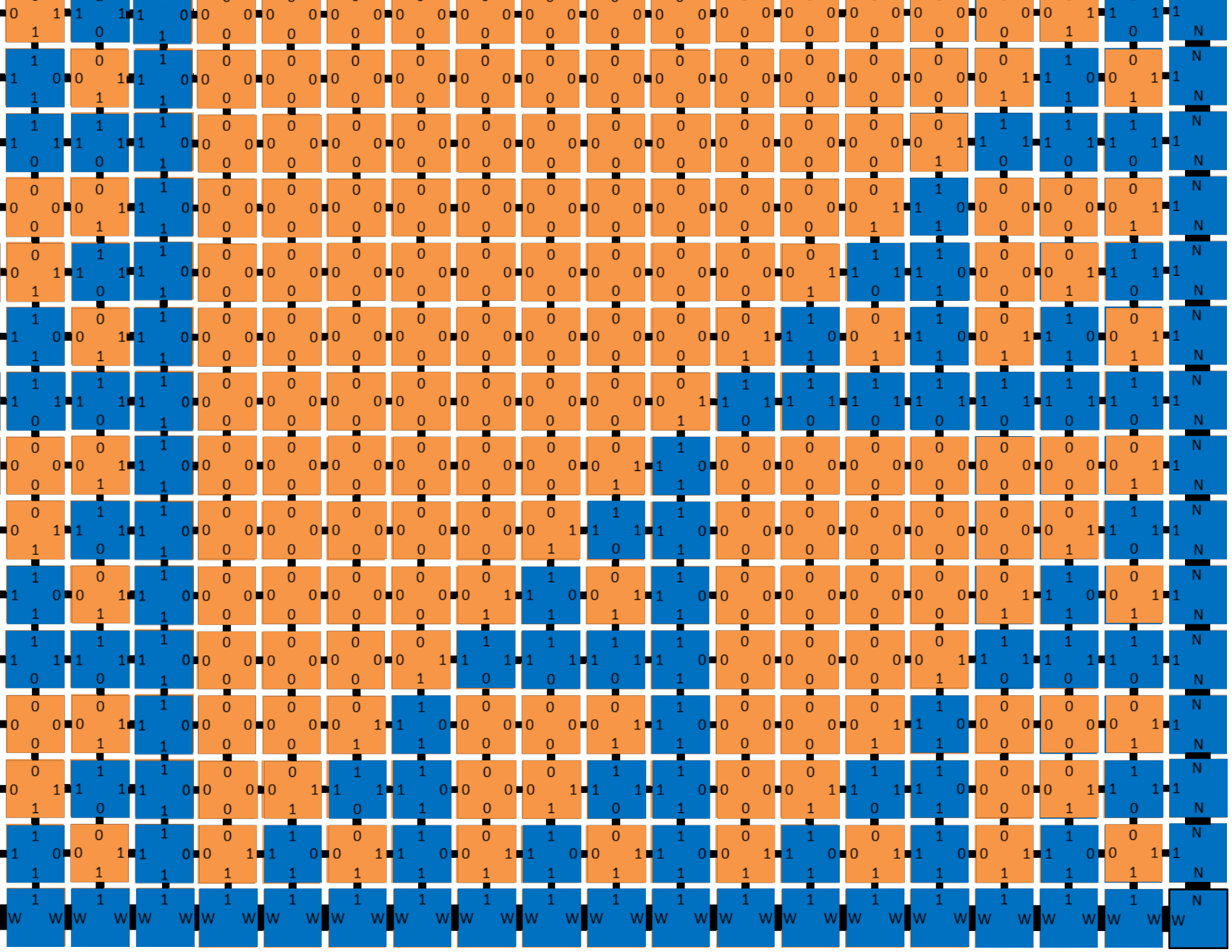


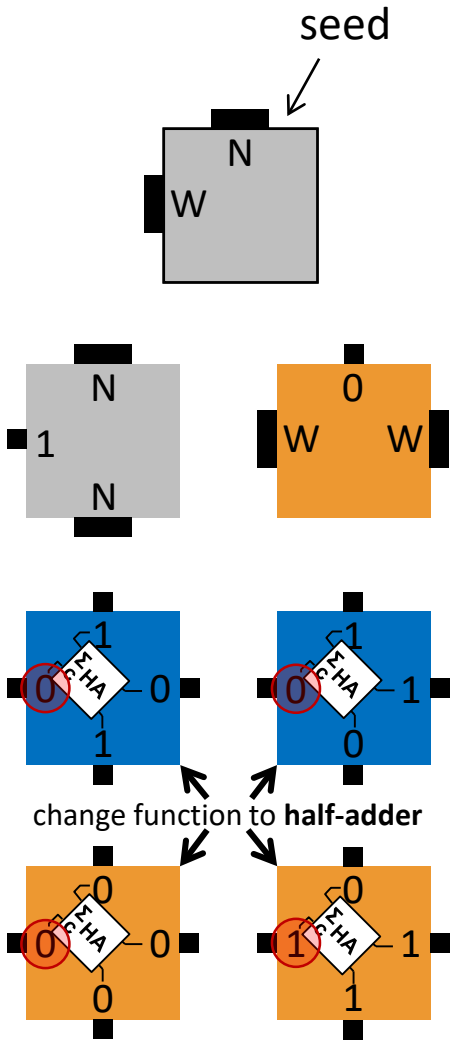
Example tile set

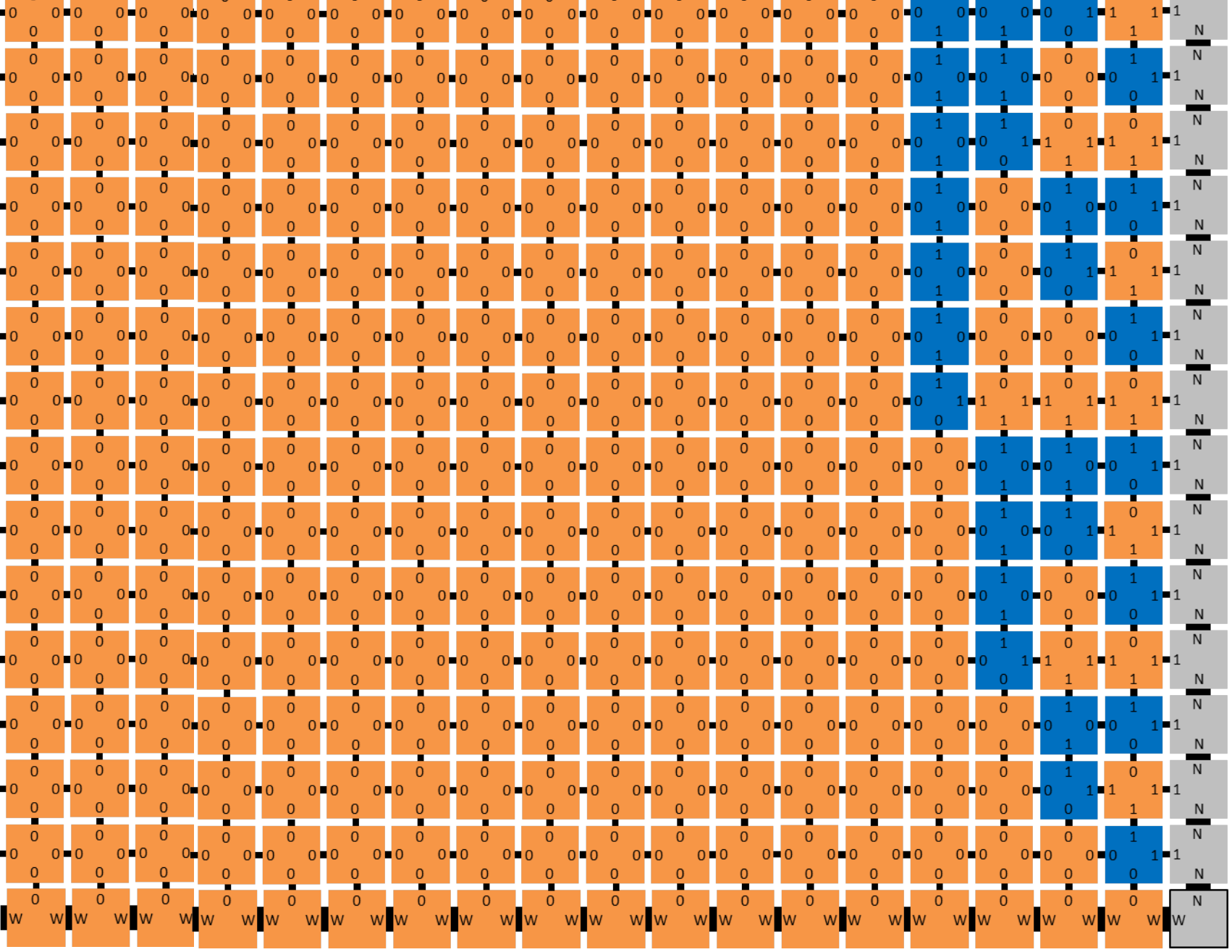


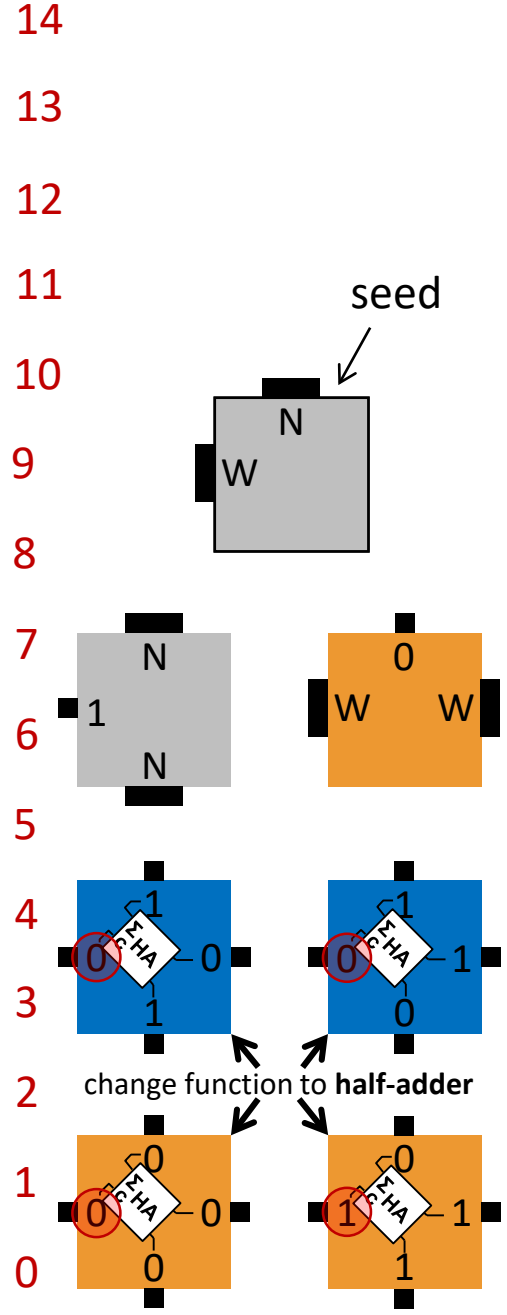



seed
N
W
N
1
N
1
W
W
1
1
0
1
1
1
0
0
0
0
0
1

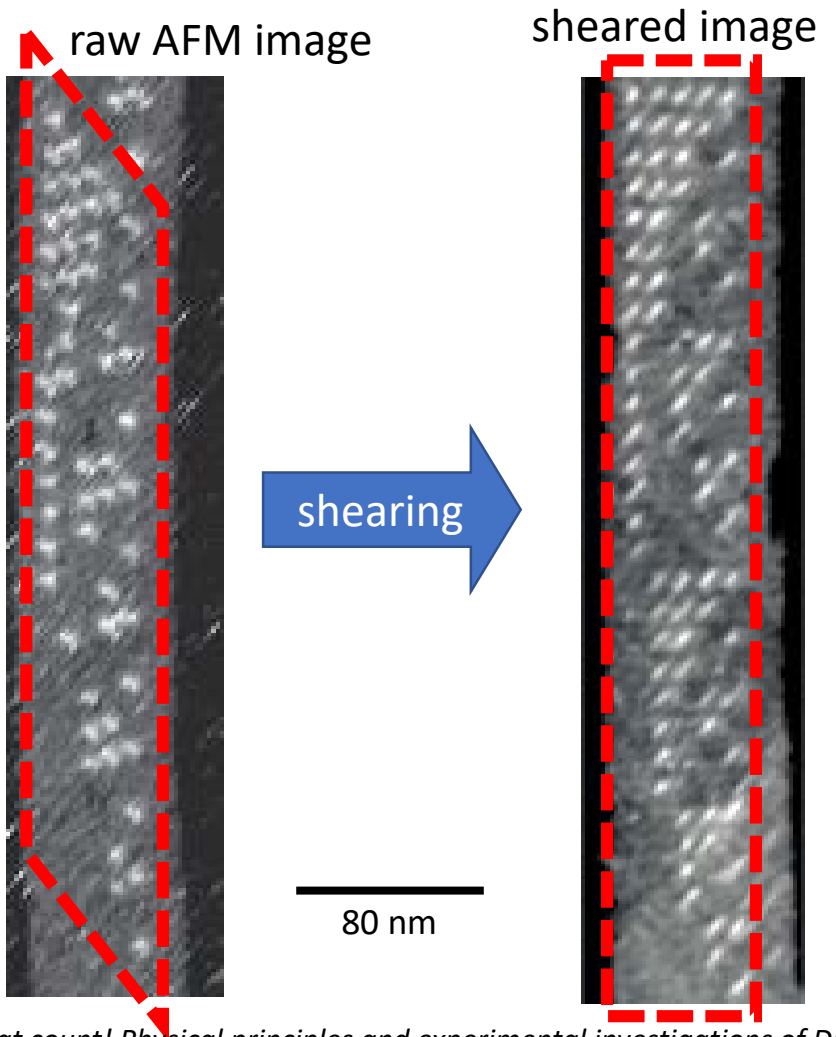



seed
N
W
N
1
N
0
W
W
1
0
ΣHA
1
0
1
0
ΣHA
1
1
0
change function to **half-adder**
0
0
ΣHA
0
0
1
0
ΣHA
1
0
1
1




seed
N
W
change function to **half-adder**

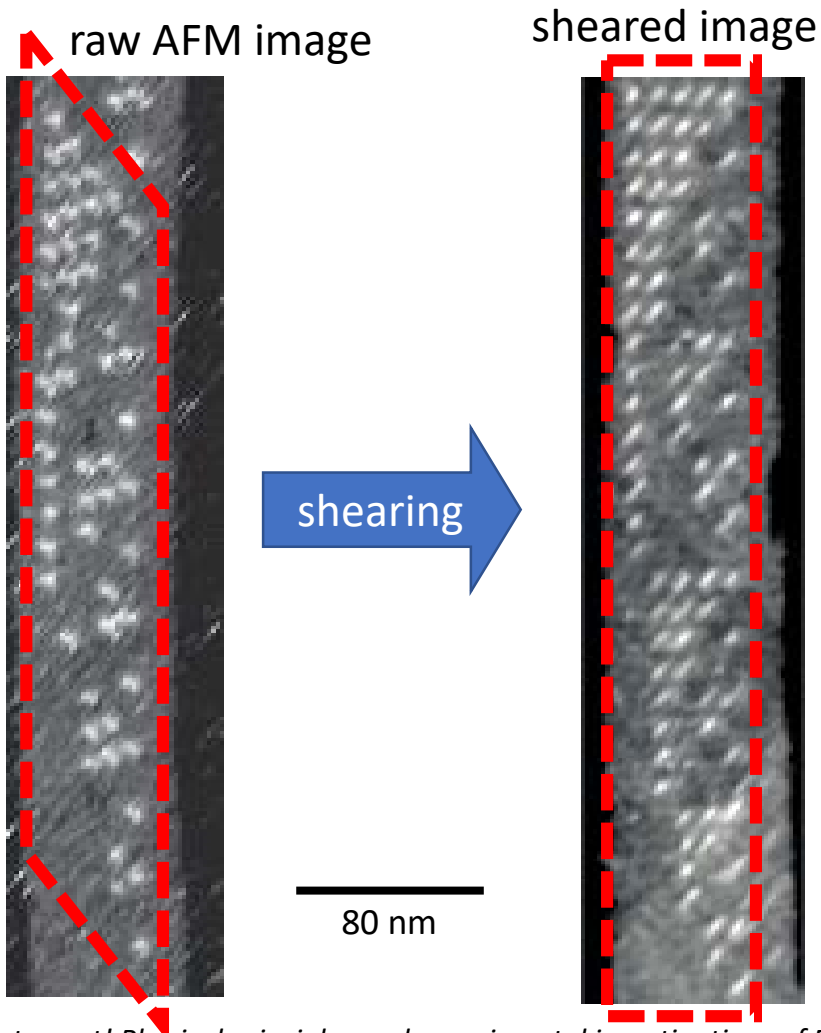
Algorithmic self-assembly in action



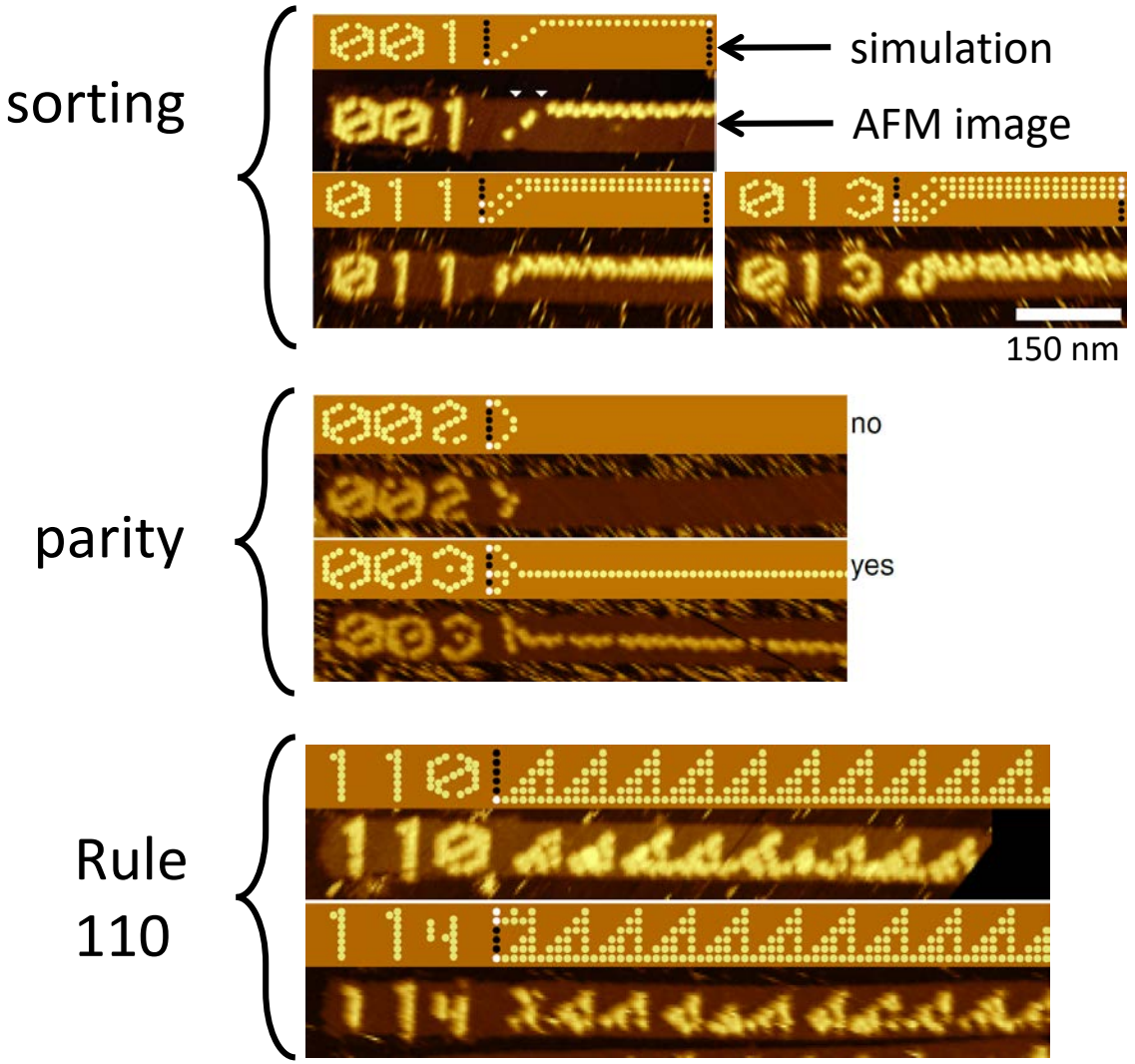
[*Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*, Constantine Evans, Ph.D. thesis, Caltech, 2014]

Algorithmic self-assembly in action

See Damien Woods' talk on Thursday!



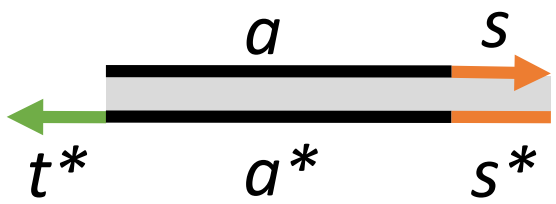
[*Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*, Constantine Evans, Ph.D. thesis, Caltech, 2014]



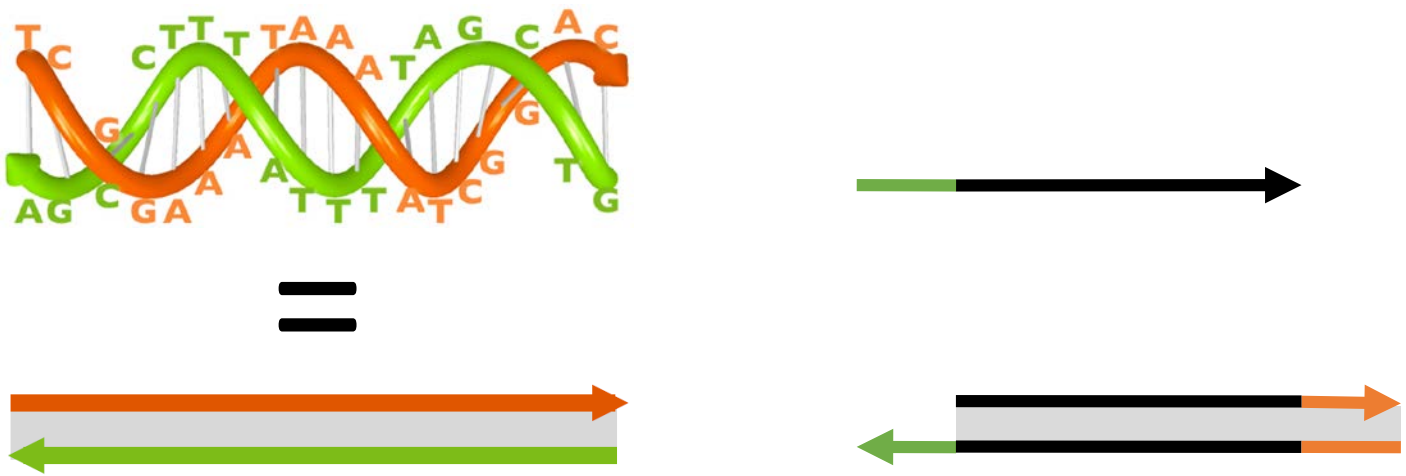
[*Iterated Boolean circuit computation via a programmable DNA tile array*. Woods, Doty, Myhrvold, Hui, Wu, Yin, Winfree, submitted]

DNA strand displacement

=



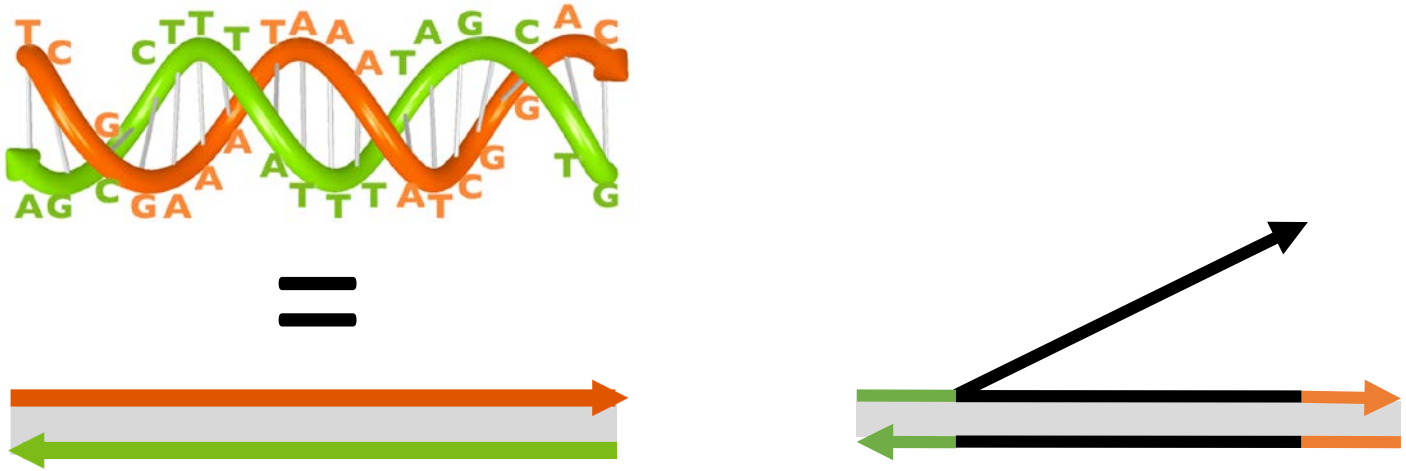
DNA strand displacement



3 basic steps:

- **bind**
- **displace**
- **unbind**

DNA strand displacement

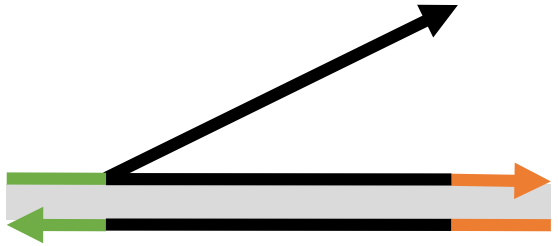


3 basic steps:

- **bind**
- **displace**
- **unbind**

DNA strand displacement

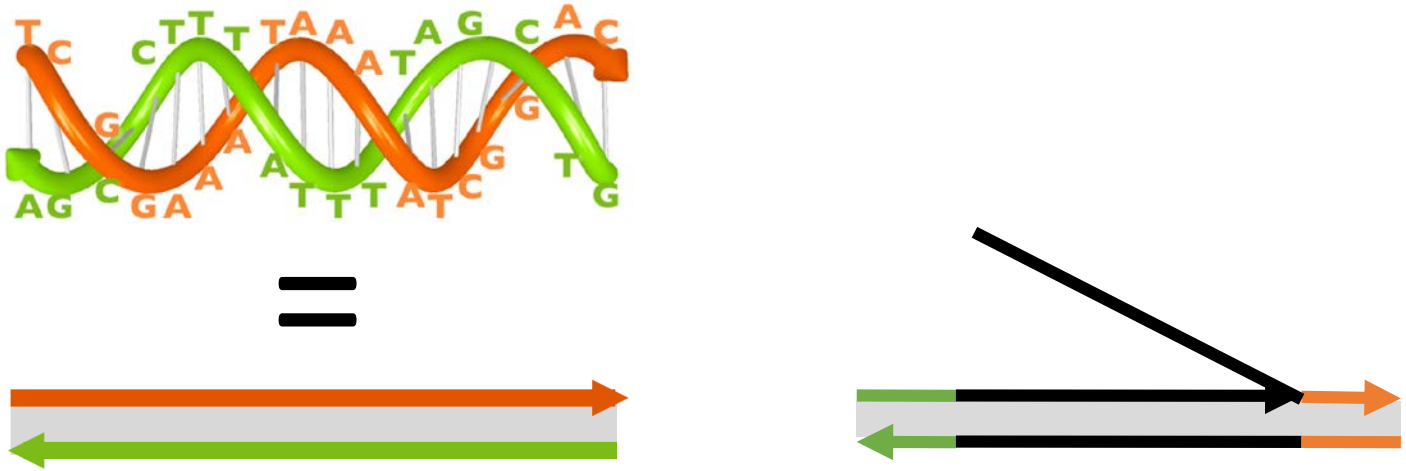
=



3 basic steps:

- **bind**
- **displace**
- **unbind**

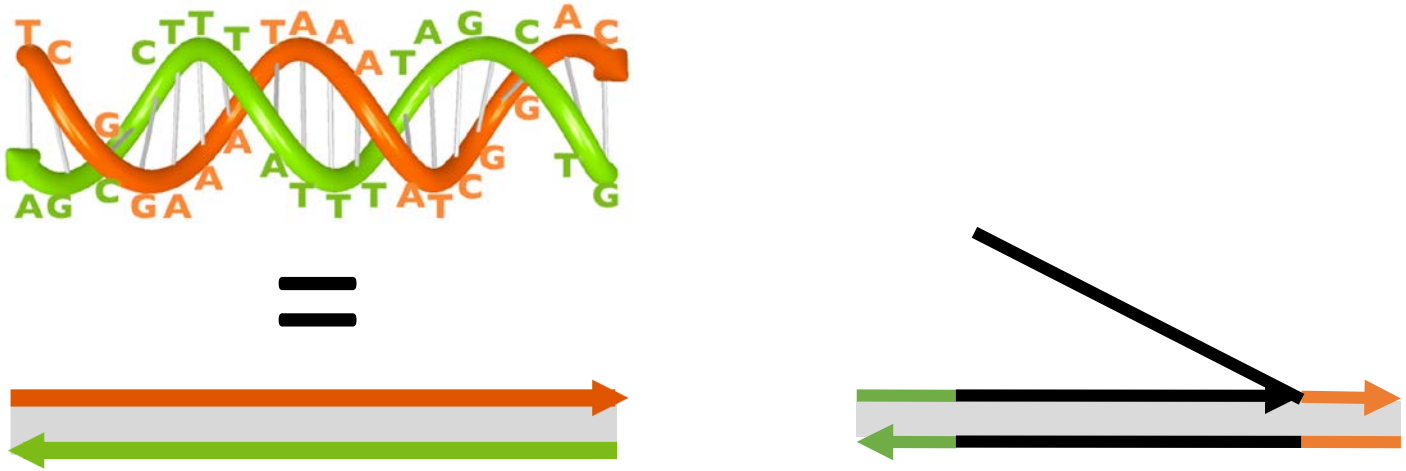
DNA strand displacement



3 basic steps:

- **bind**
- **displace**
- **unbind**

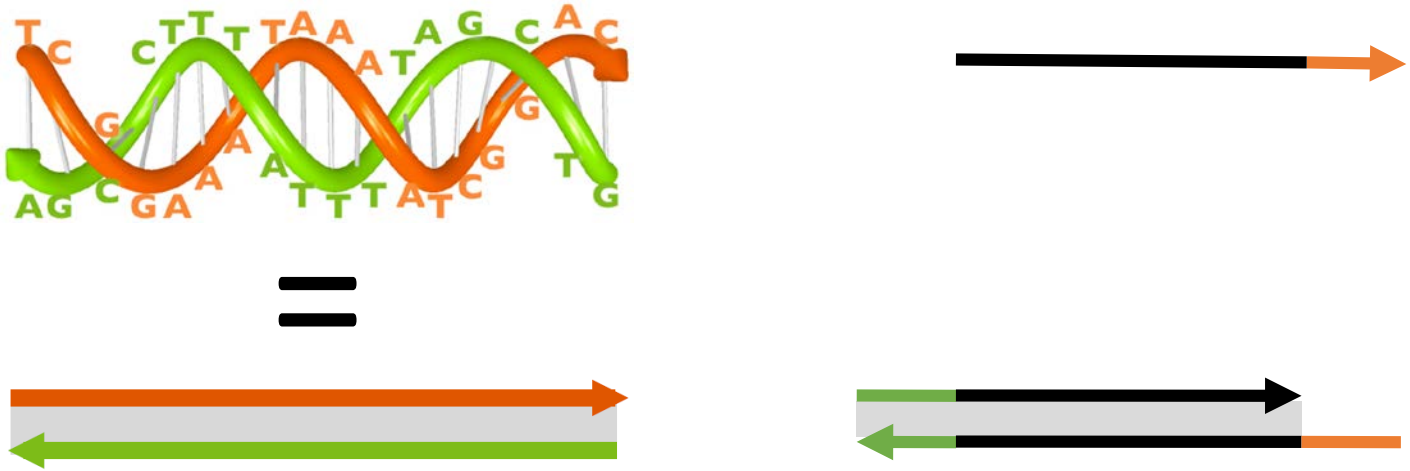
DNA strand displacement



3 basic steps:

- **bind**
- **displace**
- **unbind**

DNA strand displacement



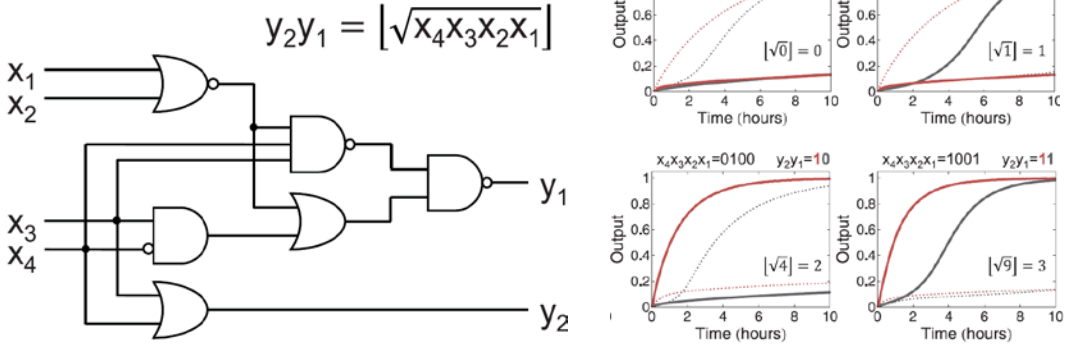
3 basic steps:

- **bind**
- **displace**
- **unbind**

DNA strand displacement implementing C = A AND B

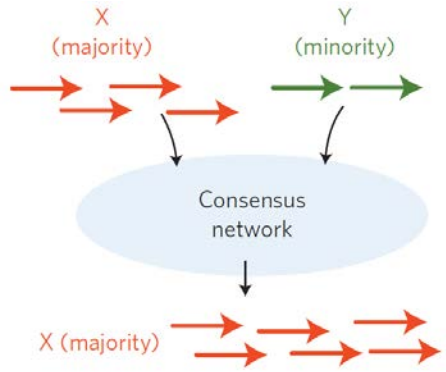
Computation with DNA strand displacement

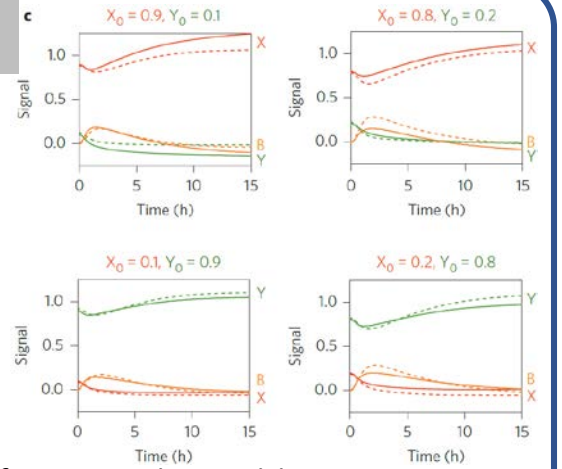
Boolean circuit computation



[*Scaling up digital circuit computation with DNA strand displacement cascades*. Qian and Winfree, **Science** 2011]

Analog majority computation

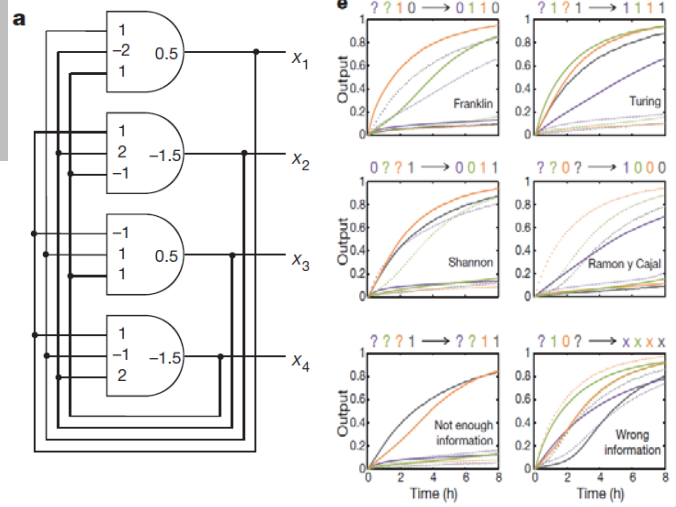




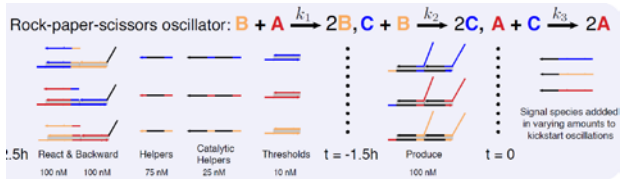
[*Programmable chemical controllers made from DNA*. Chen, Dalchau, Srinivas, Phillips, Cardelli, Soloveichik, Seelig, **Nature Nanotechnology** 2013.]

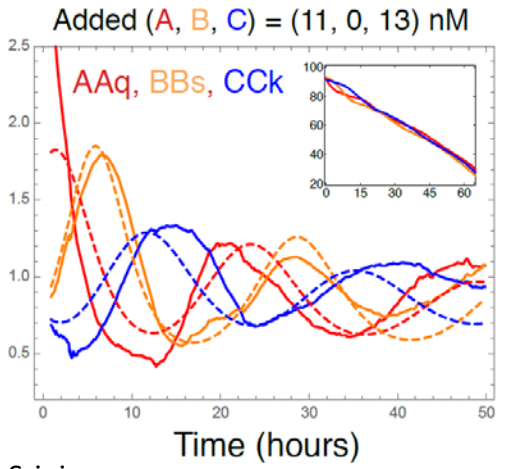
Classification with Hopfield associative networks

[*Neural network computation with DNA strand displacement cascades*. Qian, Winfree, Bruck. **Nature** 2011]



Chemical oscillator





[*Enzyme-free nucleic acid dynamical systems.* Srinivas, Parkin, Seelig, Winfree, Soloveichik, **Science** 2017.]

Outline

Part 1: Computation via molecular binding (kinetically)

Part 2: Computation via molecular binding (thermodynamically)

Part 3: Thermodynamic self-assembly

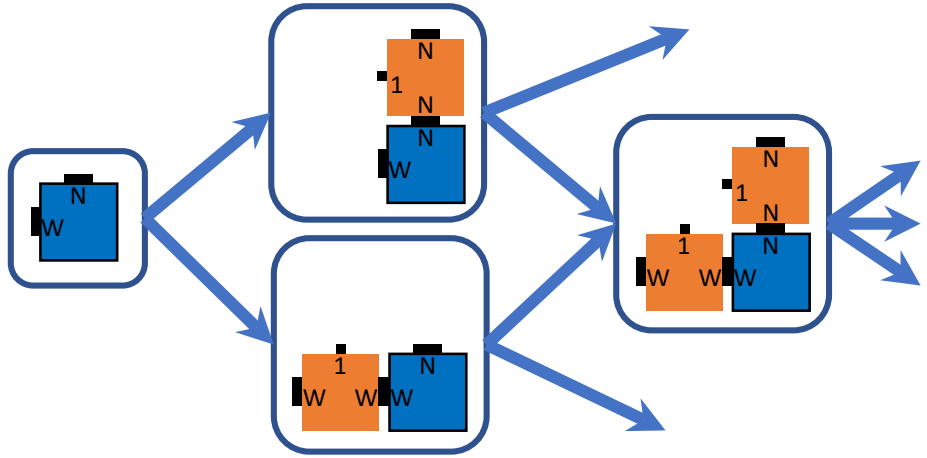
Kinetic models

kinetic ≈ predicts possible/probable paths from initial configuration to others

Kinetic models

kinetic ≈ predicts possible/probable paths from initial configuration to others

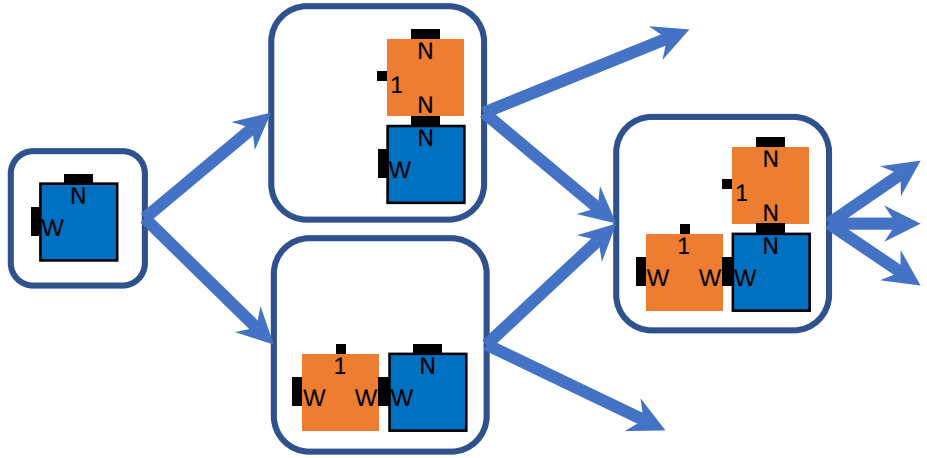
- tile assembly
  - initial configuration = seed
  - other configurations = assemblies containing seed
  - transition = attachment of tile

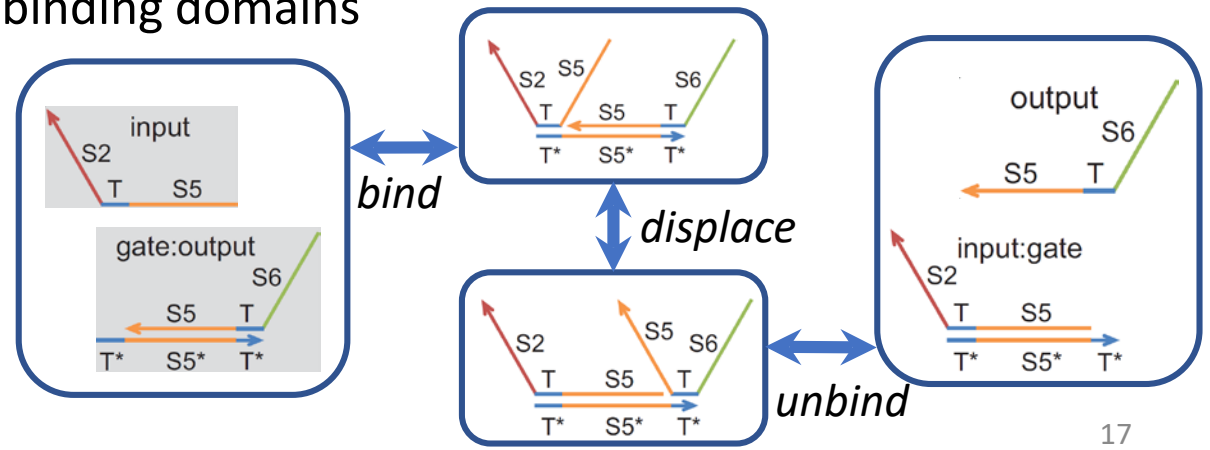


Kinetic models

kinetic ≈ predicts possible/probable paths from initial configuration to others

- tile assembly
  - initial configuration = seed
  - other configurations = assemblies containing seed
  - transition = attachment of tile
- strand displacement
  - initial configuration = initial binding of domains on DNA strands
  - other configurations = other ways of binding domains
  - transition = *bind*|*displace*|*unbind*





Source: *Scaling up digital circuit computation with DNA strand displacement cascades*. Qian and Winfree, Science 2011.

Thermodynamic models

- thermodynamic ≈ defines *free energy* $G(\alpha)$ of each possible configuration α and predicts that Pr[observing α] $\propto e^{-G(\alpha)/kT}$
- minimum free energy configuration = *most probable* configuration

Thermodynamic models

- thermodynamic ≈ defines *free energy* $G(\alpha)$ of each possible configuration α and predicts that Pr[observing α] $\propto e^{-G(\alpha)/kT}$
- minimum free energy configuration = *most probable* configuration
- How to find energy of a configuration of DNA strands?

Thermodynamic models

- thermodynamic ≈ defines *free energy* $G(\alpha)$ of each possible configuration α and predicts that Pr[observing α] $\propto e^{-G(\alpha)/kT}$
- minimum free energy configuration = *most probable* configuration
- How to find energy of a configuration of DNA strands?

  Why, it's elementary:

Appendix A. Pseudocode for the Multistranded Partition Function Algorithm.

Initialize (Q, Q^b, Q^m) // $\mathcal{O}(N^2)$ space
Set all values to 0 except $Q_{i,i-1} = 1$ for $i = 1, \ldots, N$
for $l = 1, N$
  for $i = 1, N-l+1$
    $j = i+l-1$
    // Q^b **recursion equations**
    if $\eta[i+\frac{1}{2}, j-\frac{1}{2}] == 0$
      $Q^b_{i,j} = \exp\{-\Delta G^{\text{hairpin}}_{i,j}/kT\}$
    for $d = i+1, j-2$ // loop over all possible 3′-most pairs $d{\cdot}e$
      for $e = d+1, j-1$
        if $\eta[i+\frac{1}{2}, d-\frac{1}{2}] == 0$ **and** $\eta[e+\frac{1}{2}, j-\frac{1}{2}] == 0$
          $Q^b_{i,j}$ += $\exp\{-\Delta G^{\text{interior}}_{i,d,e,j}/kT\}\, Q^b_{d,e}$
        if $\eta[e+\frac{1}{2}, j-\frac{1}{2}] == 0$ **and** $\eta[i+\frac{1}{2}] == 0$ **and** $\eta[d-\frac{1}{2}] == 0$ // multiloop: no top-level nicks
          $Q^b_{i,j}$ += $Q^m_{i+1,d-1}\, Q^b_{d,e} \exp\{-[\Delta G^{\text{multi}}_{\text{init}} + 2\Delta G^{\text{multi}}_{\text{bp}} + (j-e-1)\Delta G^{\text{multi}}_{\text{base}}]/kT\}$
    for $c \in \{i, \ldots, j-1\}$ s.t. $\eta[c+\frac{1}{2}] == 1$ // loop over all top-level nicks $\in [i+\frac{1}{2}, j-\frac{1}{2}]$
      if $(\eta[i+\frac{1}{2}] == 0$ **and** $\eta[j-\frac{1}{2}] == 0)$ **or** $(i == j-1)$ **or**
        $(c == i$ **and** $\eta[j-\frac{1}{2}] == 0)$ **or** $(c == j-1$ **and** $\eta[i+\frac{1}{2}] == 0)$ **then**
        $Q^b_{i,j}$ += $Q_{i+1,c}\, Q_{c+1,j-1}$ // exterior loops

// Q, Q^m **recursion equations**
if $\eta[i+\frac{1}{2}, j-\frac{1}{2}] == 0$ **then** $Q_{i,j} = 1$ // empty substructure
  else $Q_{i,j} = 0$ // unconnected substructure
for $d = i, j-1$ // loop over all possible 3′-most pairs $d{\cdot}e$
  for $e = d+1, j$
    if $\eta[e+\frac{1}{2}, j-\frac{1}{2}] == 0$
      if $\eta[d-\frac{1}{2}] == 0$ **or** $d == i$
        $Q_{i,j}$ += $Q_{i,d-1}\, Q^b_{d,e}$
      if $\eta[i+\frac{1}{2}, d-\frac{1}{2}] == 0$
        $Q^m_{i,j}$ += $\exp\{-[\Delta G^{\text{multi}}_{\text{bp}} + (d-i)\Delta G^{\text{multi}}_{\text{base}} + (j-e)\Delta G^{\text{multi}}_{\text{base}}]/kT\}\, Q^b_{d,e}$ // single pair in Q^m
      if $\eta[d-\frac{1}{2}] == 0$
        $Q^m_{i,j}$ += $Q^m_{i,d-1}\, Q^b_{d,e} \exp\{-[\Delta G^{\text{multi}}_{\text{bp}} + (j-e)\Delta G^{\text{multi}}_{\text{base}}]/kT\}$ // more than one pair in Q^m
return $[Q_{1,N} \exp\{-(L-1)\Delta G^{\text{assoc}}/kT\}]$ // partition function $\overline{Q}(\pi)$ for ordering π

Source: *Thermodynamic Analysis of Interacting Nucleic Acid Strands.*
Dirks, Bois, Schaeffer, Winfree, Pierce, SIAM Review 2011.

Simplified model: Thermodynamic binding networks

Simplified model: Thermodynamic binding networks

Goal 1: Isolate the contribution of thermodynamics alone to the correctness of a molecular binding system

- no notion of “initial configuration” or “transitions”

Simplified model: Thermodynamic binding networks

Goal 1: Isolate the contribution of thermodynamics alone to the correctness of a molecular binding system

- no notion of “initial configuration” or “transitions”

Goal 2: Abstract away DNA-specific details and model arbitrary molecules with specific binding domains

- no “geometry”
- can “graft onto” kinetic models

Simplified model: Thermodynamic binding networks

Not a goal: be a predictive model telling us what the system *will* do

- It's an adversarial model *constraining the system's misbehavior* when it violates assumptions of a more detailed molecular model.

Simplified model: Thermodynamic binding networks

Not a goal: be a predictive model telling us what the system *will* do

- It's an adversarial model *constraining the system's misbehavior* when it violates assumptions of a more detailed molecular model.

Errors in practice take "illegal" transitions in kinetic model

- tile binds with strength 1 instead of 2
- strand A displaces B despite lack of toehold

Simplified model: Thermodynamic binding networks

Not a goal: be a predictive model telling us what the system *will* do

- It's an adversarial model *constraining the system's misbehavior* when it violates assumptions of a more detailed molecular model.

Errors in practice take "illegal" transitions in kinetic model

- tile binds with strength 1 instead of 2
- strand A displaces B despite lack of toehold

Can we design systems so that the desired final configuration is also the thermodynamically most stable?

Thermodynamic binding networks: Definition

monomer type = collection of domains

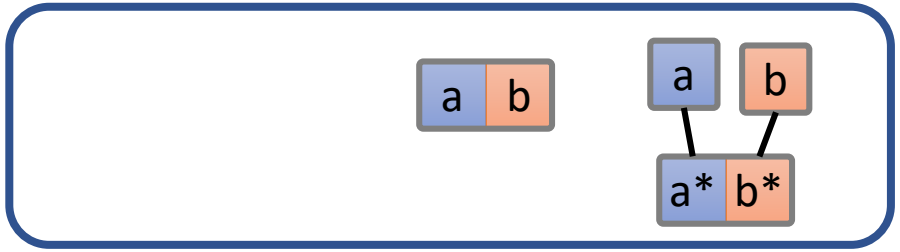
e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

Thermodynamic binding networks: Definition

monomer type = collection of domains

e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

configuration = matching of complementary domains (how domains are bound)



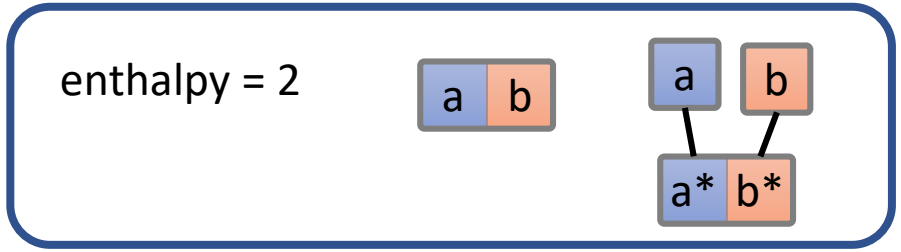
Thermodynamic binding networks: Definition

monomer type = collection of domains

e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

configuration = matching of complementary domains (how domains are bound)

enthalpy of configuration = # bonds



Thermodynamic binding networks: Definition

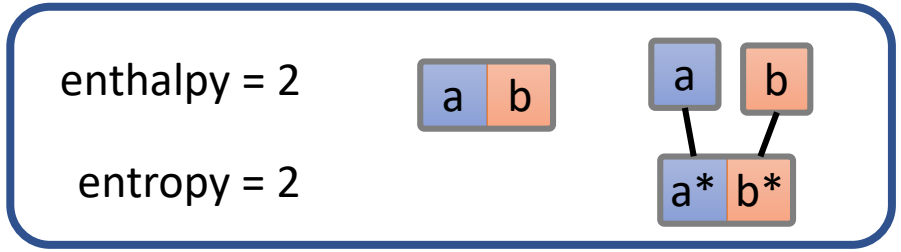
monomer type = collection of domains

e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

configuration = matching of complementary domains (how domains are bound)

enthalpy of configuration = # bonds

entropy of configuration = # free complexes (*a.k.a. polymers*)

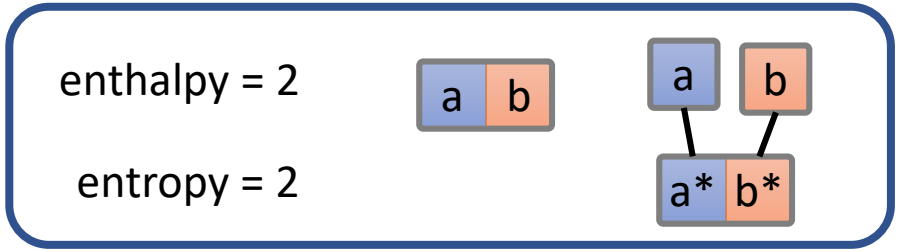


Thermodynamic binding networks: Definition

monomer type = collection of domains

e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

configuration = matching of complementary domains (how domains are bound)



enthalpy of configuration = # bonds

entropy of configuration = # free complexes (*a.k.a. polymers*)

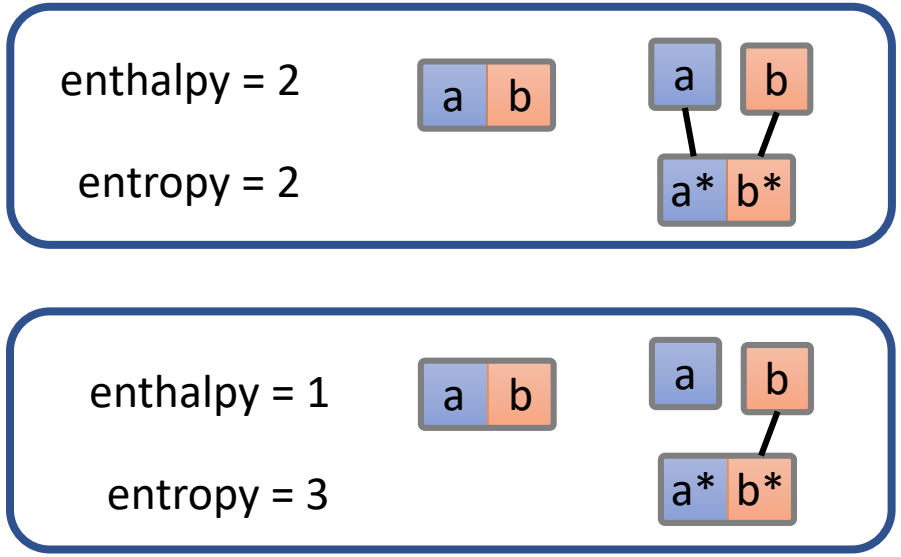
energy of configuration = weighted sum of enthalpy and entropy

Thermodynamic binding networks: Definition

monomer type = collection of domains

e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

configuration = matching of complementary domains (how domains are bound)



enthalpy of configuration = # bonds

entropy of configuration = # free complexes (*a.k.a. polymers*)

energy of configuration = weighted sum of enthalpy and entropy

Thermodynamic binding networks: Definition

monomer type = collection of domains

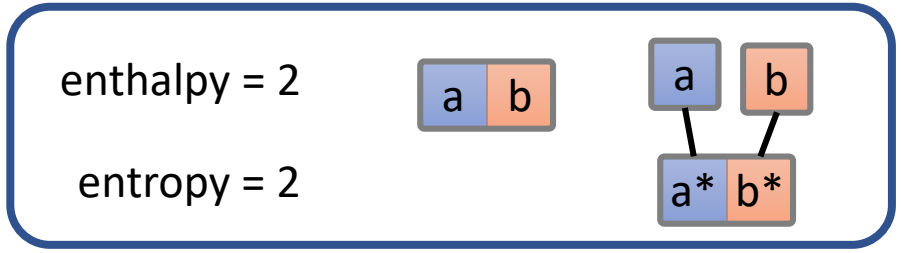
e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

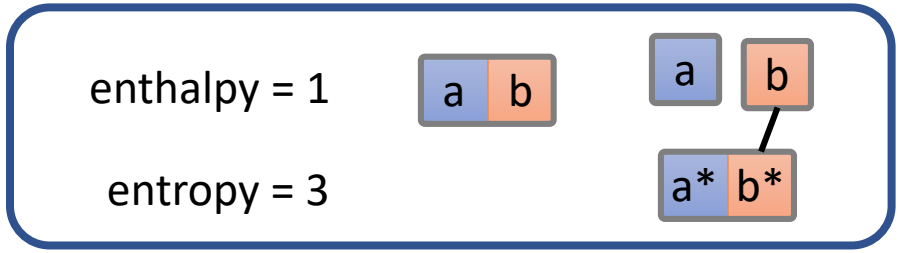
configuration = matching of complementary domains (how domains are bound)

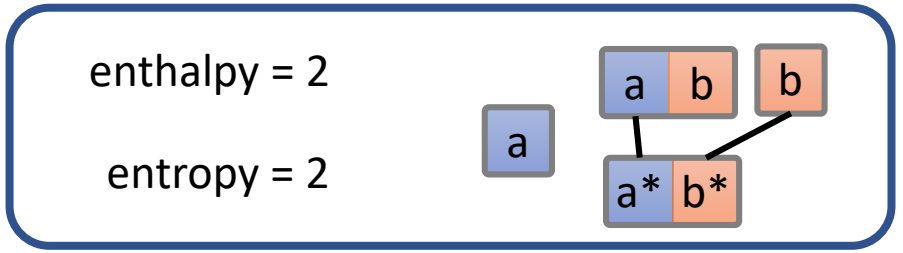
enthalpy of configuration = # bonds

entropy of configuration = # free complexes (*a.k.a. polymers*)

energy of configuration = weighted sum of enthalpy and entropy







Thermodynamic binding networks: Definition

monomer type = collection of domains

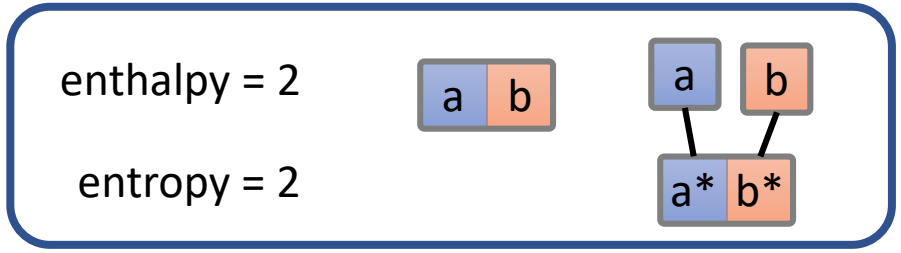
e.g., { *a*, *b*, *b*, *c*, *d**} or {*d*, *a**}

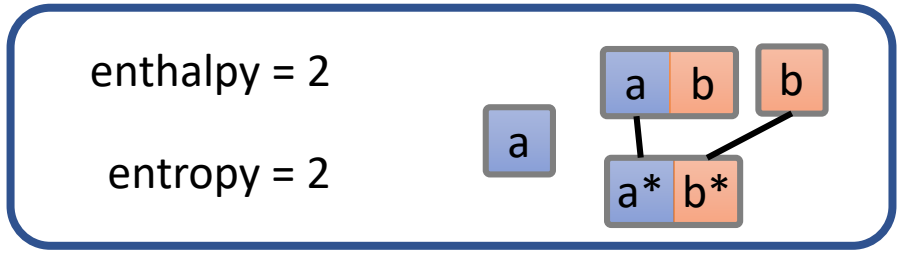
configuration = matching of complementary domains (how domains are bound)

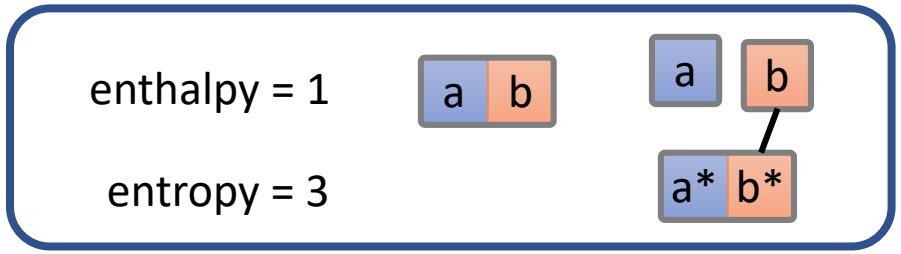
enthalpy of configuration = # bonds

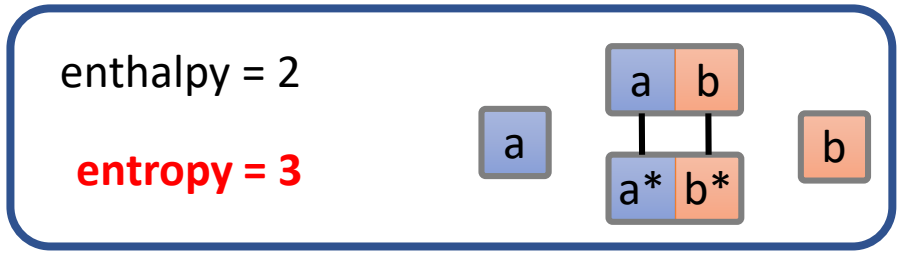
entropy of configuration = # free complexes (*a.k.a. polymers*)

energy of configuration = weighted sum of enthalpy and entropy









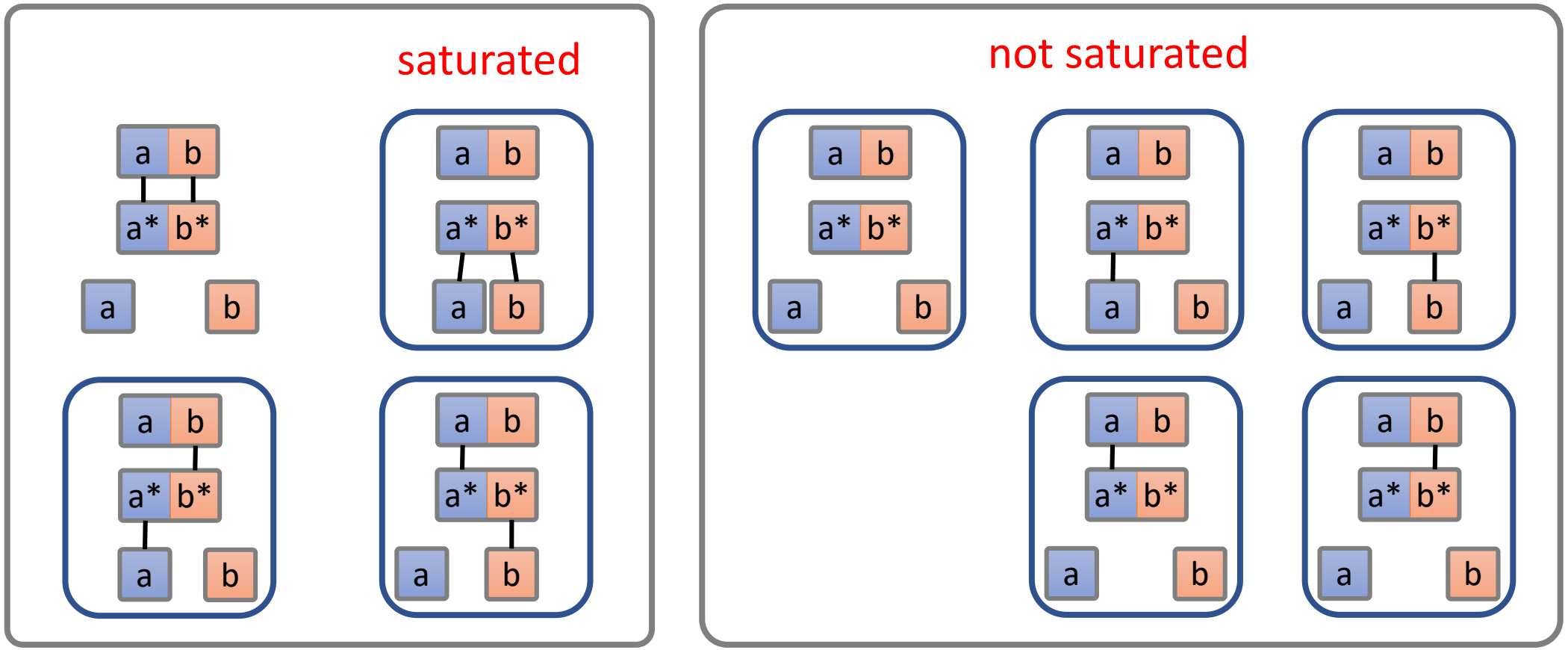
Tradeoff favoring enthalpy *infinitely* over entropy

monomer collection = vector $\mathbf{c} \in \mathbb{N}^M$ indicating counts of M monomer types

Tradeoff favoring enthalpy *infinitely* over entropy

monomer collection = vector $\mathbf{c} \in \mathbb{N}^M$ indicating counts of M monomer types

configuration of **c** is saturated if enthalpy is maximal (*no pair of unbound complementary domains*)

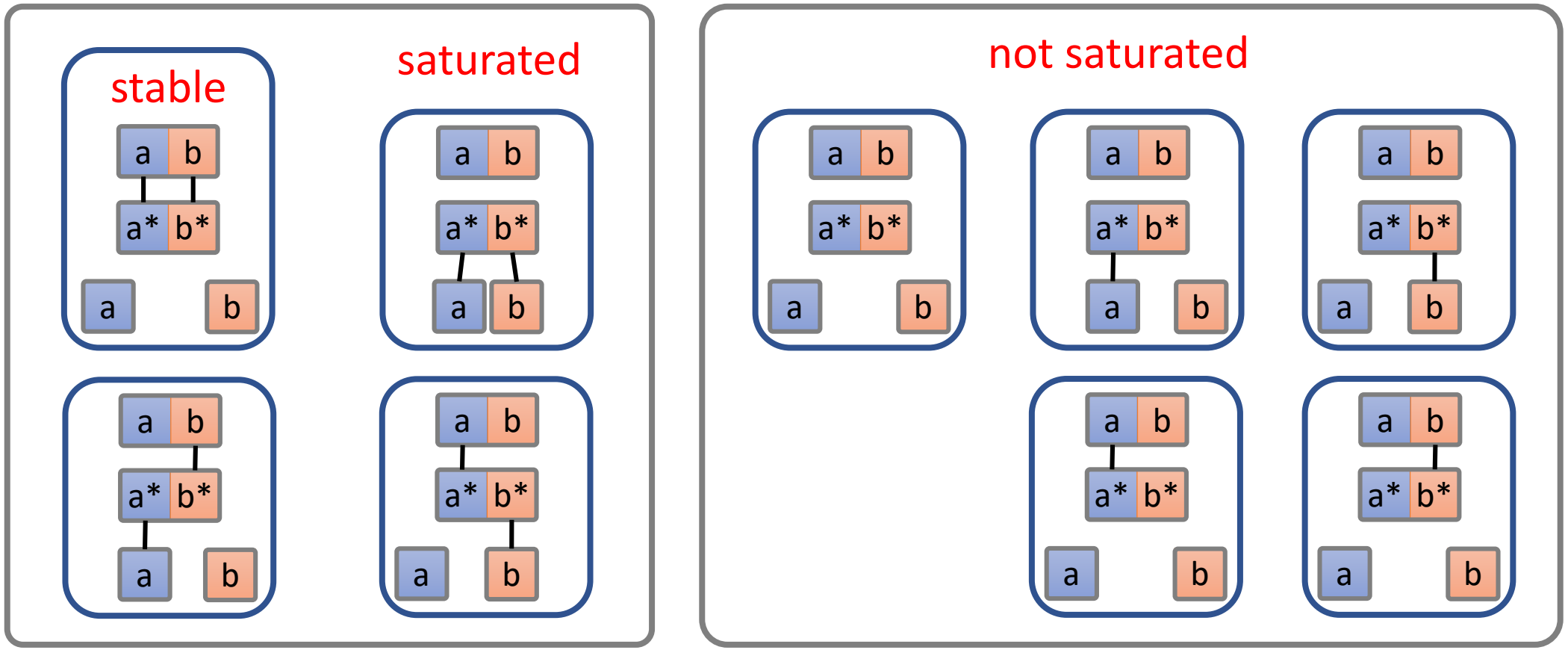


Tradeoff favoring enthalpy *infinitely* over entropy

monomer collection = vector $\mathbf{c} \in \mathbb{N}^M$ indicating counts of M monomer types

configuration of **c** is saturated if enthalpy is maximal (*no pair of unbound complementary domains*)

configuration of **c** is stable if saturated and entropy is maximal (*no saturated config. of* **c** *has more polymers*)



Outline

Part 1: Computation via molecular binding (kinetically)

Part 2: Computation via molecular binding (thermodynamically)

Part 3: Thermodynamic self-assembly

A modest goal

- **Informal**: Design monomers that self-assemble arbitrarily large polymers.
  - size of a polymer = # monomers in the polymer

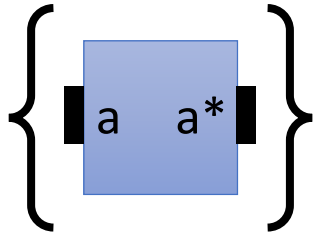
A modest goal

- **Informal**: Design monomers that self-assemble arbitrarily large polymers.
  - size of a polymer = # monomers in the polymer
- **Formal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.

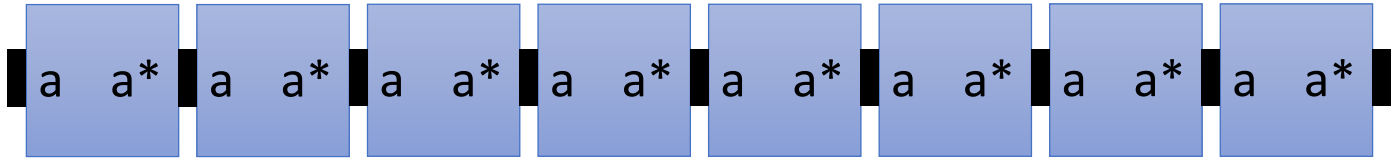
A modest goal

- **Informal**: Design monomers that self-assemble arbitrarily large polymers.
  - size of a polymer = # monomers in the polymer
- **Formal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.
- Easy to do in Abstract Tile Assembly Model:

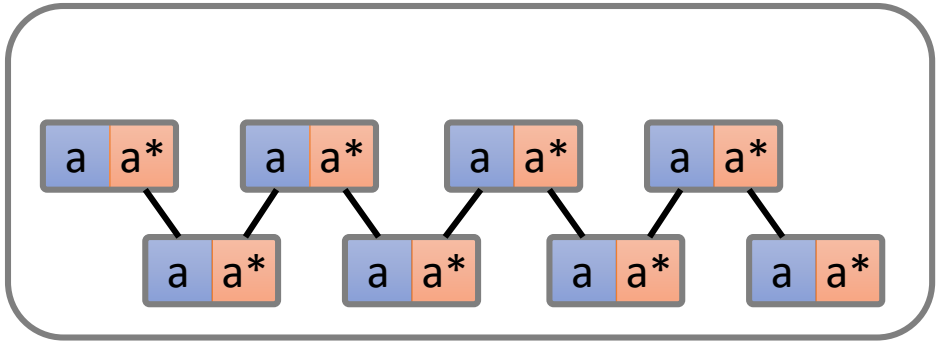
set of monomer types:



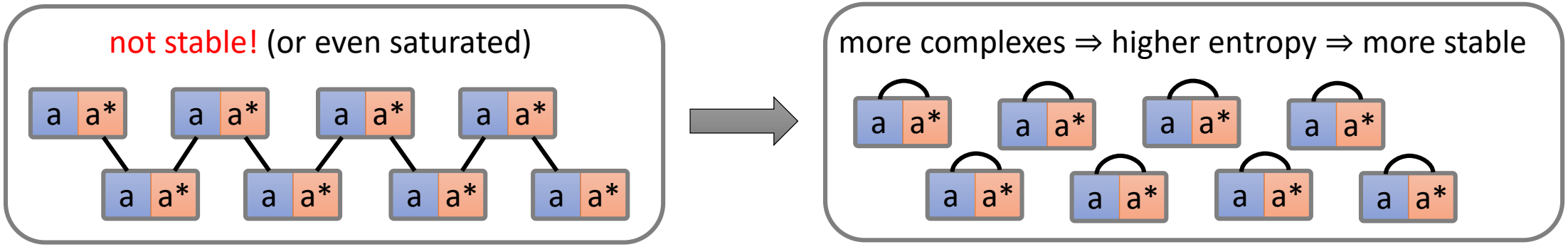
size-8 polymer (assembly) formed with 8 copies of monomer



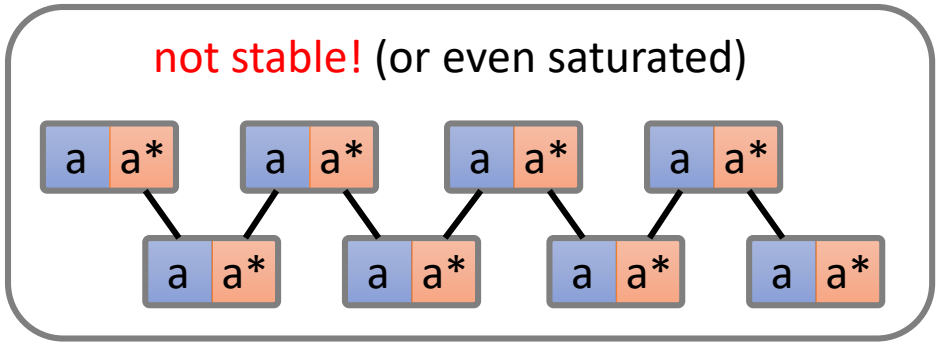
Difficulty of self-assembling large polymers

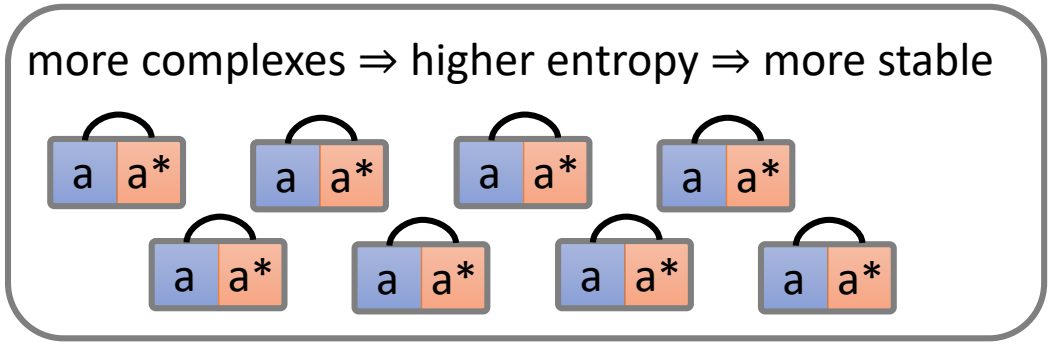


Difficulty of self-assembling large polymers

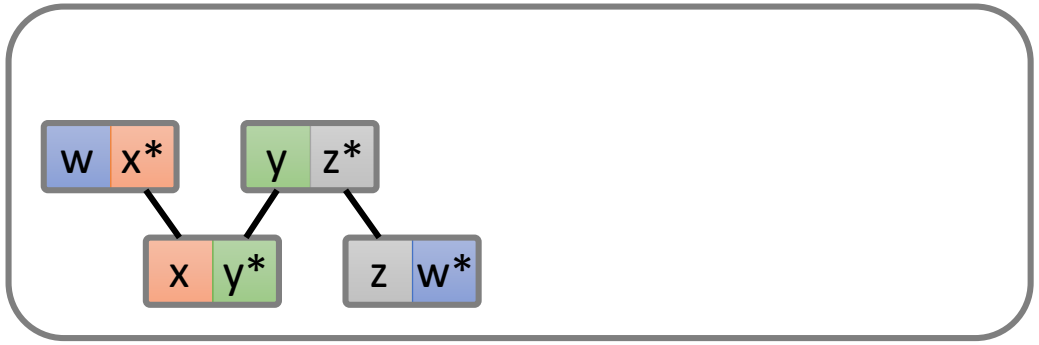


Difficulty of self-assembling large polymers

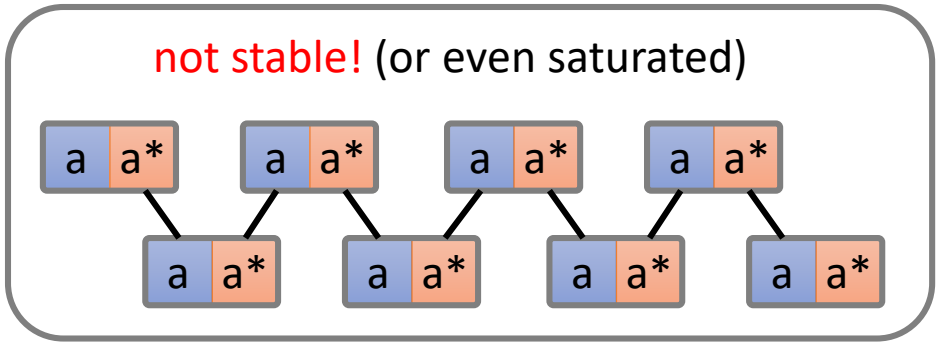


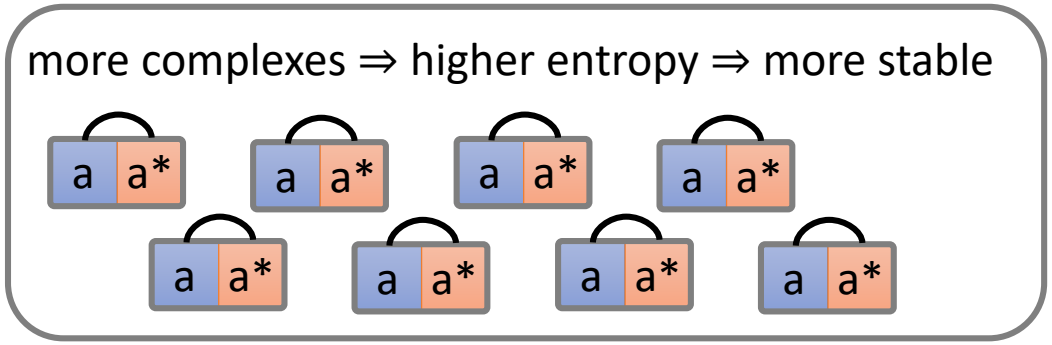


attempt 2:

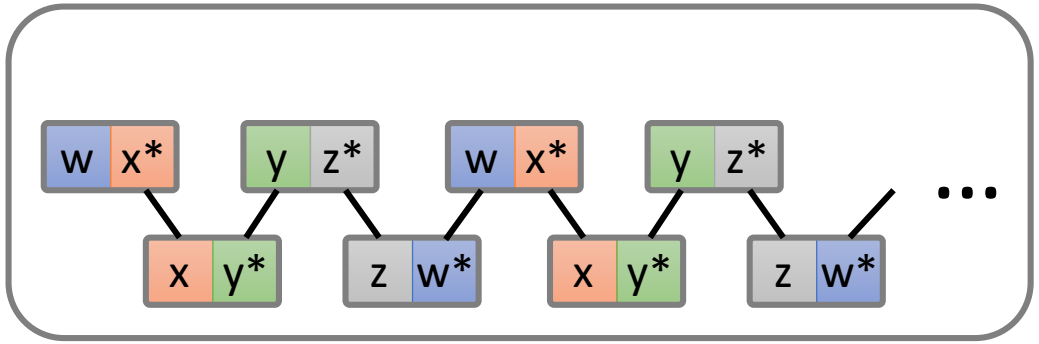


Difficulty of self-assembling large polymers

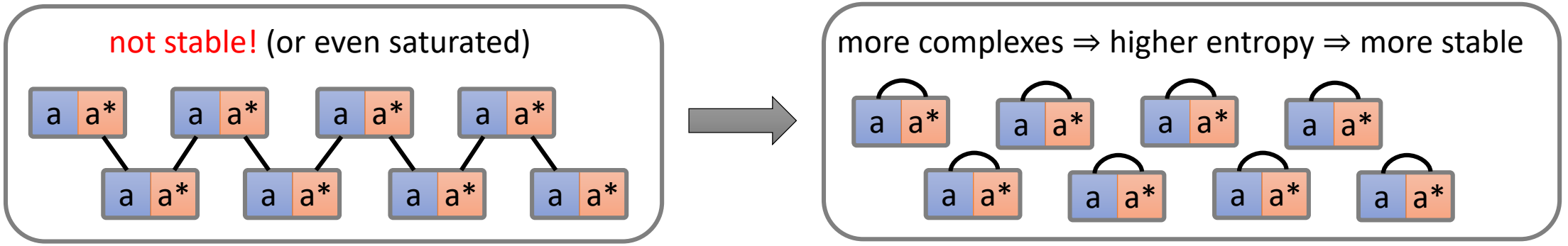




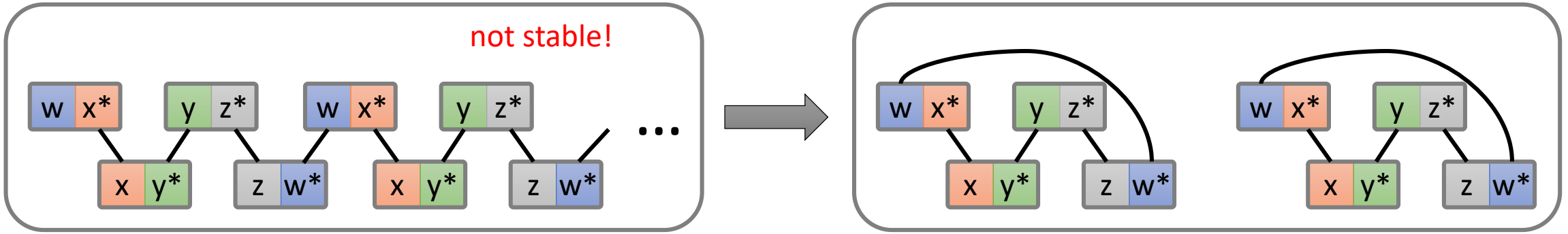
attempt 2:



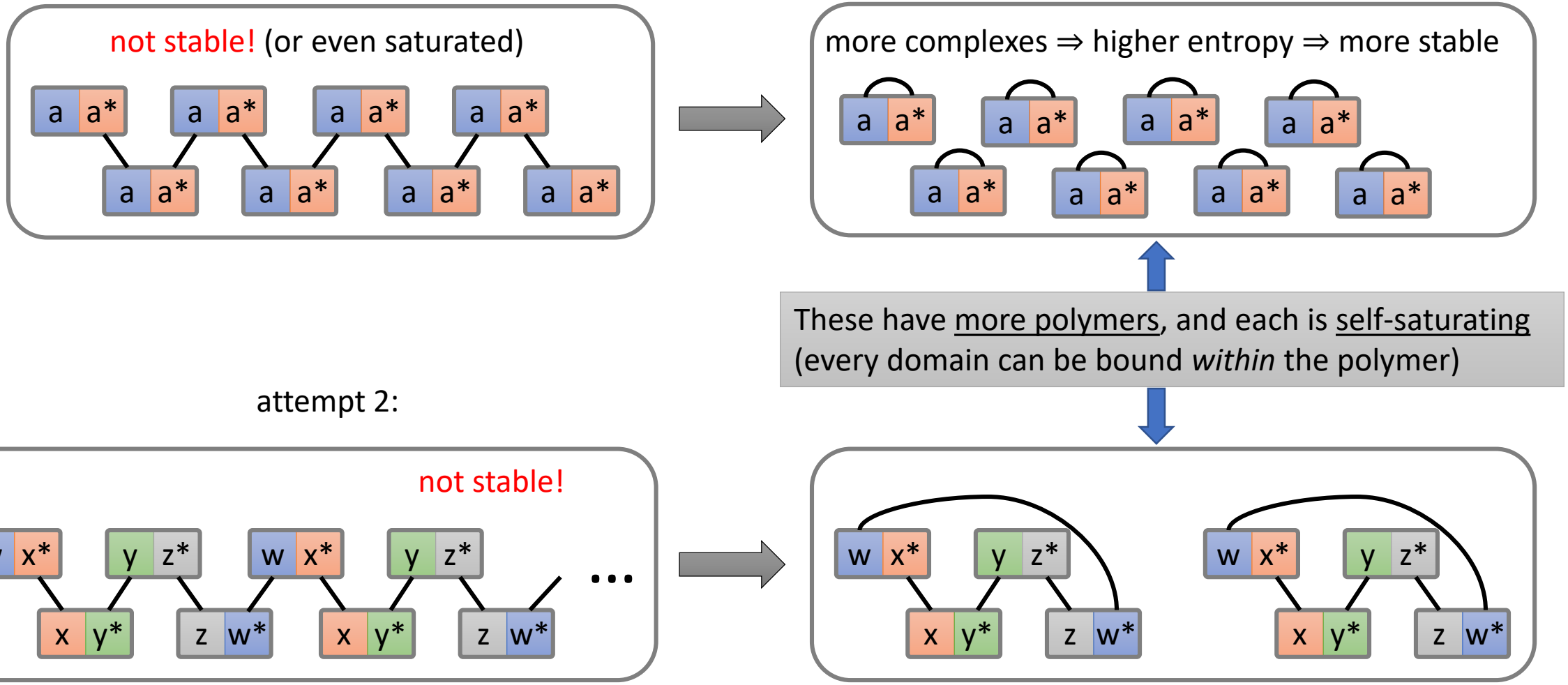
Difficulty of self-assembling large polymers



attempt 2:



Difficulty of self-assembling large polymers



An even more modest goal

Original goal: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.

An even more modest goal

~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

Revised goal: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

An even more modest goal

~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

Revised goal: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

How large can we make S relative to D and M?

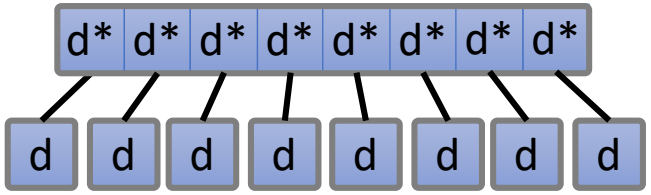
An even more modest goal

~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

Revised goal: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

How large can we make S relative to D and M?

$D, M = O(1)$, S = arbitrarily large



An even more modest goal

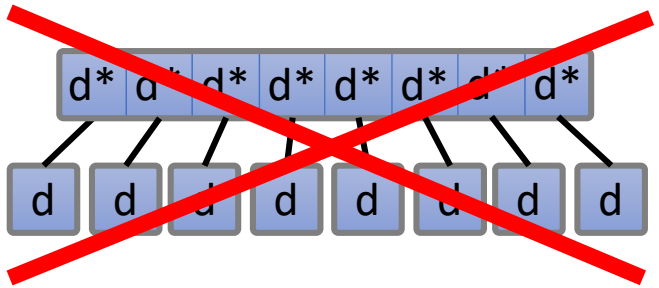
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

and $O(1)$ domains per monomer

Re-**Revised goal**: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

How large can we make S relative to D and M?

$D, M = O(1)$, S = arbitrarily large



An even more modest goal

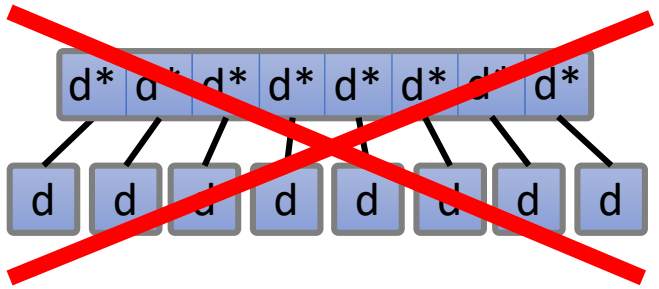
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

and $O(1)$ domains per monomer

Re-**Revised goal**: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

How large can we make S relative to D and M?

$D, M = O(1)$, S = arbitrarily large



$S \approx D$

An even more modest goal

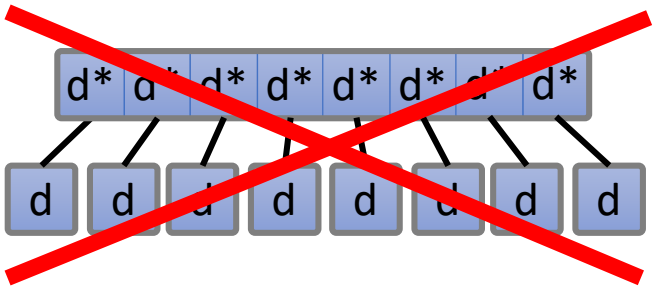
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

and $O(1)$ domains per monomer

Re-**Revised goal**: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

How large can we make S relative to D and M?

$D, M = O(1)$, S = arbitrarily large



$S \approx D$

An even more modest goal

~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

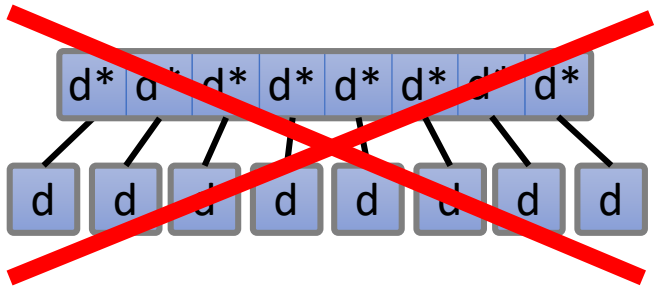
and $O(1)$ domains per monomer

Re-**Revised goal**: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

How large can we make S relative to D and M?

$S \approx D^2$

$D, M = O(1)$, S = arbitrarily large



$S \approx D$

An even more modest goal

~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

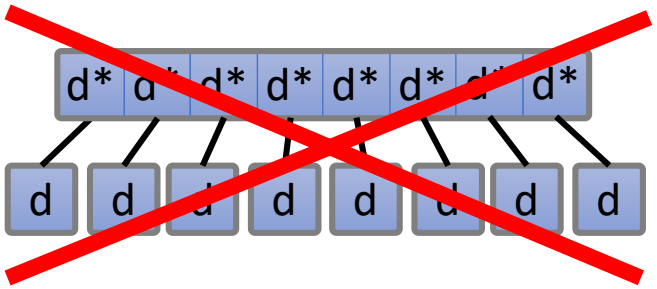
and $O(1)$ domains per monomer

Re-**Revised goal**: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

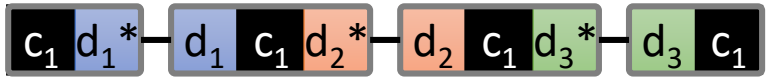
How large can we make S relative to D and M?

$S \approx D^2$

$D, M = O(1)$, S = arbitrarily large



$S \approx D$



An even more modest goal

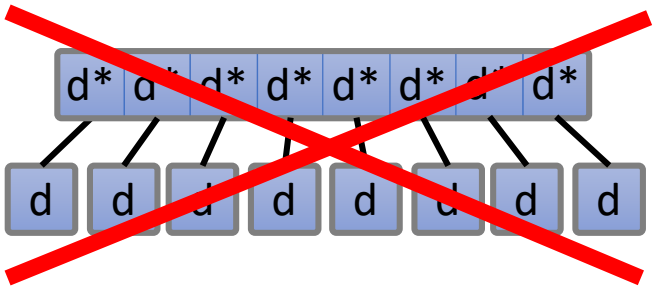
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable polymer of size at least S.~~

and $O(1)$ domains per monomer

Re-**Revised goal**: For all $S \in \mathbb{N}$, design a set of M monomer types using D domain types with a stable polymer of size at least S.

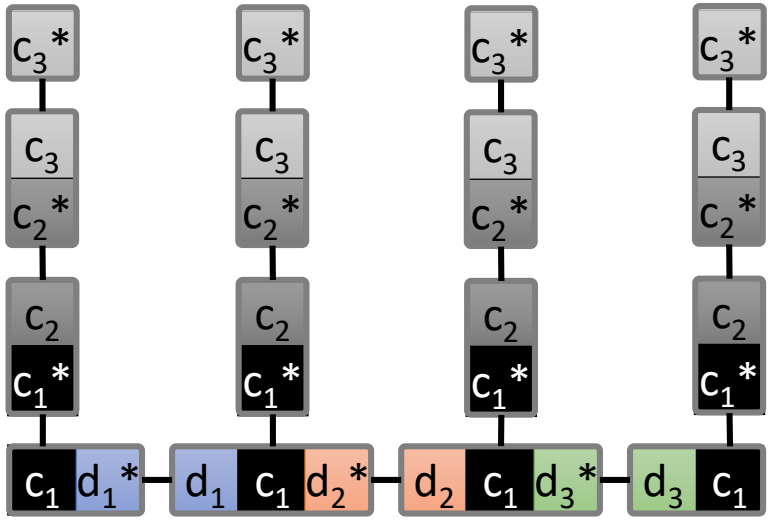
How large can we make S relative to D and M?

$D, M = O(1)$, S = arbitrarily large



$S \approx D$

$S \approx D^2$

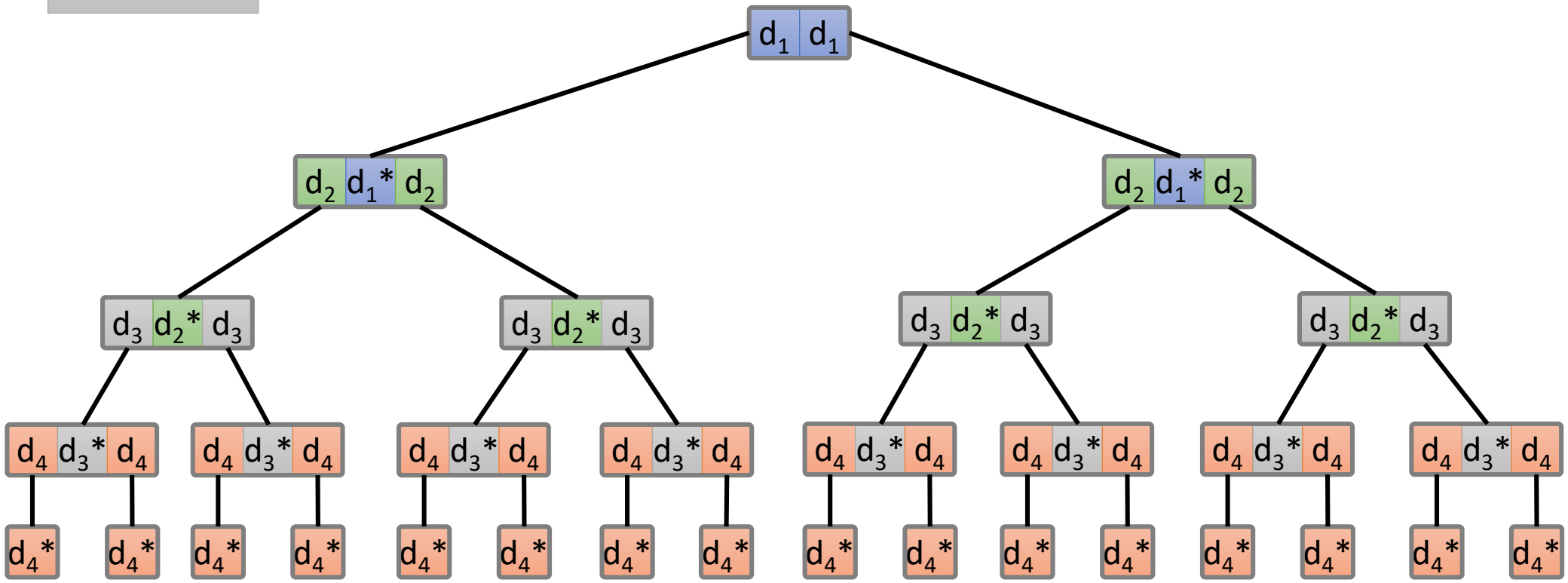


How large can we make *S* relative to *D* and *M*?

$S \approx 2^D$?

How large can we make *S* relative to *D* and *M*?

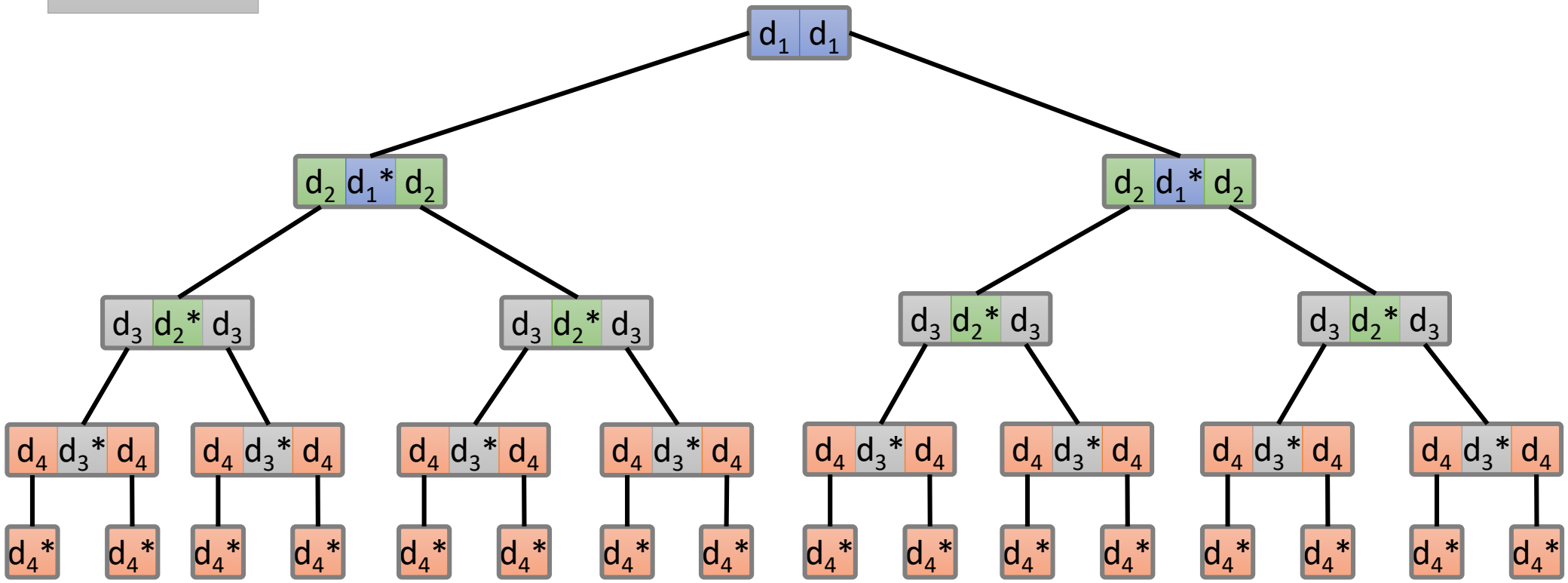
$S \approx 2^D$?



How large can we make *S* relative to *D* and *M*?

$S \approx 2^D$?

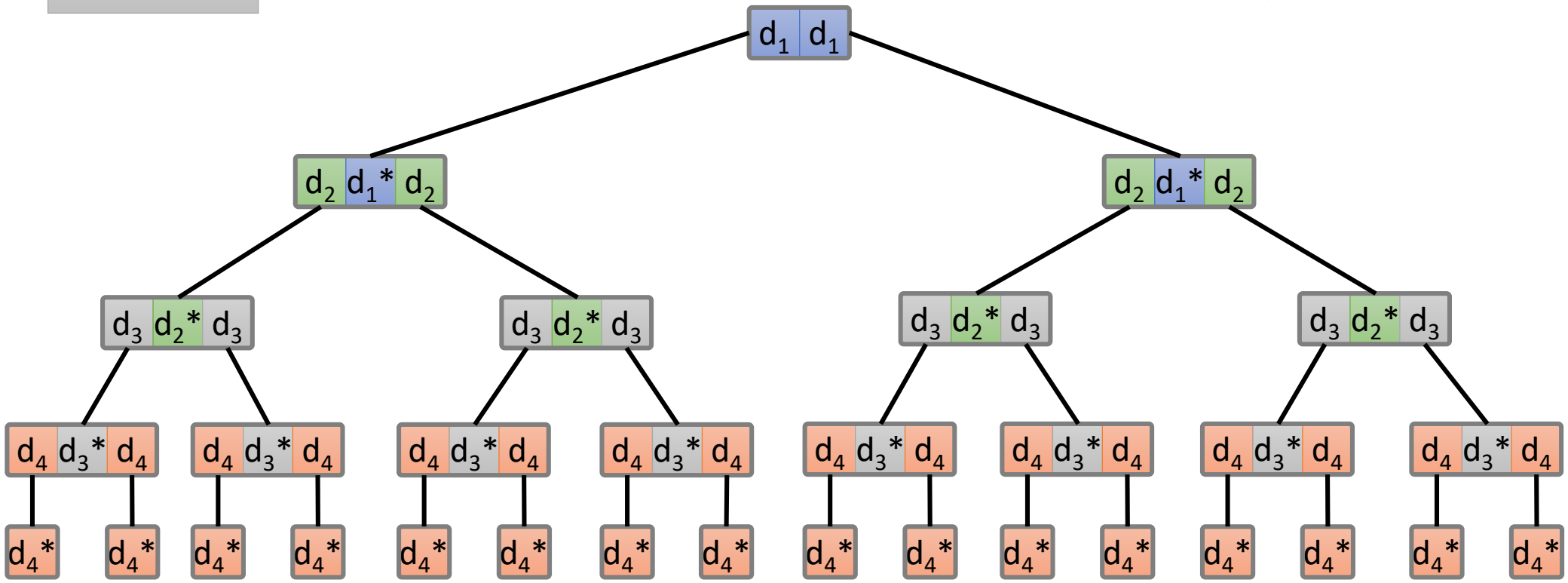
$S \approx 2^{2^D}$??



How large can we make *S* relative to *D* and *M*?

$S \approx 2^D$?

~~$S \approx 2^{2^D}$??~~



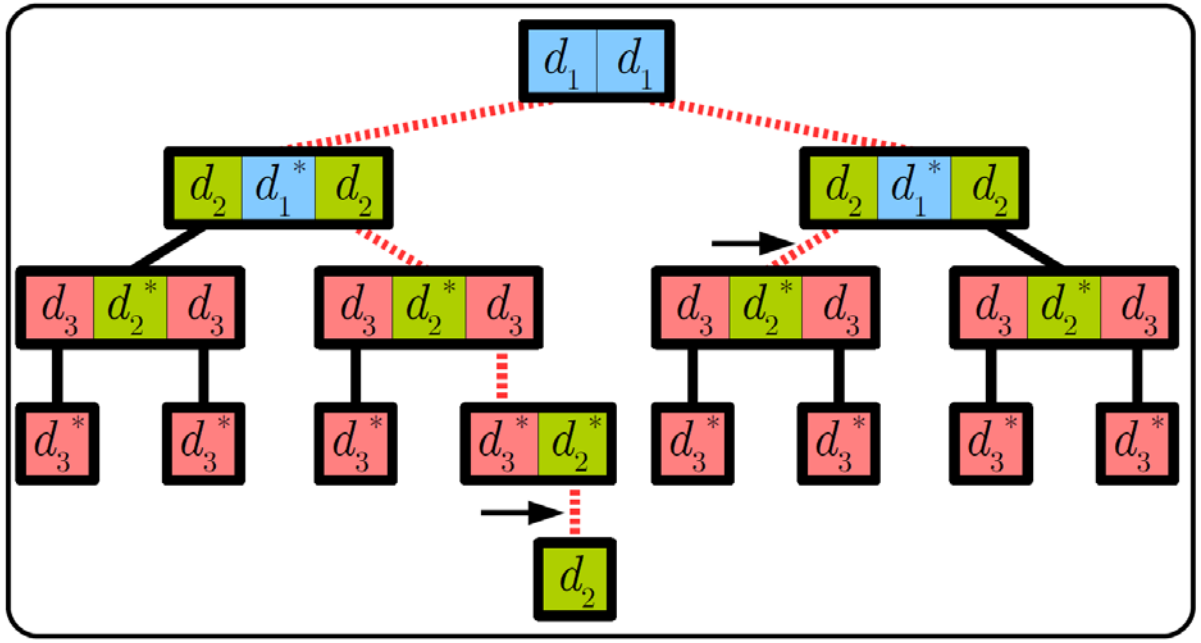
Stable polymers have at most exponential size

Theorem: Any thermodynamic binding network with

- D domain types,
- M monomer types,
- $\leq A$ domains per monomer type (note $D/A \leq M \leq A^{D+1}$)

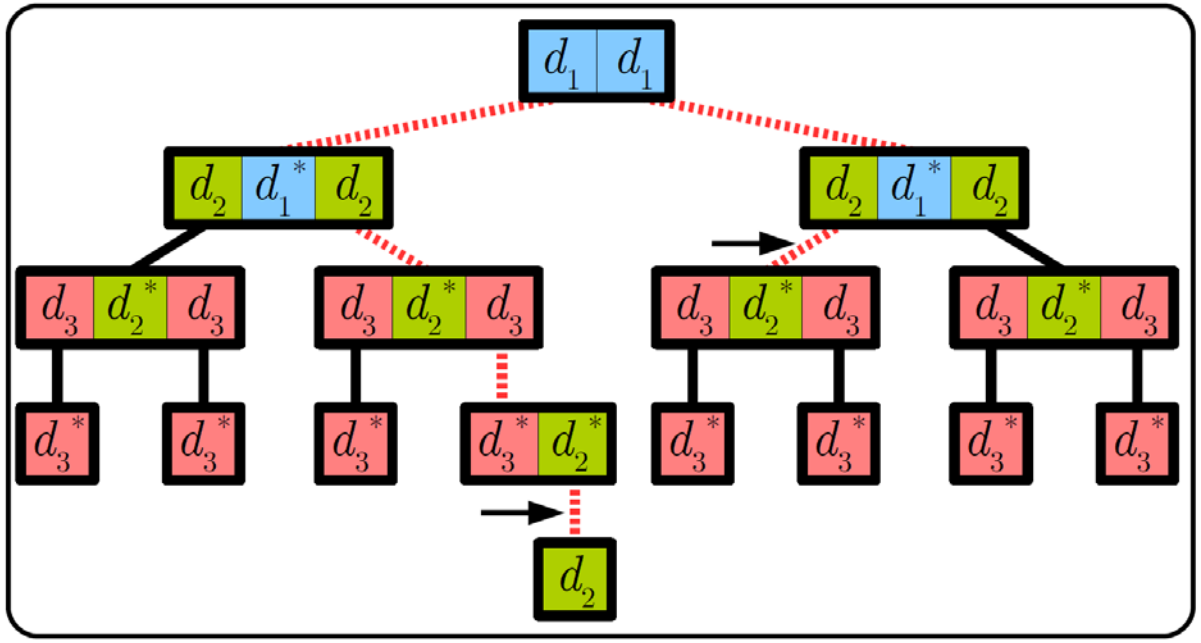
has polymers of size $\leq 2(M+D)(AD)^{2D+3} = \text{poly}(D^D)$ if $A = O(1)$

Easy proof if binding graph is acyclic (tree)



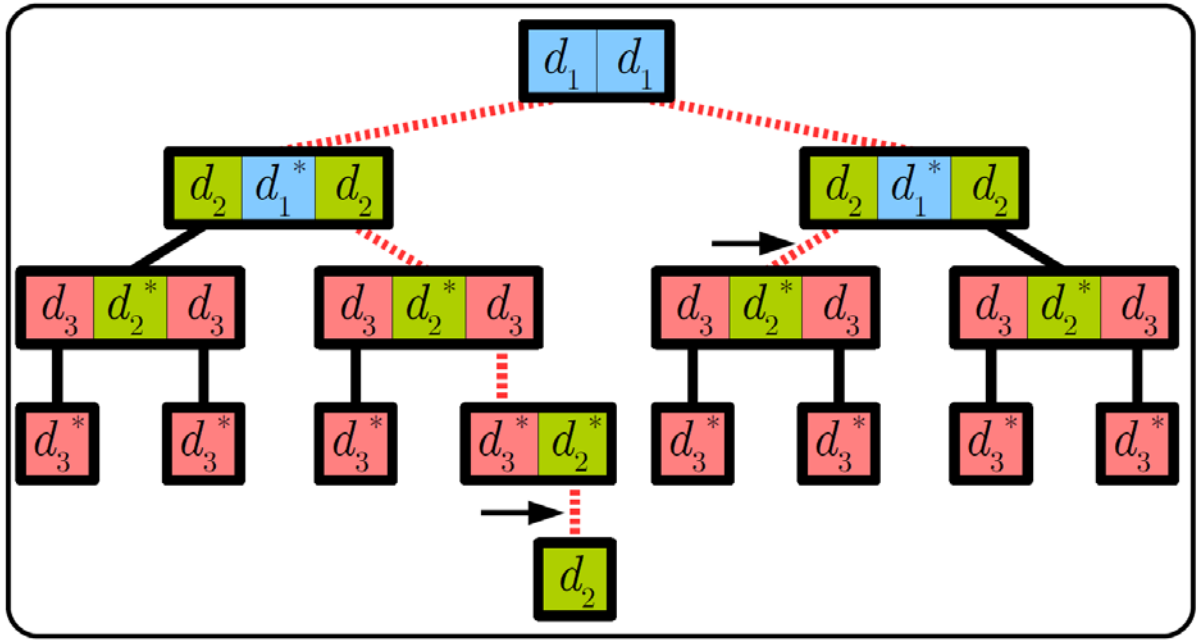
Easy proof if binding graph is acyclic (tree)

- Since monomers have O(1) domains, binding graph is bounded degree



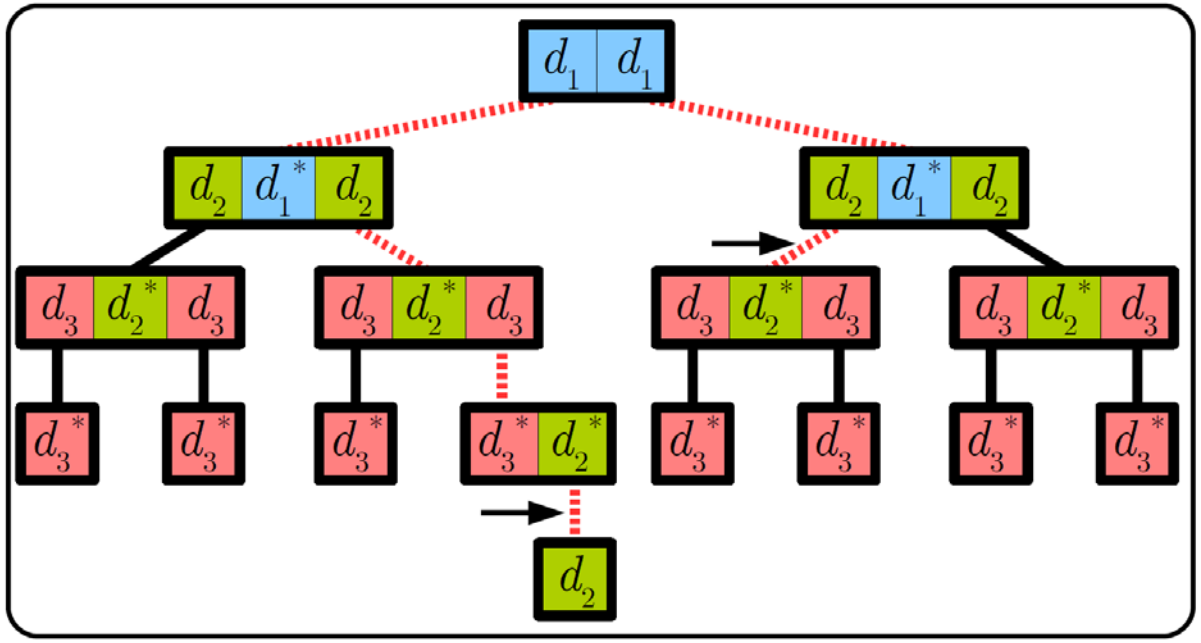
Easy proof if binding graph is acyclic (tree)

- Since monomers have O(1) domains, binding graph is bounded degree
- # nodes of tree is at most exponential in depth (longest path length ≤ 2·depth)



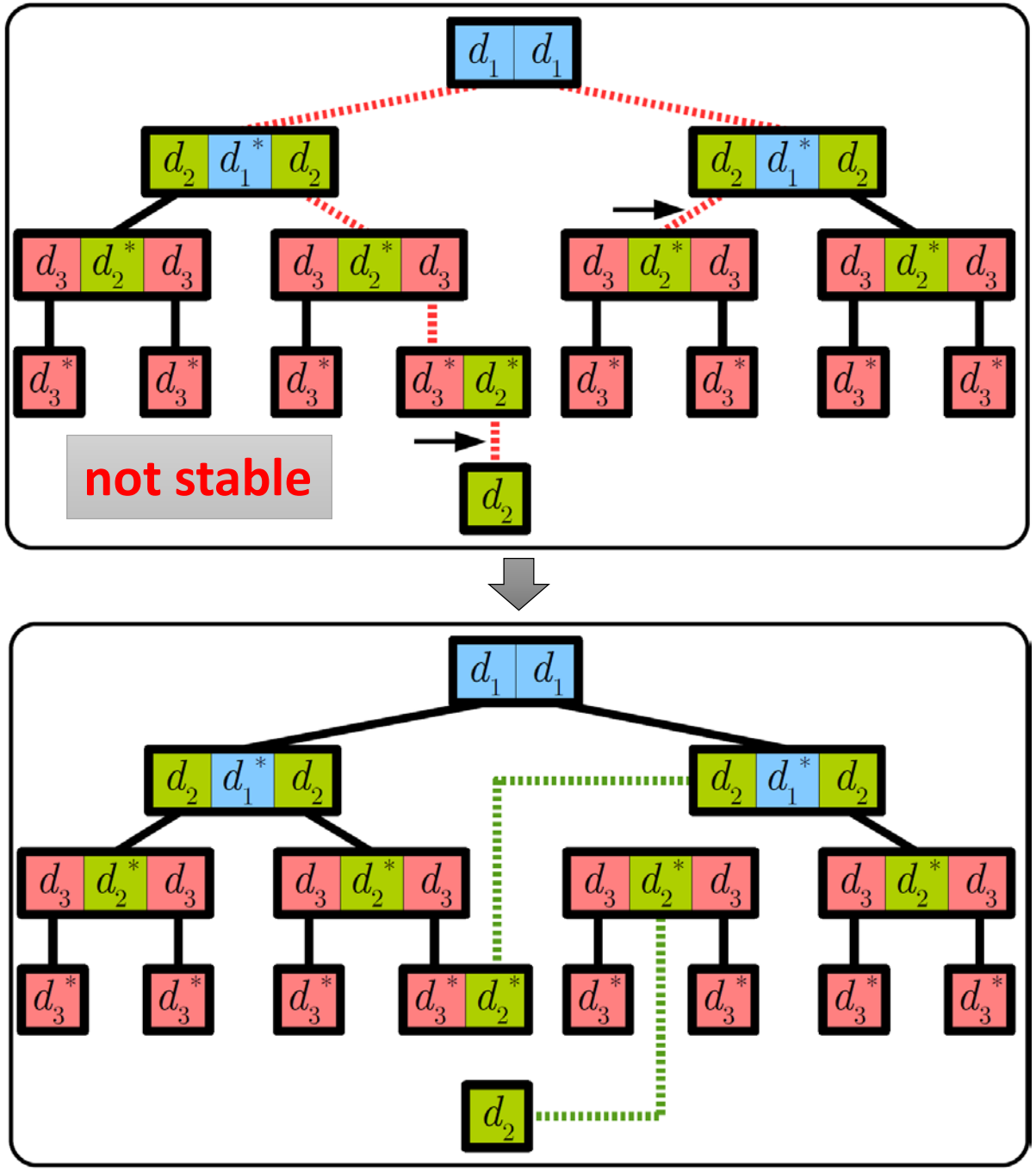
Easy proof if binding graph is acyclic (tree)

- Since monomers have O(1) domains, binding graph is bounded degree
- # nodes of tree is at most exponential in depth (longest path length ≤ 2·depth)
- If some path has > 2*D* edges, it must repeat some ordered pair (d_i,d_i*) or (d_i*,d_i)



Easy proof if binding graph is acyclic (tree)

- Since monomers have O(1) domains, binding graph is bounded degree
- # nodes of tree is at most exponential in depth (longest path length ≤ 2·depth)
- If some path has > 2*D* edges, it must repeat some ordered pair (d_i,d_i*) or (d_i*,d_i)
- Break into two saturated polymers as shown.



A digression into computational complexity

- INTEGER-PROGRAMMING problem

  Given: integer matrix **A**, integer vector **b**

  Question: is there a nonnegative integer vector **x** such that **Ax** = **b**?

A digression into computational complexity

- INTEGER-PROGRAMMING problem

  Given: integer matrix **A**, integer vector **b**

  Question: is there a nonnegative integer vector **x** such that **Ax** = **b**?

- 0/1-INTEGER-PROGRAMMING is **NP**-complete (Karp 1972).

A digression into computational complexity

- INTEGER-PROGRAMMING problem

  Given: integer matrix **A**, integer vector **b**

  Question: is there a nonnegative integer vector **x** such that **Ax** = **b**?

- 0/1-INTEGER-PROGRAMMING is **NP**-complete (Karp 1972).
- Non-obvious fact: INTEGER-PROGRAMMING is in **NP**. *(independently due to [Borosh and Treybig 1976], [Gathen and Sieveking 1978], [Kannan and Monma 1978])*

A digression into computational complexity

- INTEGER-PROGRAMMING problem

  Given: integer matrix **A**, integer vector **b**

  Question: is there a nonnegative integer vector **x** such that **Ax** = **b**?

- 0/1-INTEGER-PROGRAMMING is **NP**-complete (Karp 1972).

- Non-obvious fact: INTEGER-PROGRAMMING is in **NP**. *(independently due to [Borosh and Treybig 1976], [Gathen and Sieveking 1978], [Kannan and Monma 1978])*

  If **Ax** = **b** has a solution, it has a "small" solution… $\max_i \mathbf{x}_i \leq \exp(\max_{ij}(\mathbf{A}_{ij},\mathbf{b}_j))$

A digression into computational complexity

- INTEGER-PROGRAMMING problem

  Given: integer matrix **A**, integer vector **b**

  Question: is there a nonnegative integer vector **x** such that **Ax** = **b**?

- 0/1-INTEGER-PROGRAMMING is **NP**-complete (Karp 1972).
- Non-obvious fact: INTEGER-PROGRAMMING is in **NP**. *(independently due to [Borosh and Treybig 1976], [Gathen and Sieveking 1978], [Kannan and Monma 1978])*

  If **Ax** = **b** has a solution, it has a "small" solution… $\max_i \mathbf{x}_i \leq \exp(\max_{ij}(\mathbf{A}_{ij},\mathbf{b}_j))$
- Papadimitriou's proof: [*On the complexity of integer programming*. Papadimitriou, JACM 1981]
  - If **x** is a *large enough* solution, there is $\mathbf{0} < \mathbf{y} < \mathbf{x}$, $\mathbf{y} \in \mathbb{N}^m$, such that **Ay** = **0**.

A digression into computational complexity

- INTEGER-PROGRAMMING problem

  Given: integer matrix **A**, integer vector **b**

  Question: is there a nonnegative integer vector **x** such that **Ax** = **b**?

- 0/1-INTEGER-PROGRAMMING is **NP**-complete (Karp 1972).
- Non-obvious fact: INTEGER-PROGRAMMING is in **NP**. *(independently due to [Borosh and Treybig 1976], [Gathen and Sieveking 1978], [Kannan and Monma 1978])*

  If **Ax** = **b** has a solution, it has a "small" solution… $\max_i \mathbf{x}_i \leq \exp(\max_{ij}(\mathbf{A}_{ij},\mathbf{b}_j))$
- Papadimitriou's proof: [*On the complexity of integer programming*. Papadimitriou, JACM 1981]
  - If **x** is a *large enough* solution, there is $\mathbf{0} < \mathbf{y} < \mathbf{x}$, $\mathbf{y} \in \mathbb{N}^m$, such that **Ay** = **0**.
  - Defining **z** = **x** – **y**, **Az** = **A**(**x** – **y**) = **Ax** – **Ay** = **Ax** – **0** = **b**.

A digression into computational complexity

- INTEGER-PROGRAMMING problem

  Given: integer matrix **A**, integer vector **b**

  Question: is there a nonnegative integer vector **x** such that **Ax** = **b**?

- 0/1-INTEGER-PROGRAMMING is **NP**-complete (Karp 1972).
- Non-obvious fact: INTEGER-PROGRAMMING is in **NP**. *(independently due to [Borosh and Treybig 1976], [Gathen and Sieveking 1978], [Kannan and Monma 1978])*

  If **Ax** = **b** has a solution, it has a "small" solution… $\max_i \mathbf{x}_i \leq \exp(\max_{ij}(\mathbf{A}_{ij},\mathbf{b}_j))$
- Papadimitriou's proof: [*On the complexity of integer programming*. Papadimitriou, JACM 1981]
  - If **x** is a *large enough* solution, there is $\mathbf{0} < \mathbf{y} < \mathbf{x}$, $\mathbf{y} \in \mathbb{N}^m$, such that **Ay** = **0**.
  - Defining **z** = **x** – **y**, **Az** = **A**(**x** – **y**) = **Ax** – **Ay** = **Ax** – **0** = **b**.
  - So **z** is a strictly smaller solution than **x**: **x** cannot be the *smallest* solution.

Monomers as vectors

- monomer {a, b*,b*, d,d,d,d,d*, e,e*} represented as (1,-2,0,3,0)

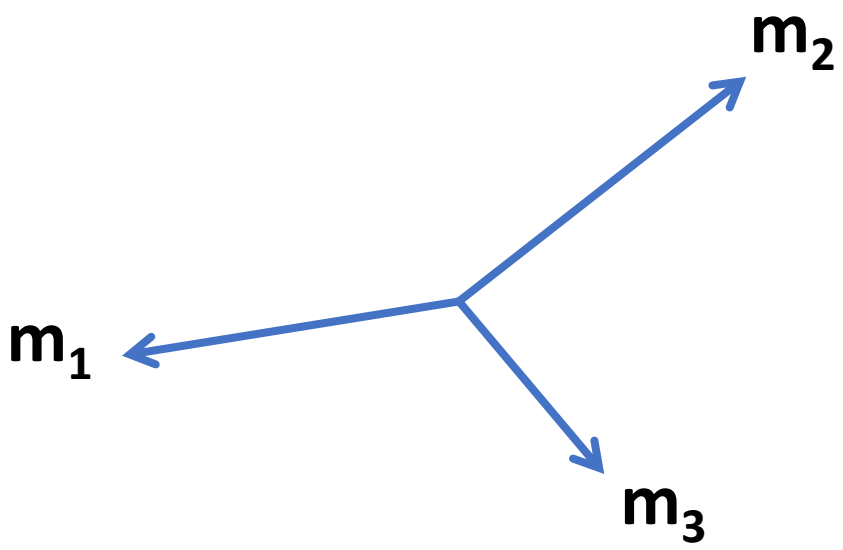
Monomers as vectors

- monomer {a, b*,b*, d,d,d,d,d*, e,e*} represented as (1,-2,0,3,0)
- sum of many monomers gives the number of excess domains in a fully bound (saturated) polymer with those monomers
  - i.e., 2 copies of above monomer 2·(1,-2,0,3,0) = (**2**,**-4**,**0**,**6**,**0**) have an excess of **2** a's, **4** b*'s, **0** c's, **6** d's, **0** e's

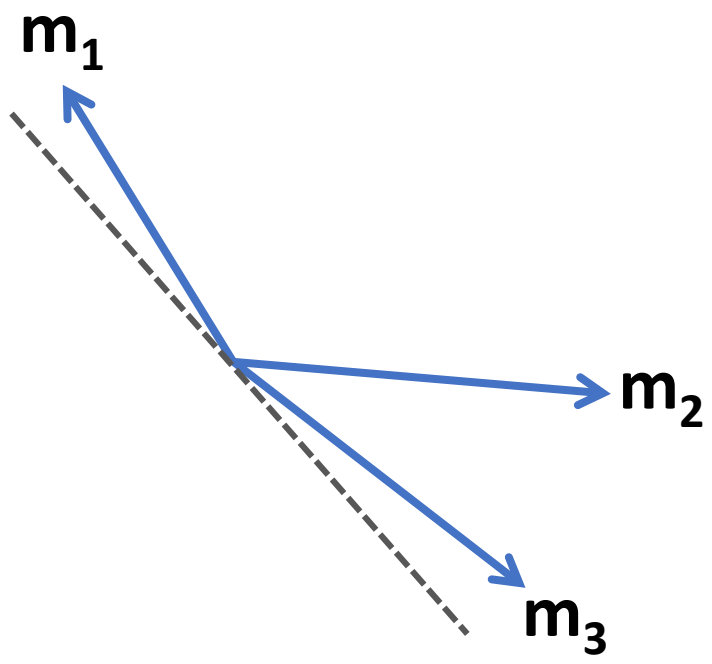
Farkas' Lemma

Given vectors $\mathbf{m}_1$, $\mathbf{m}_2$, … , they obey one of two constraints:

a) are directions of balanced forces



b) lie on one side of some hyperplane



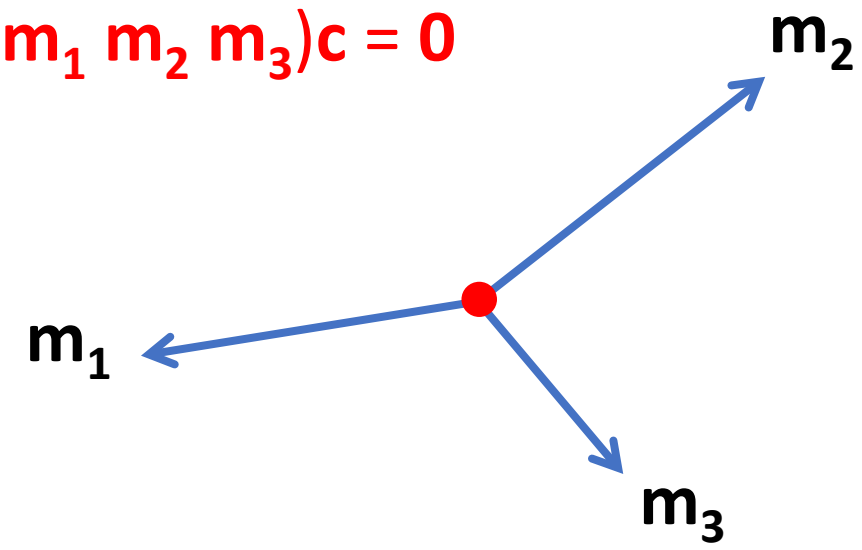
Farkas' Lemma

Given vectors $\mathbf{m}_1$, $\mathbf{m}_2$, … , they obey one of two constraints:

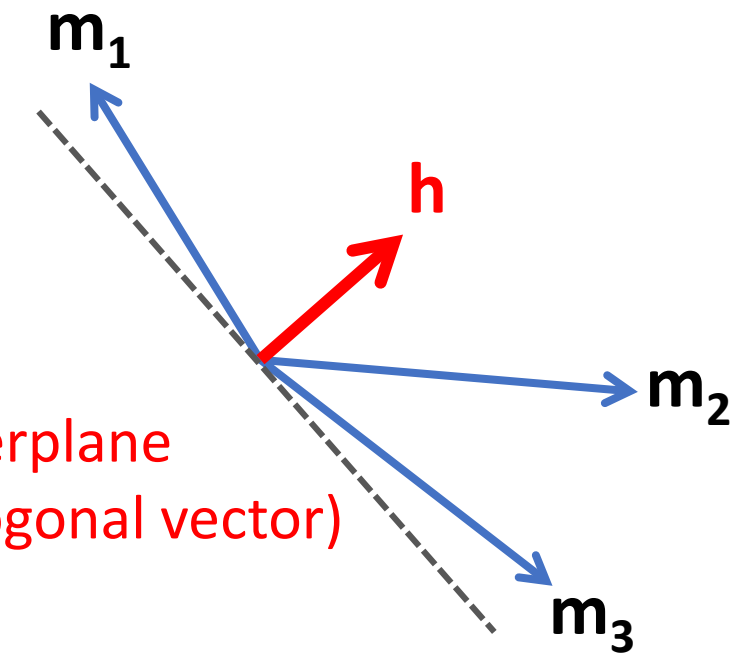
a) are directions of balanced forces

$\exists \mathbf{c}$ (counts of monomers)

$(\mathbf{m}_1\ \mathbf{m}_2\ \mathbf{m}_3)\mathbf{c} = \mathbf{0}$



b) lie on one side of some hyperplane

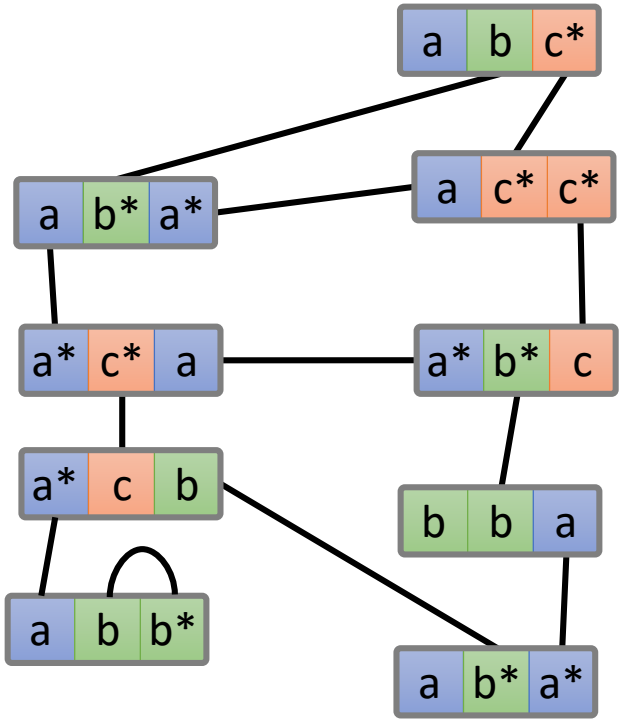


$\exists \mathbf{h}$ (hyperplane orthogonal vector)

$\mathbf{h} \cdot \mathbf{m}_i \geq 1$

How to prove exponential polymer size bound for polymers with cycles in binding graph?

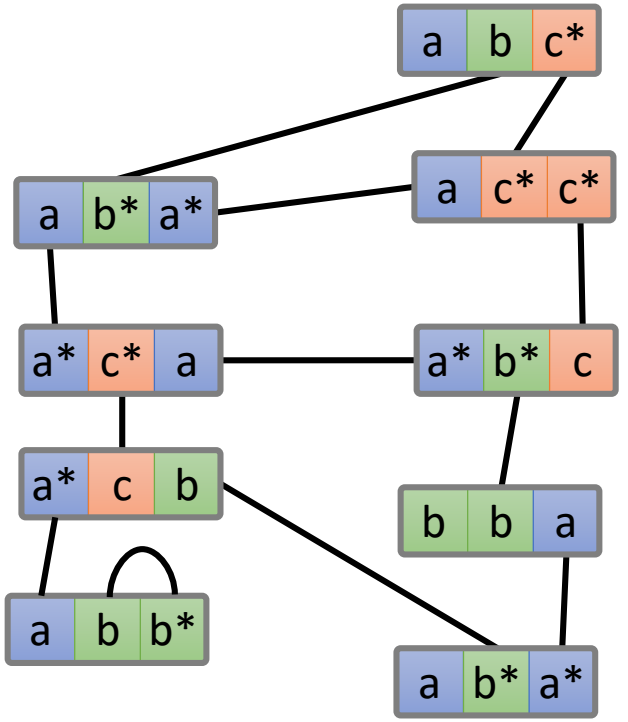
monomer collection $\mathbf{c} \in \mathbb{N}^M$



How to prove exponential polymer size bound for polymers with cycles in binding graph?

- **A** = *d* x *m* matrix: $\mathbf{A}_{ij}$ = monomer $\mathbf{m}_j$'s excess of domain d_i over d_i*

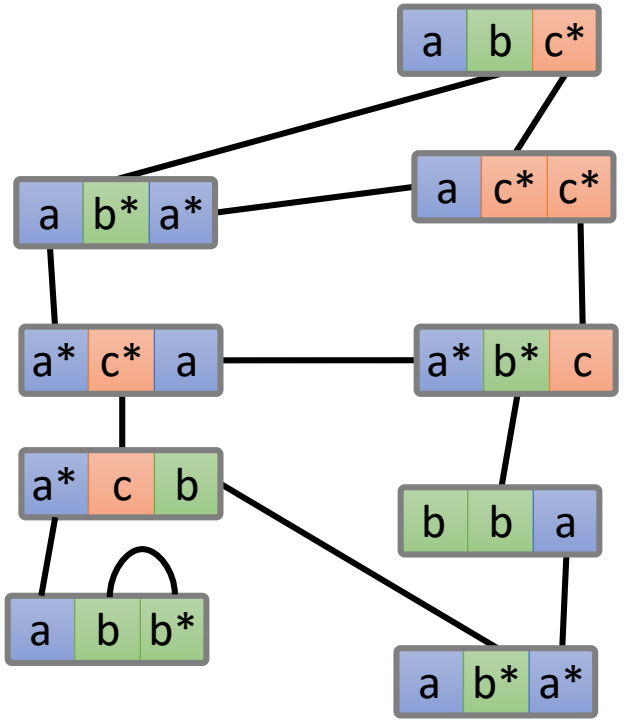
monomer collection $\mathbf{c} \in \mathbb{N}^M$



How to prove exponential polymer size bound for polymers with cycles in binding graph?

- **A** = *d* x *m* matrix: $\mathbf{A}_{ij}$ = monomer $\mathbf{m}_j$'s excess of domain d_i over d_i*
- If **Ac** = **b**, then $\mathbf{b}_i$ = total # unbound d_i in any saturated configuration of **x**

monomer collection $\mathbf{c} \in \mathbb{N}^M$



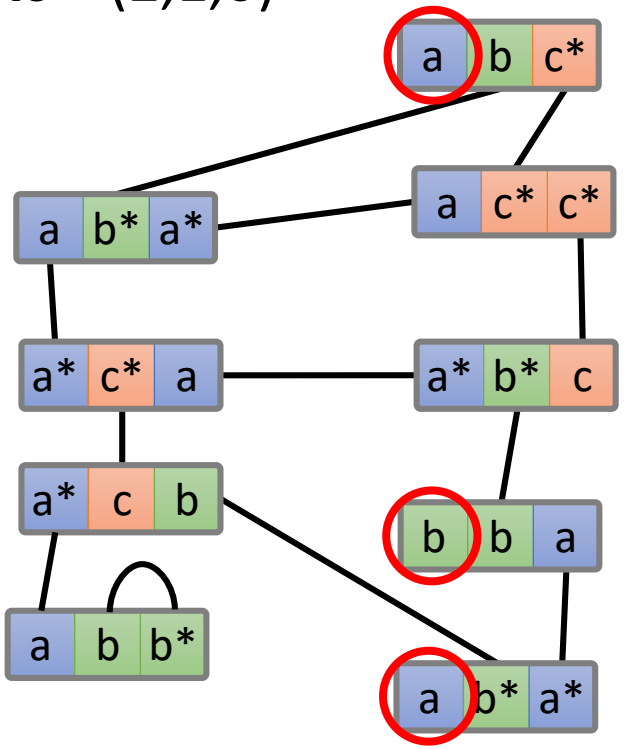
How to prove exponential polymer size bound for polymers with cycles in binding graph?

- **A** = *d* x *m* matrix: $\mathbf{A}_{ij}$ = monomer $\mathbf{m}_j$'s excess of domain d_i over d_i^*
- If **Ac** = **b**, then $\mathbf{b}_i$ = total # unbound d_i in any saturated configuration of **x**

monomer collection $\mathbf{c} \in \mathbb{N}^M$

a b c

Ac = (2,1,0)



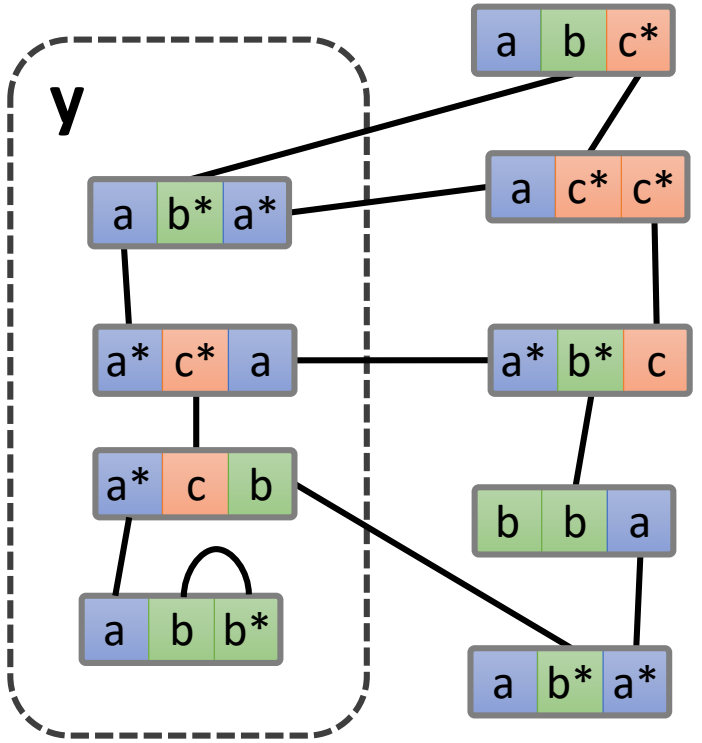
How to prove exponential polymer size bound for polymers with cycles in binding graph?

- **A** = *d* x *m* matrix: $\mathbf{A}_{ij}$ = monomer $\mathbf{m}_j$'s excess of domain d_i over d_i*
- If **Ac** = **b**, then $\mathbf{b}_i$ = total # unbound d_i in any saturated configuration of **x**
- If |**c**| > exponential in *D*, Papadimtriou's proof gives us subcollection **y** < **c** such that **Ay** = **0**, *(Farkas' Lemma says that if this fails, then monomer vectors all lie on one side of a hyperplane, see next slide)*

monomer collection **c** $\in \mathbb{N}^M$

a b c

Ac = (2,1,0)



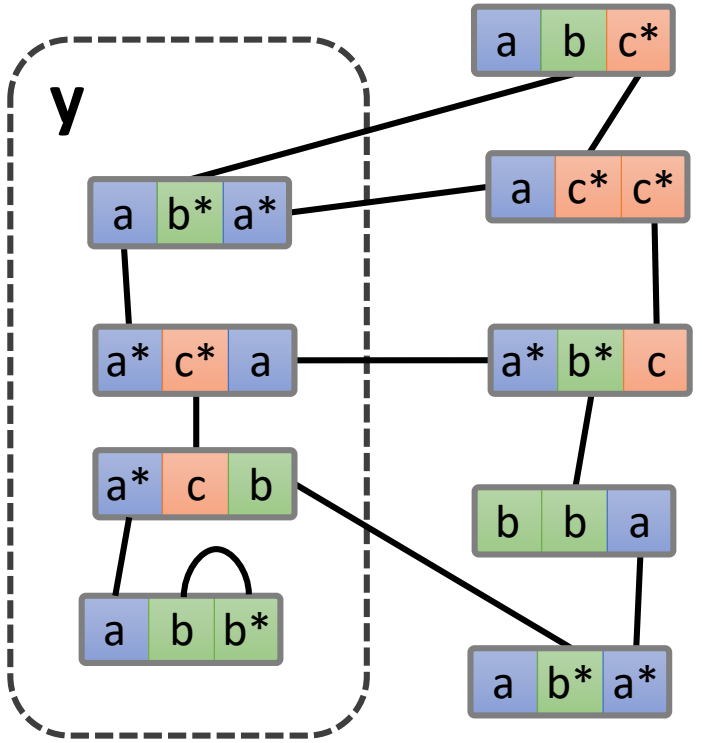
How to prove exponential polymer size bound for polymers with cycles in binding graph?

- **A** = *d* x *m* matrix: $\mathbf{A}_{ij}$ = monomer $\mathbf{m}_j$'s excess of domain d_i over d_i*
- If **Ac** = **b**, then $\mathbf{b}_i$ = total # unbound d_i in any saturated configuration of **x**
- If |**c**| > exponential in *D*, Papadimtriou's proof gives us subcollection **y** < **c** such that **Ay** = **0**, *(Farkas' Lemma says that if this fails, then monomer vectors all lie on one side of a hyperplane, see next slide)*
- i.e., #d_i in **y** = #d_i* in **y**, so **y** is self-saturating.

monomer collection **c** $\in \mathbb{N}^M$

a b c

Ac = (2,1,0)



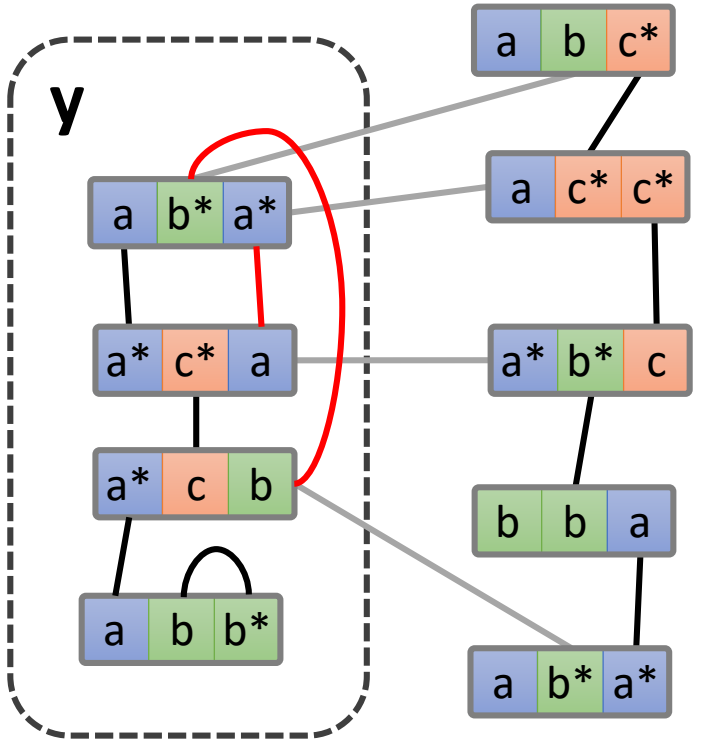
How to prove exponential polymer size bound for polymers with cycles in binding graph?

- **A** = *d* x *m* matrix: $\mathbf{A}_{ij}$ = monomer $\mathbf{m}_j$'s excess of domain d_i over d_i*
- If **Ac** = **b**, then $\mathbf{b}_i$ = total # unbound d_i in any saturated configuration of **x**
- If |**c**| > exponential in *D*, Papadimtriou's proof gives us subcollection **y** < **c** such that **Ay** = **0**, *(Farkas' Lemma says that if this fails, then monomer vectors all lie on one side of a hyperplane, see next slide)*
- i.e., #d_i in **y** = #d_i* in **y**, so **y** is self-saturating.
- So whatever bonds were broken to separate **y** can be re-bound within **y**.

monomer collection **c** $\in \mathbb{N}^M$

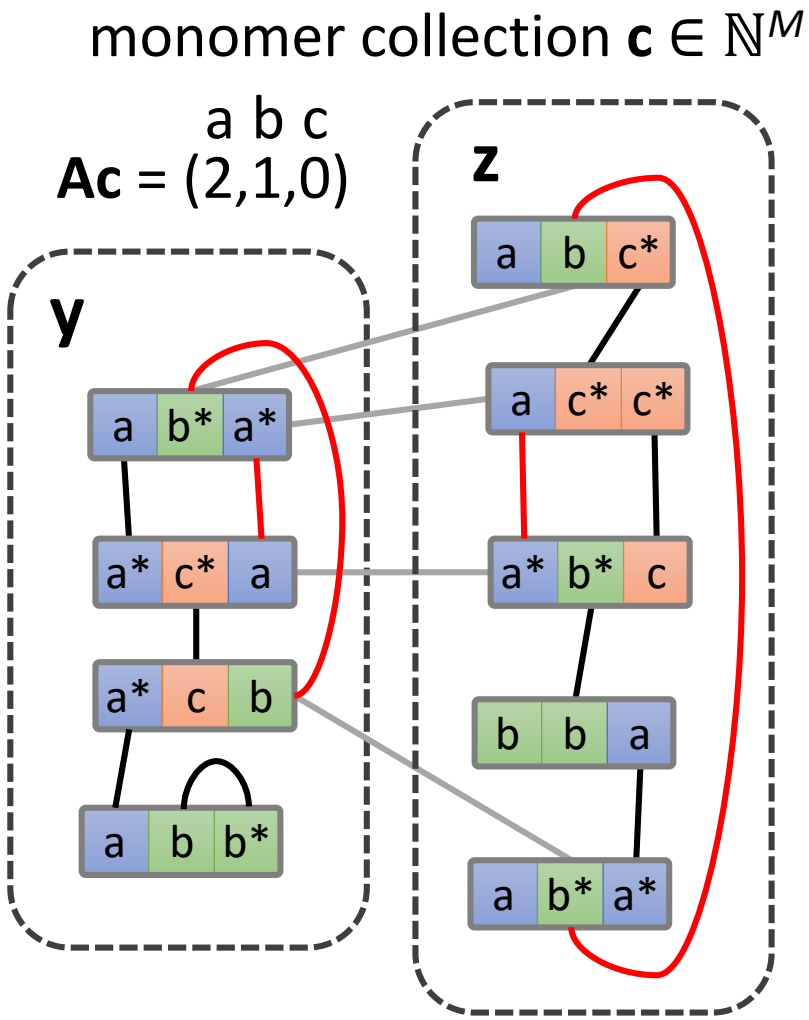
a b c

Ac = (2,1,0)



How to prove exponential polymer size bound for polymers with cycles in binding graph?

- **A** = *d* x *m* matrix: $\mathbf{A}_{ij}$ = monomer $\mathbf{m}_j$'s excess of domain d_i over d_i*
- If **Ac** = **b**, then $\mathbf{b}_i$ = total # unbound d_i in any saturated configuration of **x**
- If |**c**| > exponential in *D*, Papadimtriou's proof gives us subcollection **y** < **c** such that **Ay** = **0**, *(Farkas' Lemma says that if this fails, then monomer vectors all lie on one side of a hyperplane, see next slide)*
- i.e., #d_i in **y** = #d_i* in **y**, so **y** is self-saturating.
- So whatever bonds were broken to separate **y** can be re-bound within **y**.
- By symmetry, the same bonds in **z** = **c** – **y** can be re-bound within **z**.



If all monomer types lie on one side of hyperplane **h**…

If all monomer types lie on one side of hyperplane **h**…

- Consider “slack monomers” {d_1*}, {d_2*},…, adding just enough to bind to all the excess d_i domains, so **saturated** (fully bound) == **all domains bound**

If all monomer types lie on one side of hyperplane **h**…

- Consider “slack monomers” {d_1*}, {d_2*},…, adding just enough to bind to all the excess d_i domains, so **saturated** (fully bound) == **all domains bound**
- If **c** is count of all monomers including slack monomers (**c**(i) = count of $\mathbf{m_i}$), then **Ac** = **0**, where each column of **A** represents a monomer (counts of domains).

If all monomer types lie on one side of hyperplane **h**…

- Consider “slack monomers” $\{d_1^*\}$, $\{d_2^*\}$,…, adding just enough to bind to all the excess d_i domains, so **saturated** (fully bound) == **all domains bound**
- If **c** is count of all monomers including slack monomers (**c**(i) = count of $\mathbf{m_i}$), then **Ac** = **0**, where each column of **A** represents a monomer (counts of domains).
- dot-product **h** on both sides: $\mathbf{h}\cdot\mathbf{Ac} = \mathbf{h}\cdot\mathbf{0} = 0$, distribute through: $\sum_i(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i) = 0$

If all monomer types lie on one side of hyperplane **h**…

- Consider "slack monomers" $\{d_1^*\}$, $\{d_2^*\}$,…, adding just enough to bind to all the excess d_i domains, so **saturated** (fully bound) == **all domains bound**
- If **c** is count of all monomers including slack monomers (**c**(i) = count of $\mathbf{m_i}$), then **Ac** = **0**, where each column of **A** represents a monomer (counts of domains).
- dot-product **h** on both sides: $\mathbf{h}\cdot\mathbf{Ac} = \mathbf{h}\cdot\mathbf{0} = 0$, distribute through: $\sum_i(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i) = 0$
- Let *S* be set of monomers with "small" counts, move them to one side:
  - $\sum_{i\in S}(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i) = \sum_{i\notin S}(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i)$

If all monomer types lie on one side of hyperplane **h**…

- Consider "slack monomers" $\{d_1^*\}$, $\{d_2^*\}$,…, adding just enough to bind to all the excess d_i domains, so **saturated** (fully bound) == **all domains bound**
- If **c** is count of all monomers including slack monomers (**c**(i) = count of $\mathbf{m_i}$), then **Ac** = **0**, where each column of **A** represents a monomer (counts of domains).
- dot-product **h** on both sides: $\mathbf{h}\cdot\mathbf{Ac} = \mathbf{h}\cdot\mathbf{0} = 0$, distribute through: $\sum_i(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i) = 0$
- Let *S* be set of monomers with "small" counts, move them to one side:

$$-\sum_{i\in S}(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i) = \sum_{i\notin S}(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i)$$

- Then "small"$_2 \geq -\sum_{i\in S}(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i) = \sum_{i\notin S}(\mathbf{h}\cdot\mathbf{m_i})\mathbf{c}(i) \geq \sum_{i\notin S}\mathbf{c}(i)$

c(i) (count of i'th monomer) is small by definition, and $\mathbf{h}\cdot\mathbf{m_i} = O(1)$

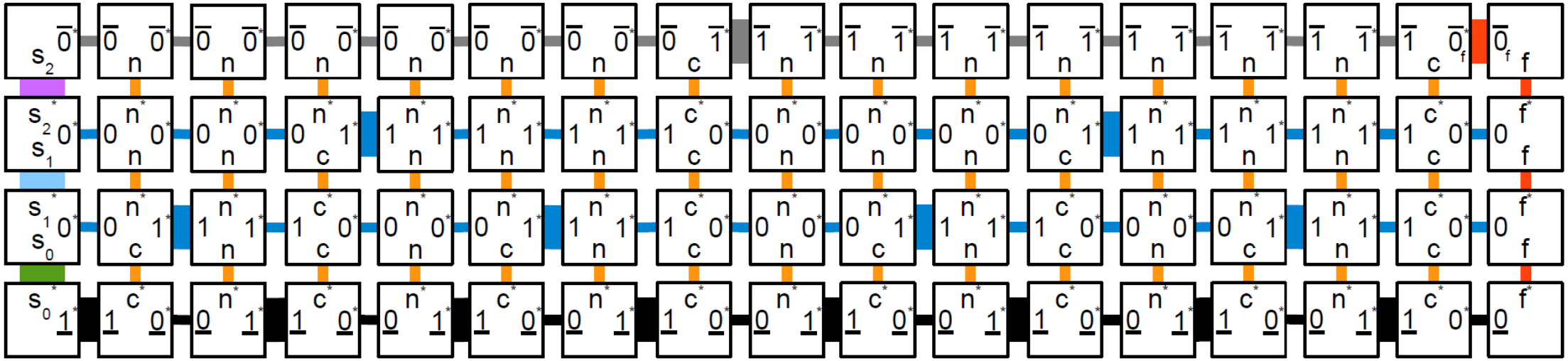
above

since $\mathbf{h}\cdot\mathbf{m_i} \geq 1$

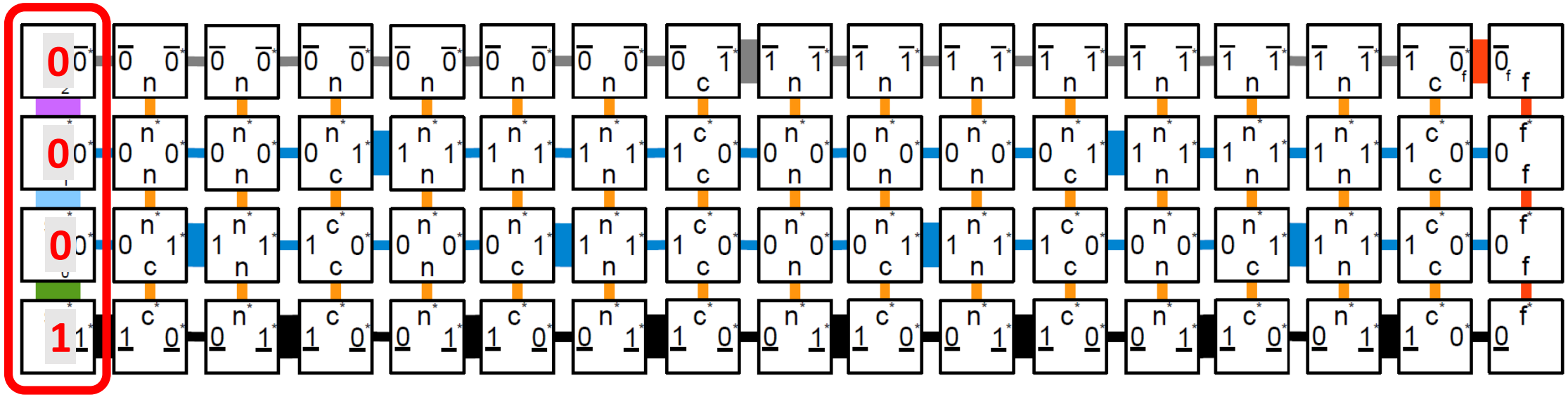
Applying thermodynamic model to tile assembly

- Let’s incorporate the thermodynamic binding network model into the abstract tile assembly model.
- How can we create a large assembly from a small number of tile types?

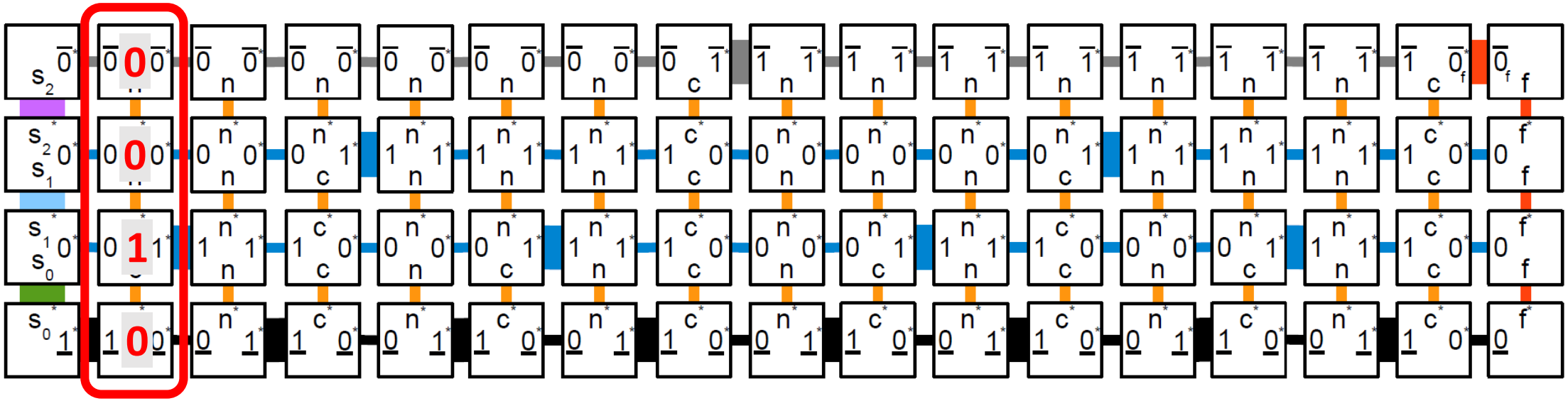
A thermodynamically unstable tile assembly counter



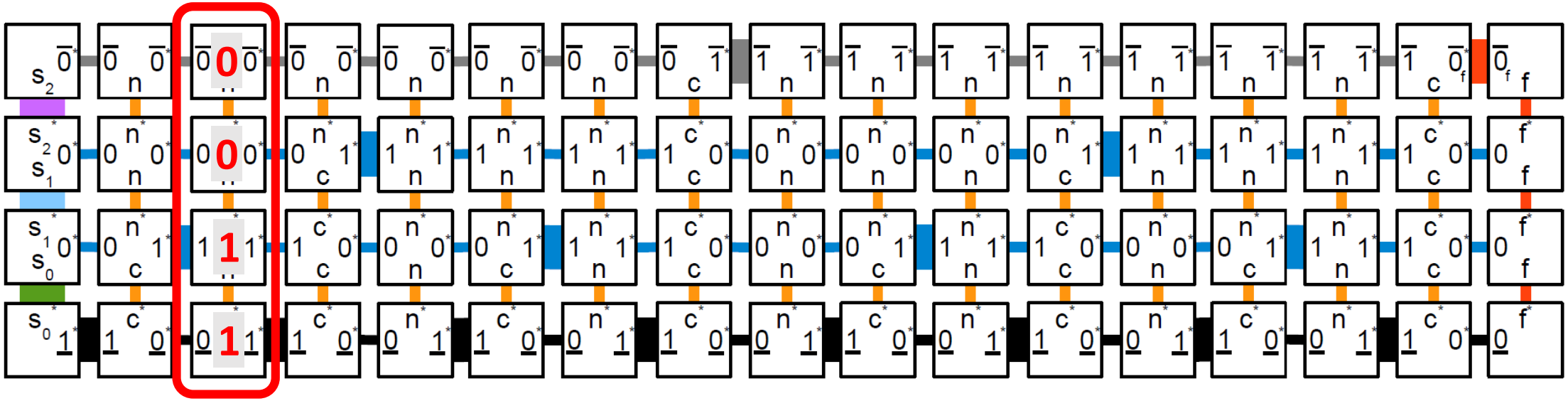
A thermodynamically unstable tile assembly counter



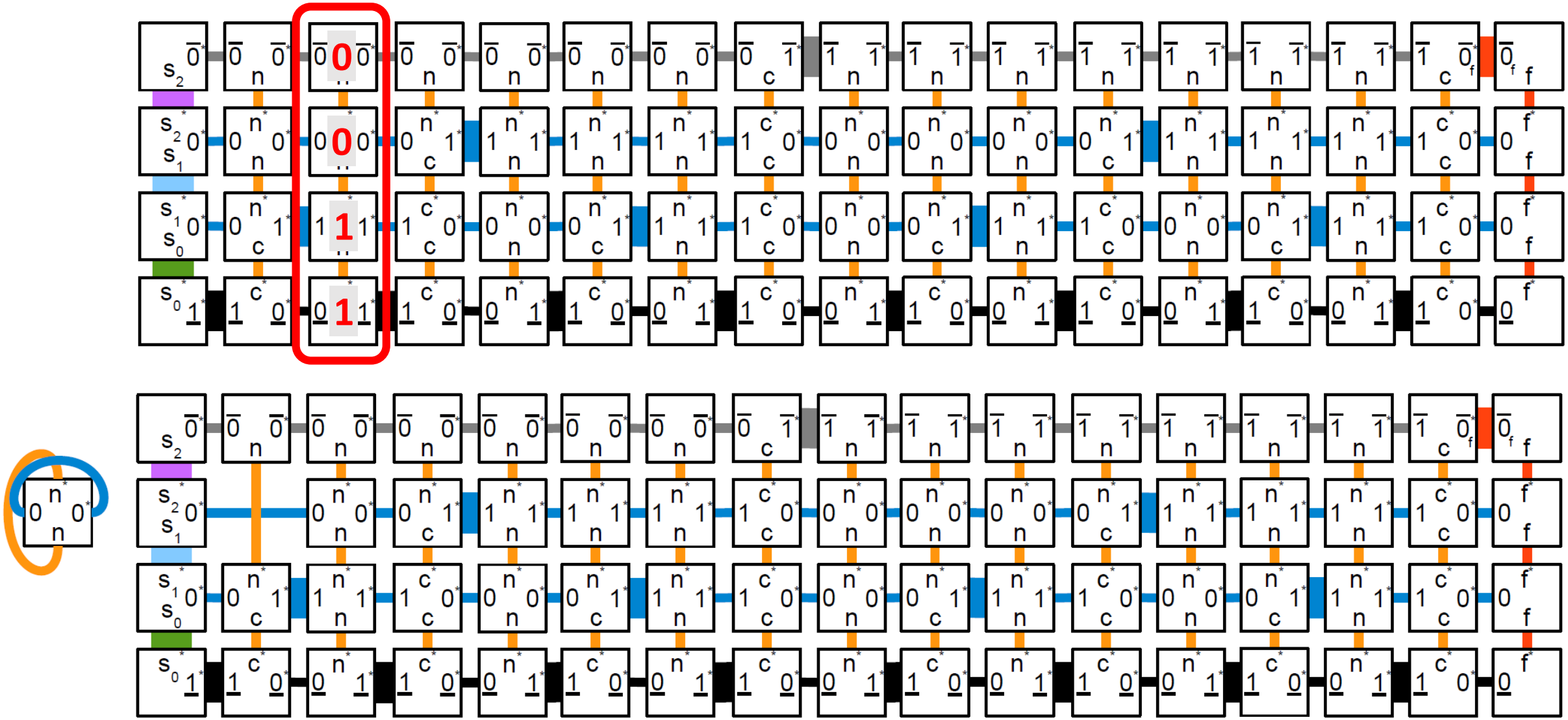
A thermodynamically unstable tile assembly counter



A thermodynamically unstable tile assembly counter

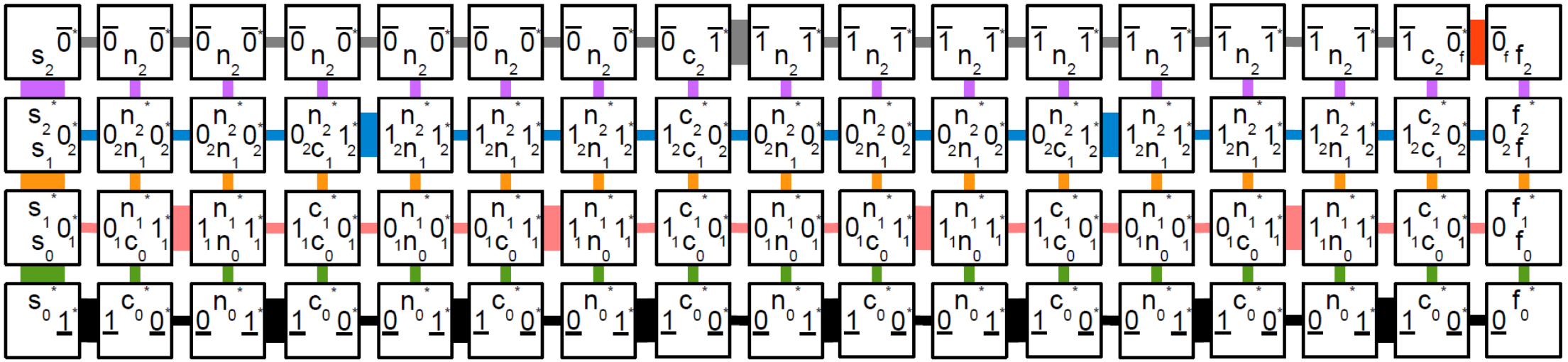


A thermodynamically unstable tile assembly counter



A thermodynamically stable tile assembly counter

Difference is that each row (corresponding to bits of the same significance) has glues labeled with the row number



Thermodynamic self-assembly at UCNC 2018

- *Thermodynamically Favorable Computation via Tile Self-assembly,* Cameron Chalk, Jacob Hendricks, Matthew Patitz, and Michael Sharp (talk on Friday!)

Conclusions

- Strong bonds (surprisingly) aren't sufficient to self-assemble large thermodynamically stable structures. *Geometry helps*!
- Kinetically self-assembling a thermodynamically stable structure has very strong guarantees on errors:
  - target structure eventually results despite arbitrary kinetic errors.
  - If it's the only stable structure, and free energy of other structures is much less, then it's the only result you'll see.
- Bad news: **NP**-complete to tell if a given configuration is unstable… even **NP**-hard to approximate entropy of stable configuration:

  [Breik, Thachuk, Heule, Soloveichik, *Computing properties of stable configurations of thermodynamic binding networks*]

Merci!

Questions?