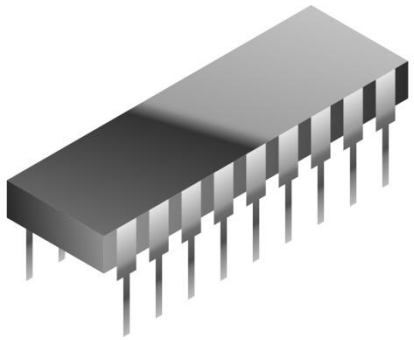
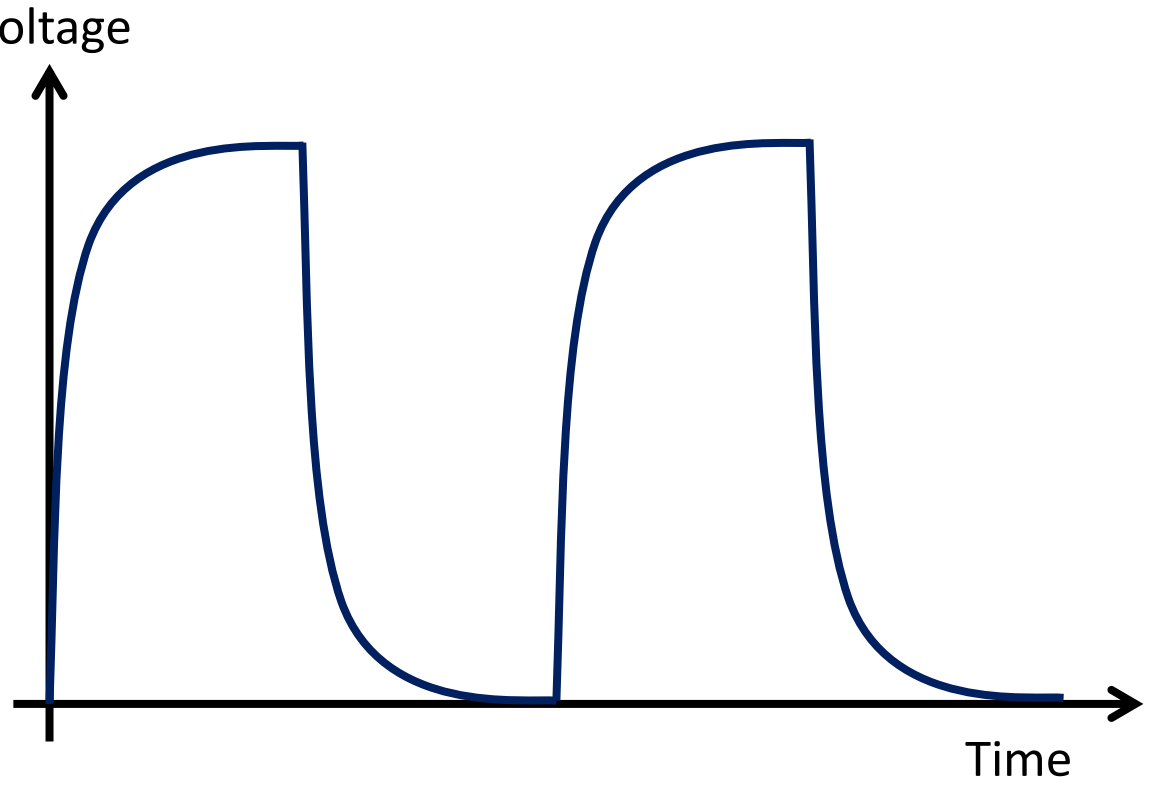
Thermodynamic binding networks

slides © 2021, David Haley and David Doty

ECS 232: Theory of Molecular Computation, UC Davis

Representing Information with Molecules

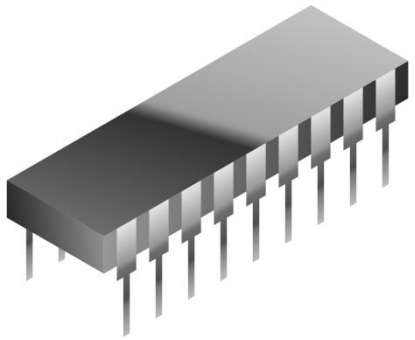


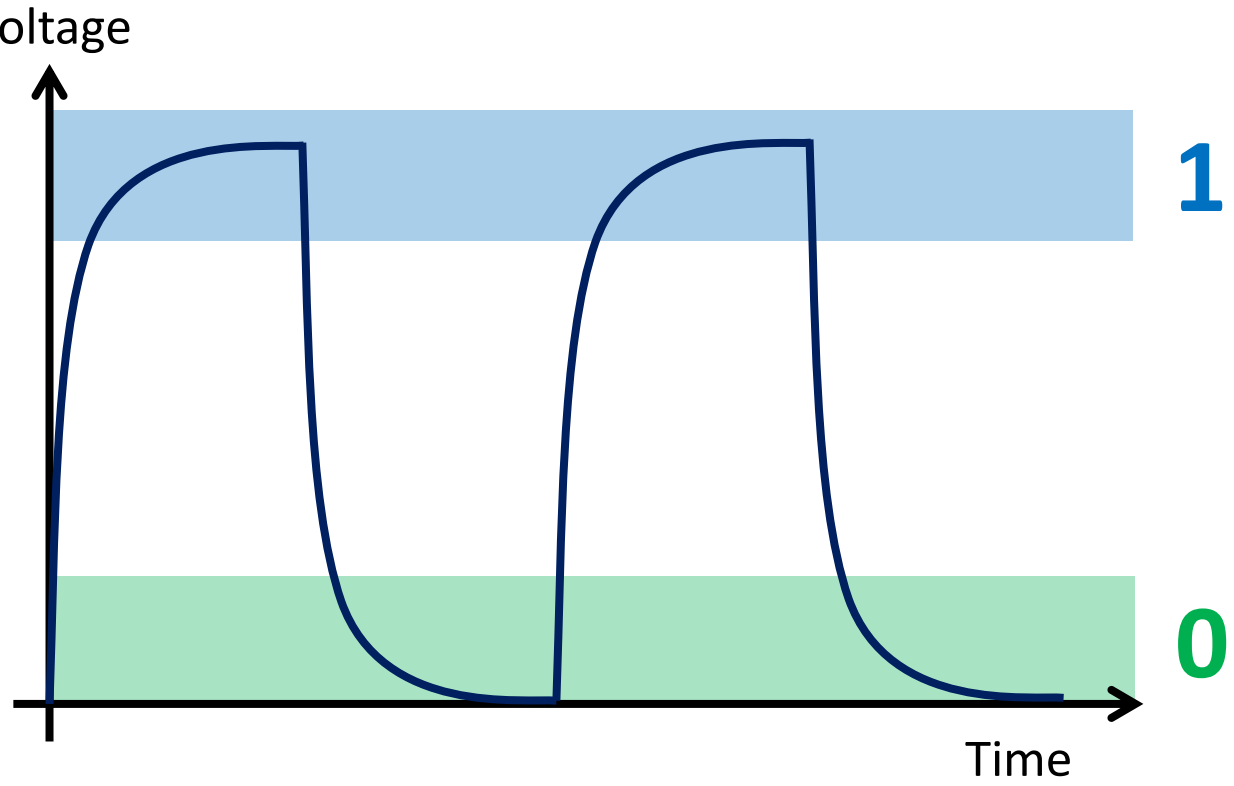


In an electronic circuit,

voltage can represent Boolean input

Representing Information with Molecules

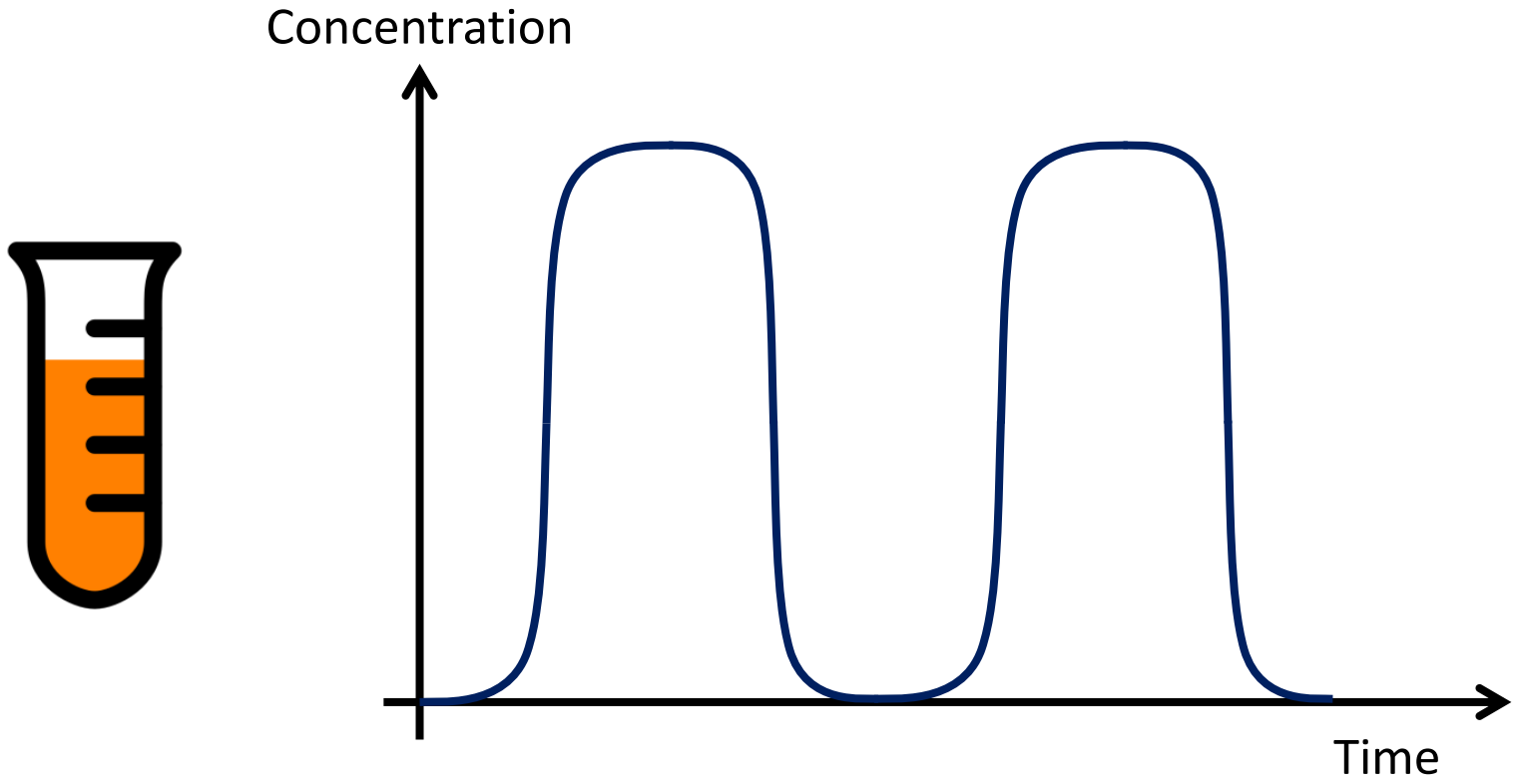




In an electronic circuit,

voltage can represent Boolean input

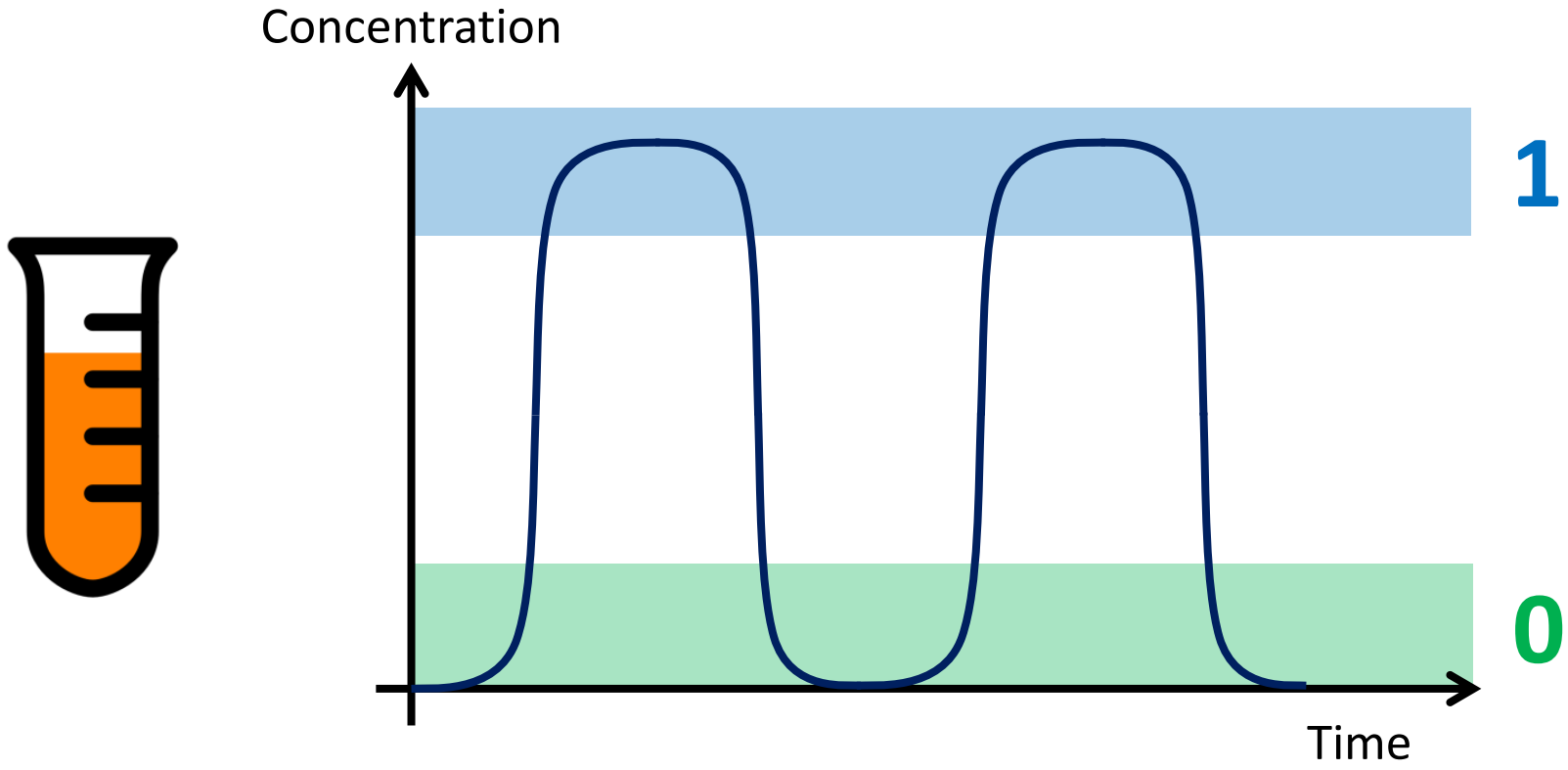
Representing Information with Molecules



In a well-mixed solution,

concentration can represent Boolean input

Representing Information with Molecules

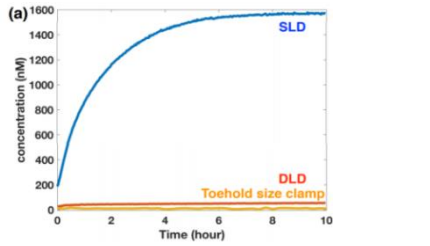


In a well-mixed solution,

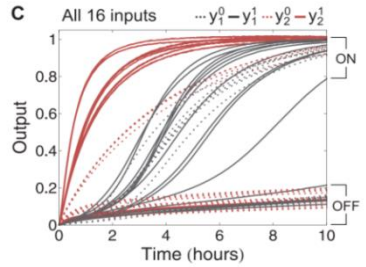
concentration can represent Boolean input

Chemical Identity Gate: Idealized vs. Actual Behavior

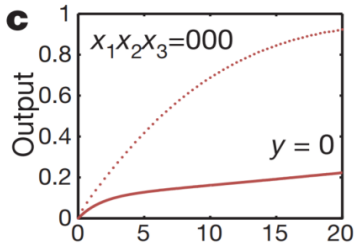
Experimental Implementation of Chemical Logic



Wang et al. ***DNA*** 23, 2017

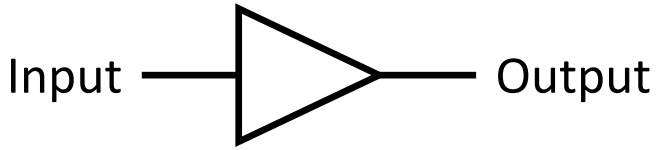


Qian et al. ***Science*** 332, 2011



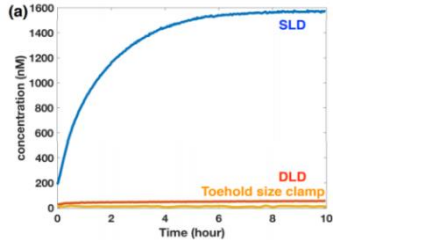
Qian et al. ***Nature*** 475, 2011

(Don't worry about the details in the pictures above)

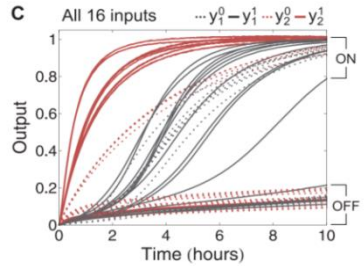


Chemical Identity Gate: Idealized vs. Actual Behavior

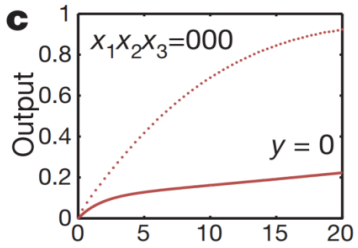
Experimental Implementation of Chemical Logic



Wang et al. ***DNA*** 23, 2017

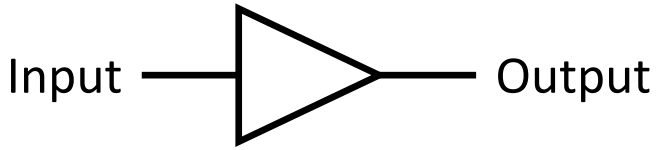


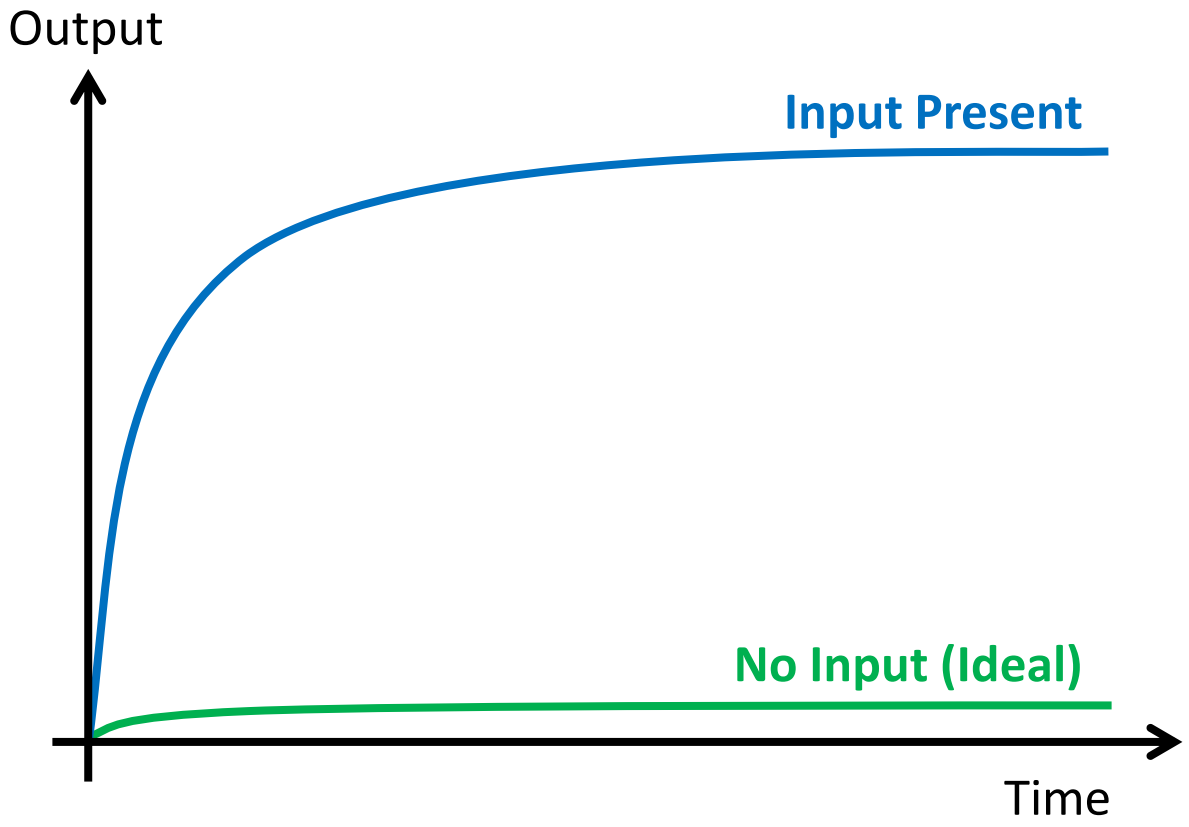
Qian et al. ***Science*** 332, 2011



Qian et al. ***Nature*** 475, 2011

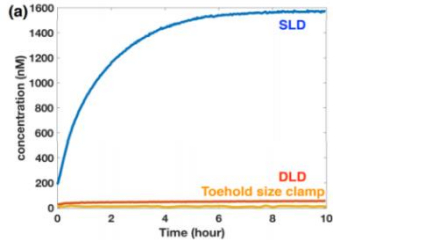
(Don't worry about the details in the pictures above)



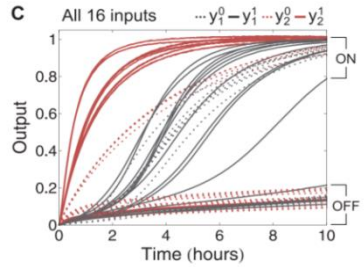


Chemical Identity Gate: Idealized vs. Actual Behavior

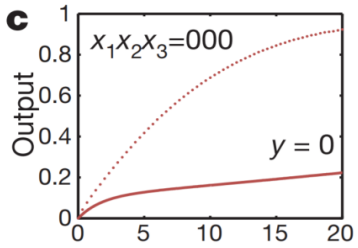
Experimental Implementation of Chemical Logic



Wang et al. ***DNA*** 23, 2017

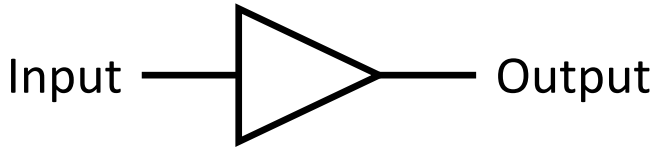


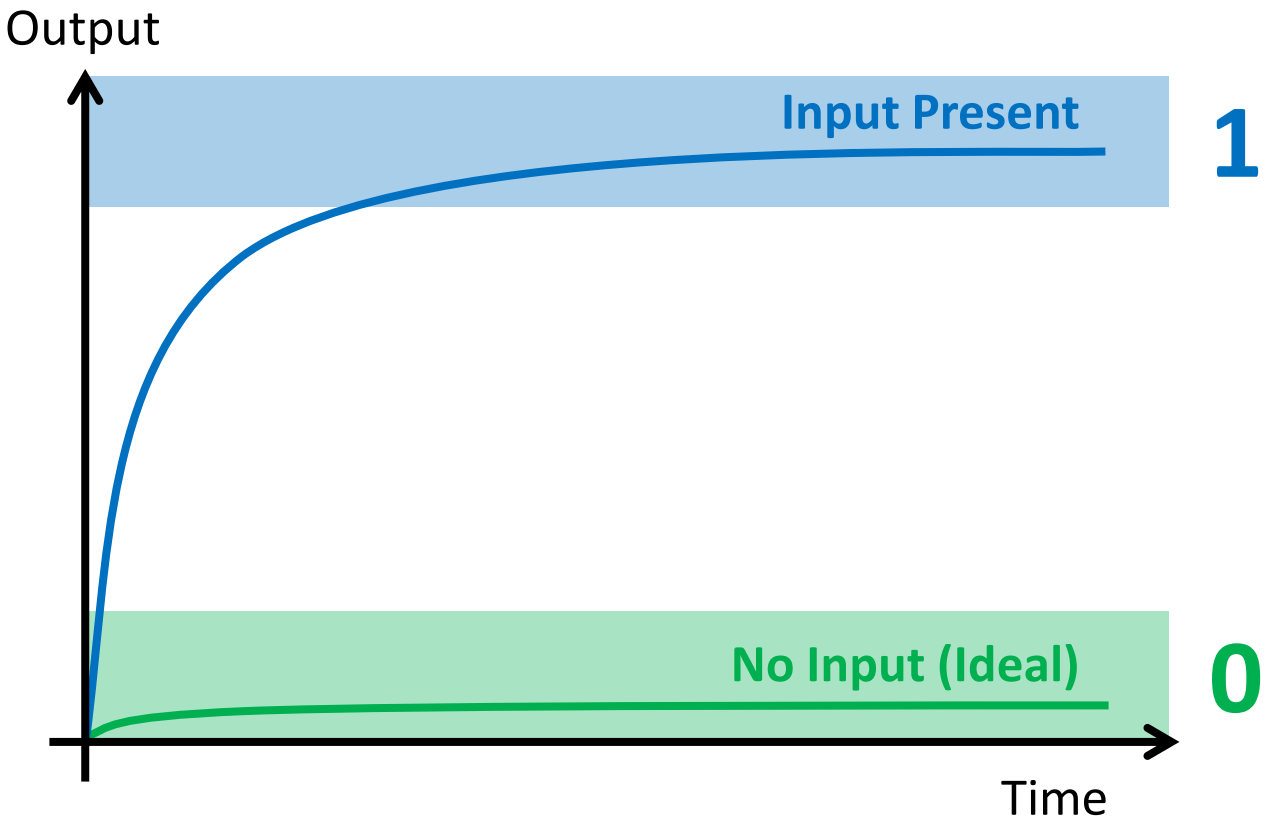
Qian et al. ***Science*** 332, 2011



Qian et al. ***Nature*** 475, 2011

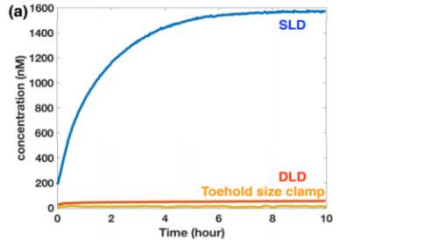
(Don't worry about the details in the pictures above)



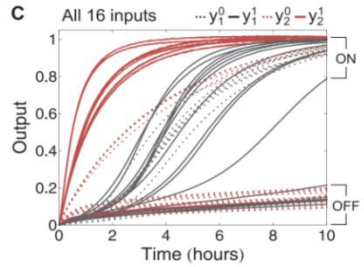


Chemical Identity Gate: Idealized vs. Actual Behavior

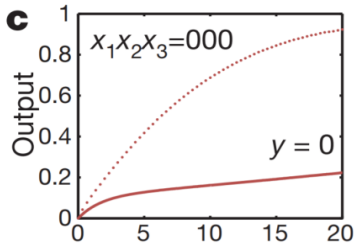
Experimental Implementation of Chemical Logic



Wang et al. ***DNA*** 23, 2017

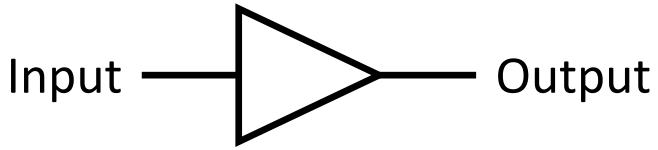


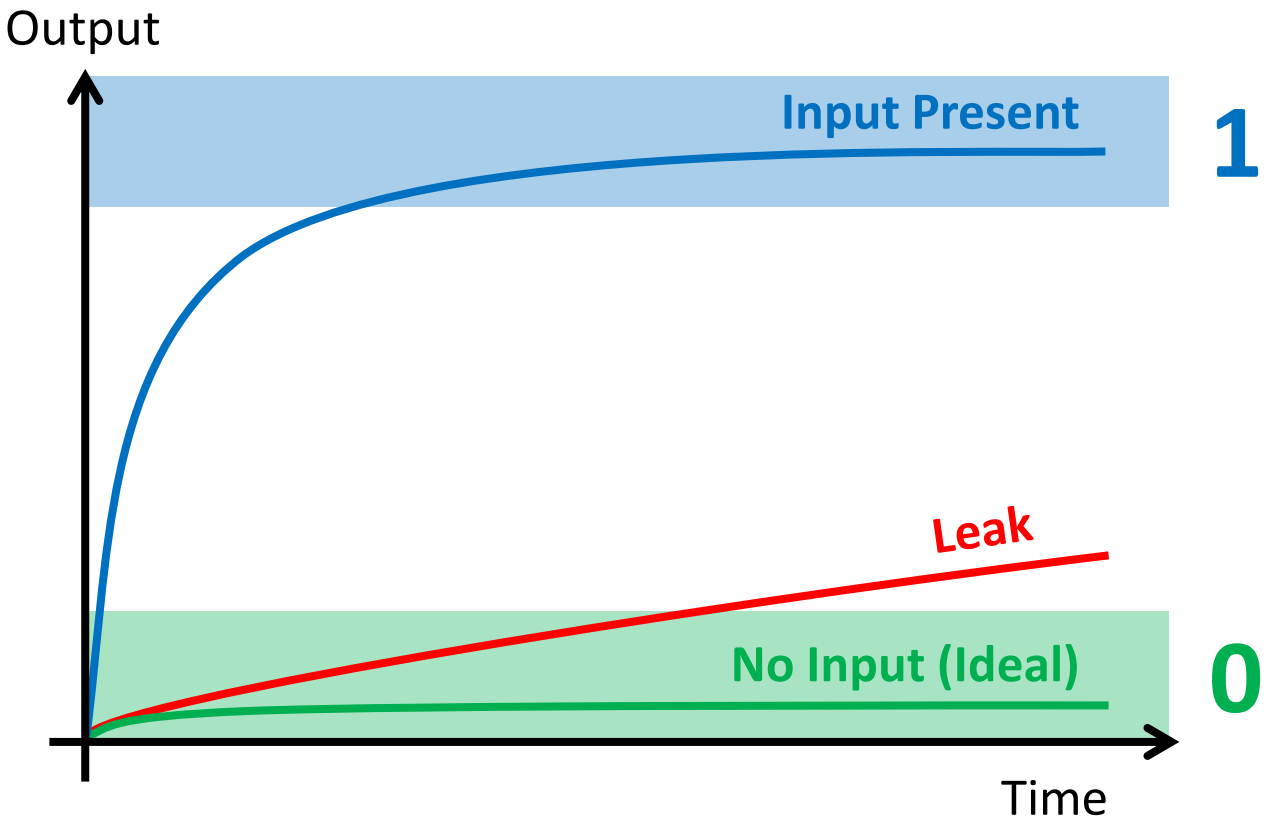
Qian et al. ***Science*** 332, 2011



Qian et al. ***Nature*** 475, 2011

(Don't worry about the details in the pictures above)





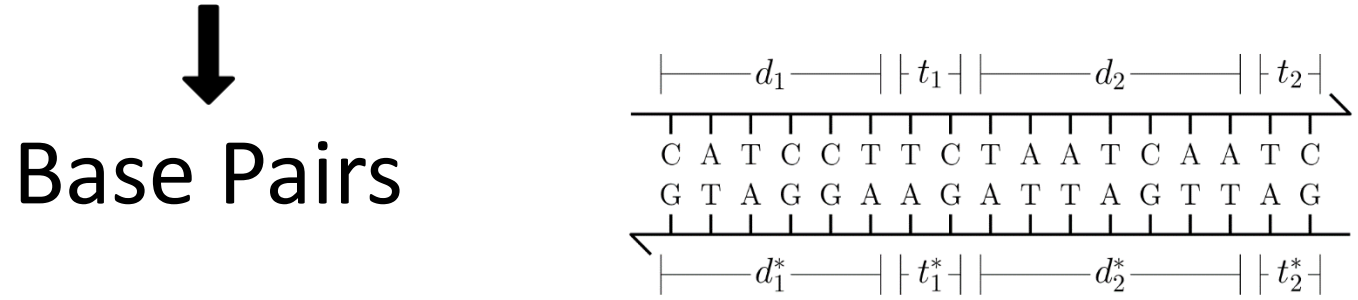
Levels of Abstraction

Levels of Abstraction

DNA

Levels of Abstraction

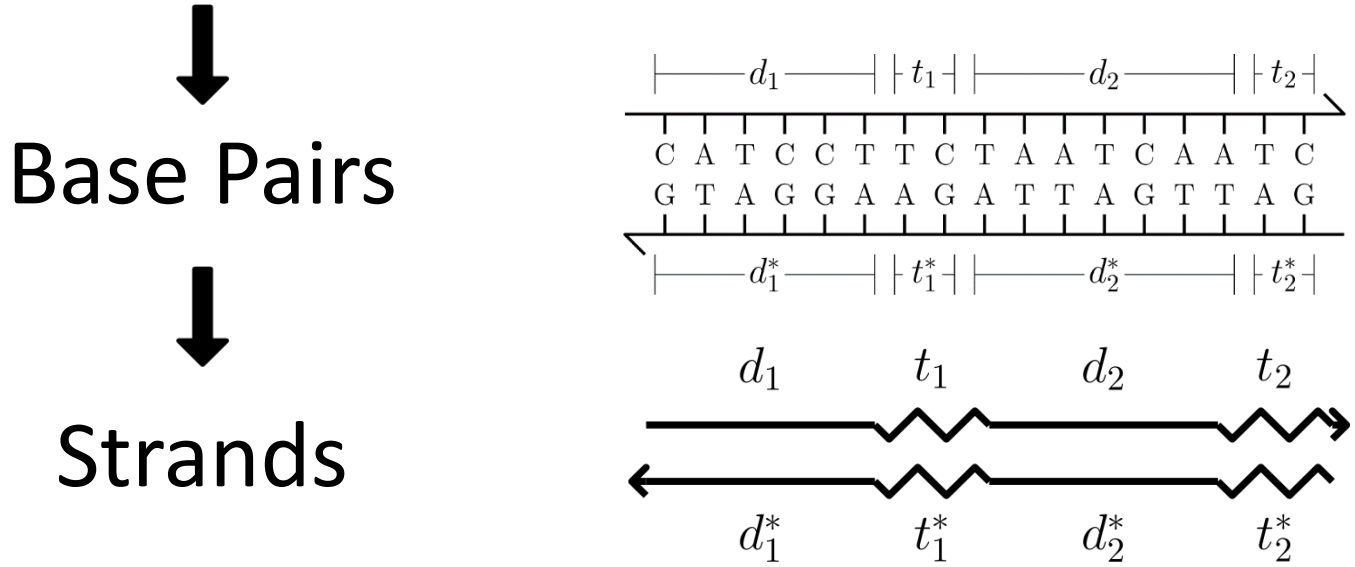
DNA



Base Pairs

Levels of Abstraction

DNA



Base Pairs

Strands

Levels of Abstraction

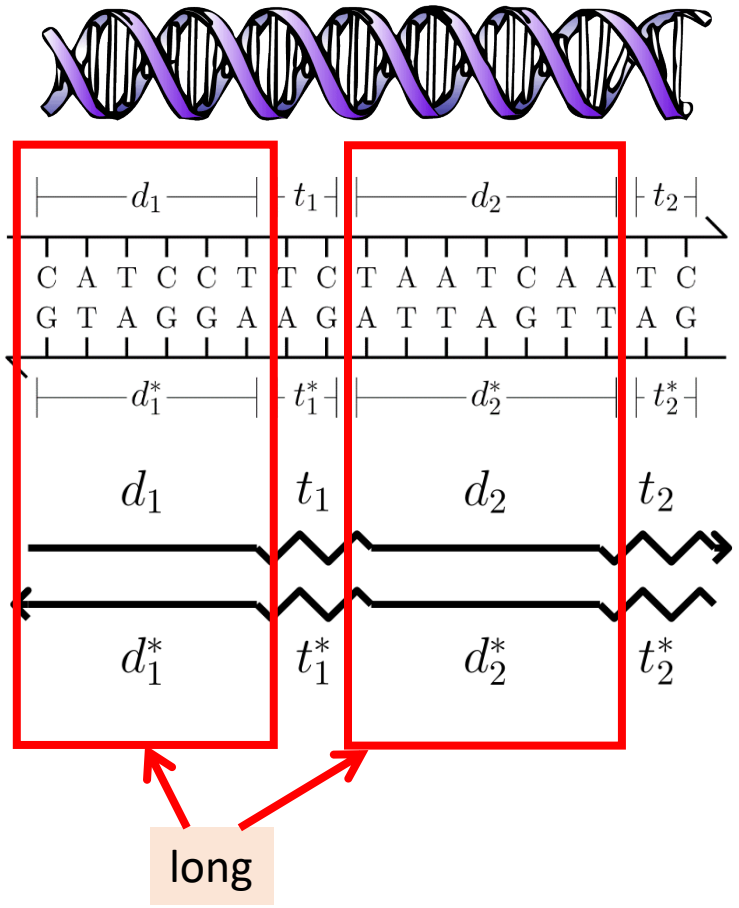
DNA

⬇

Base Pairs

⬇

Strands

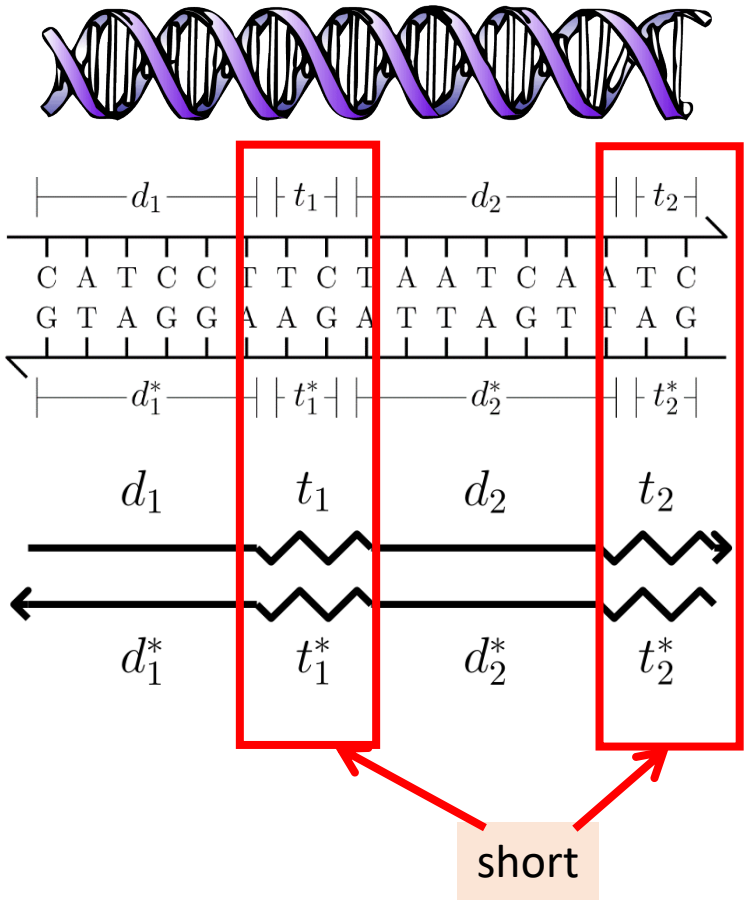


Levels of Abstraction

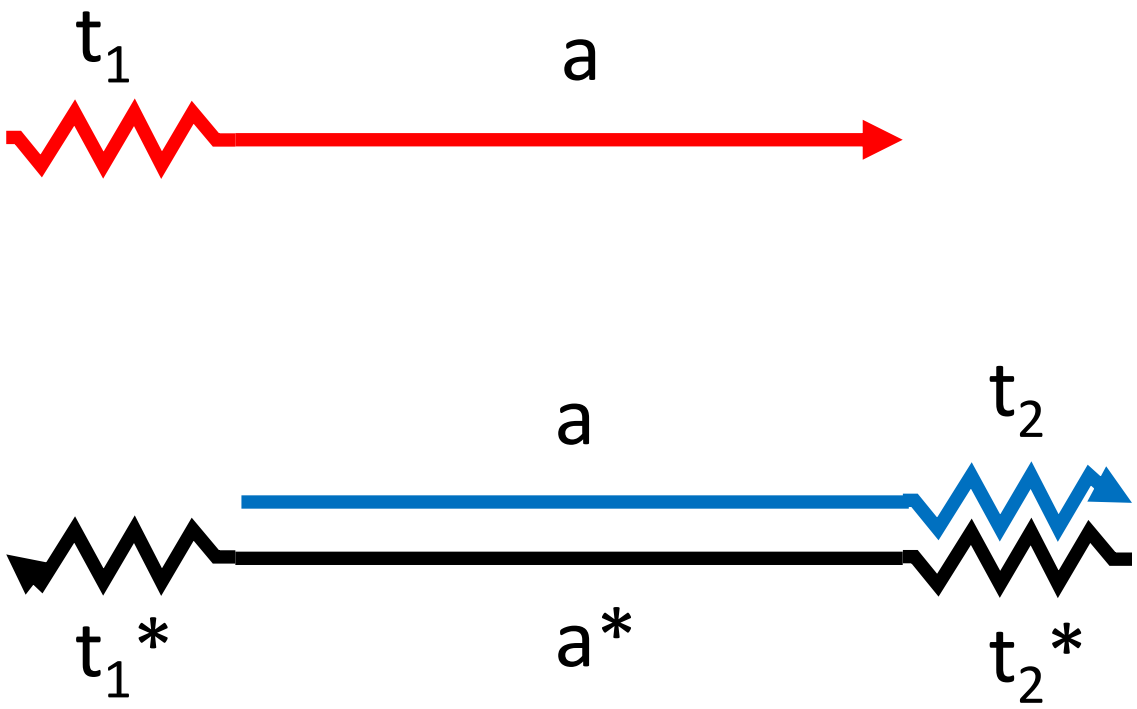
DNA

Base Pairs

Strands

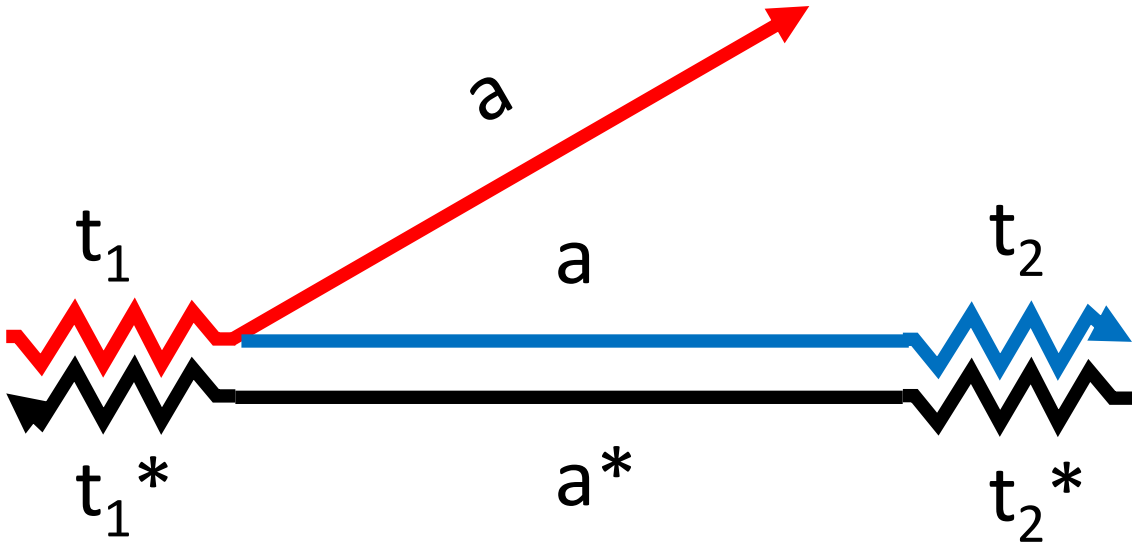


DNA strand displacement



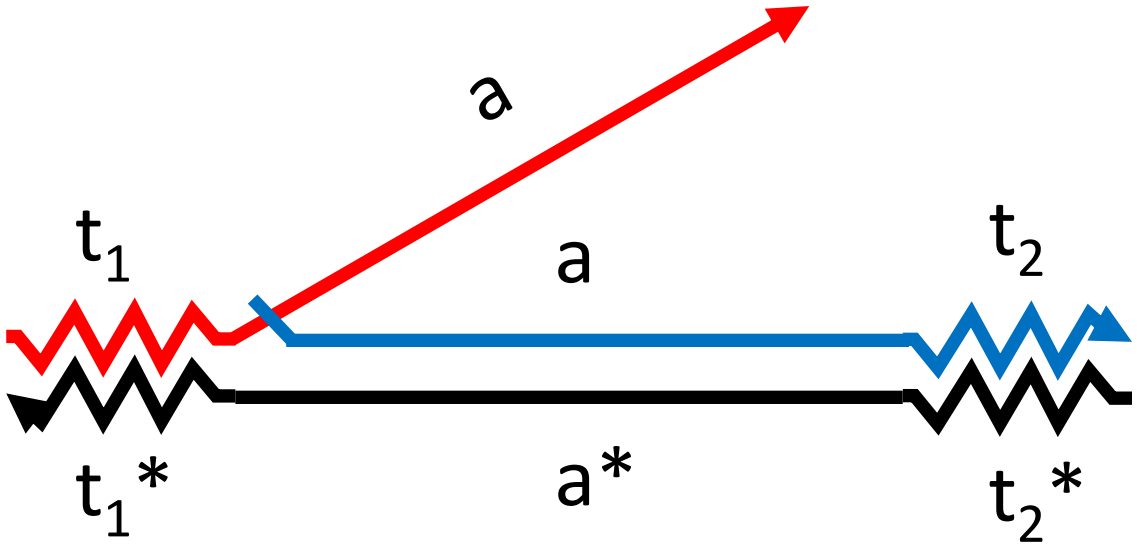
DNA strand displacement

Bind



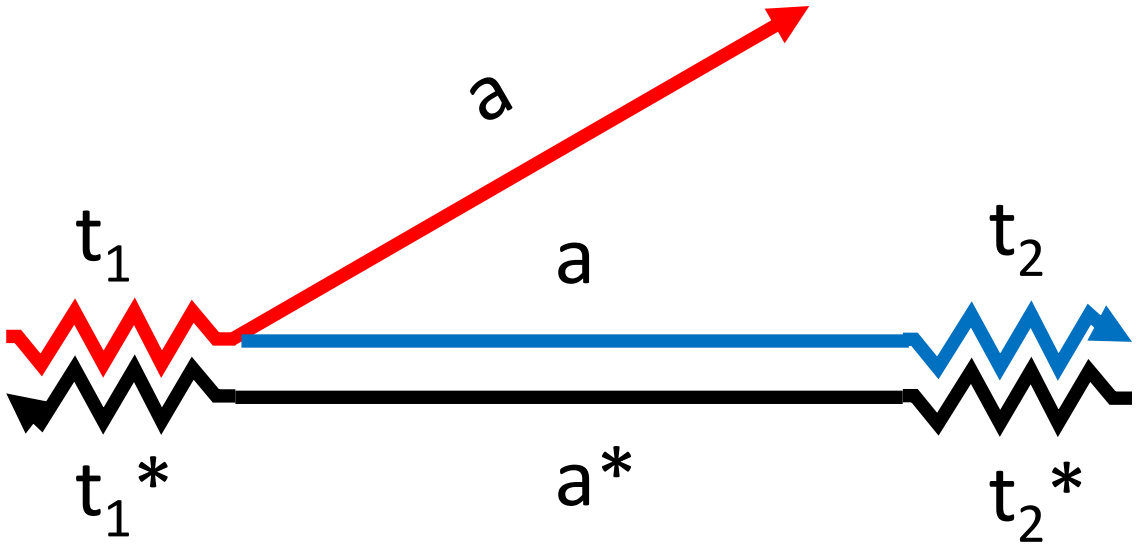
DNA strand displacement

Bind



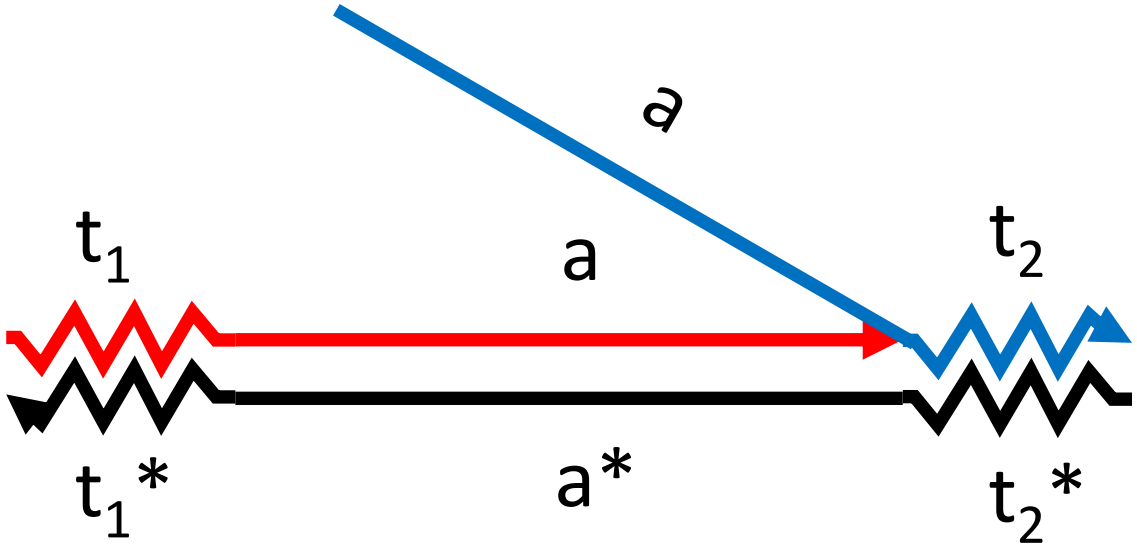
DNA strand displacement

Bind



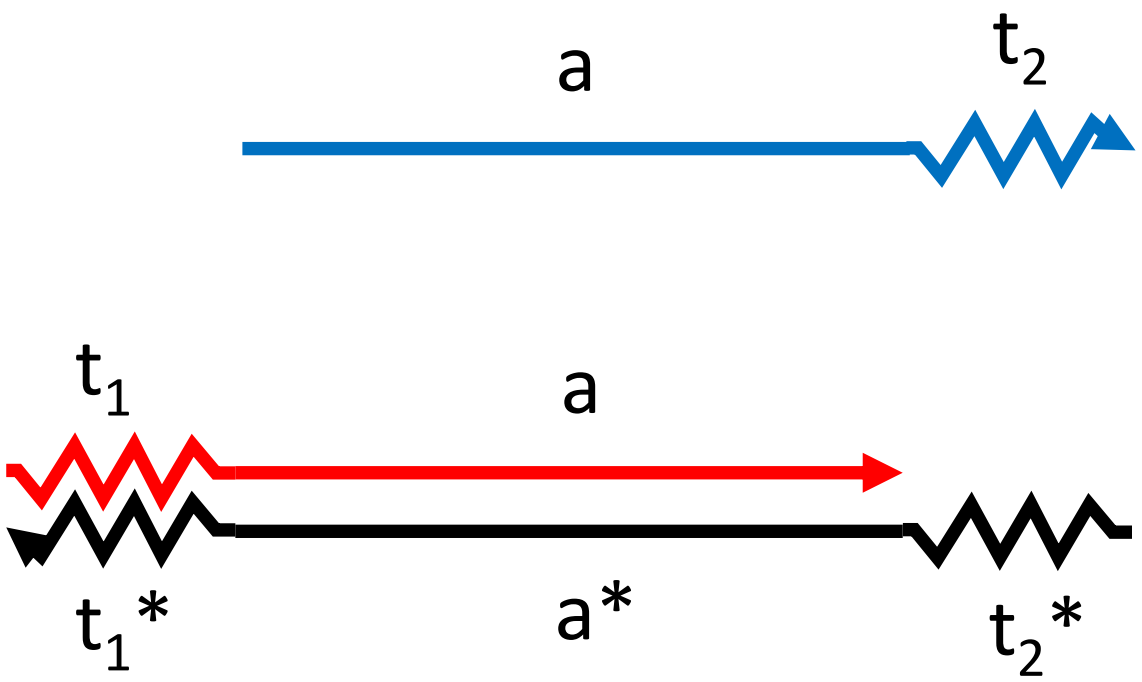
DNA strand displacement

Bind



Displace

DNA strand displacement

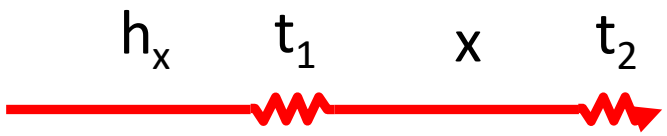


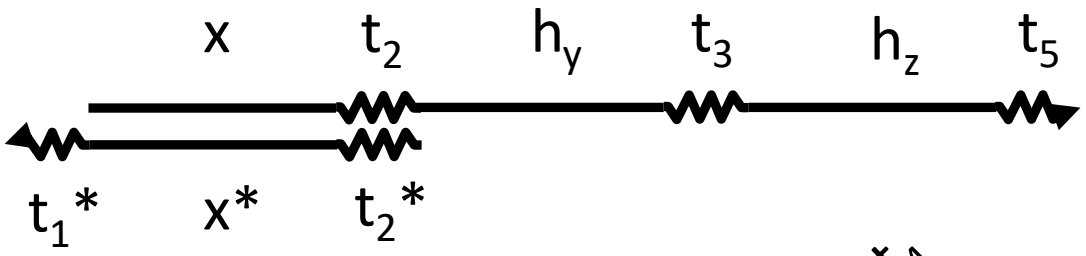
Bind

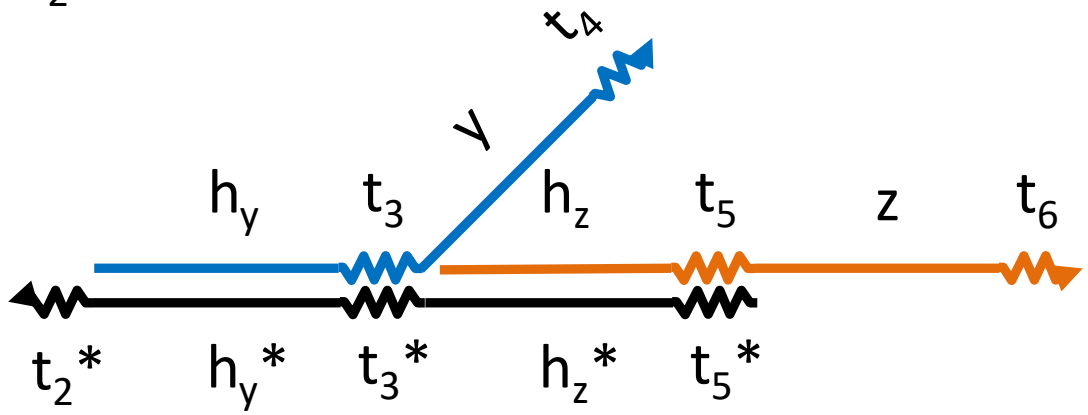
Displace

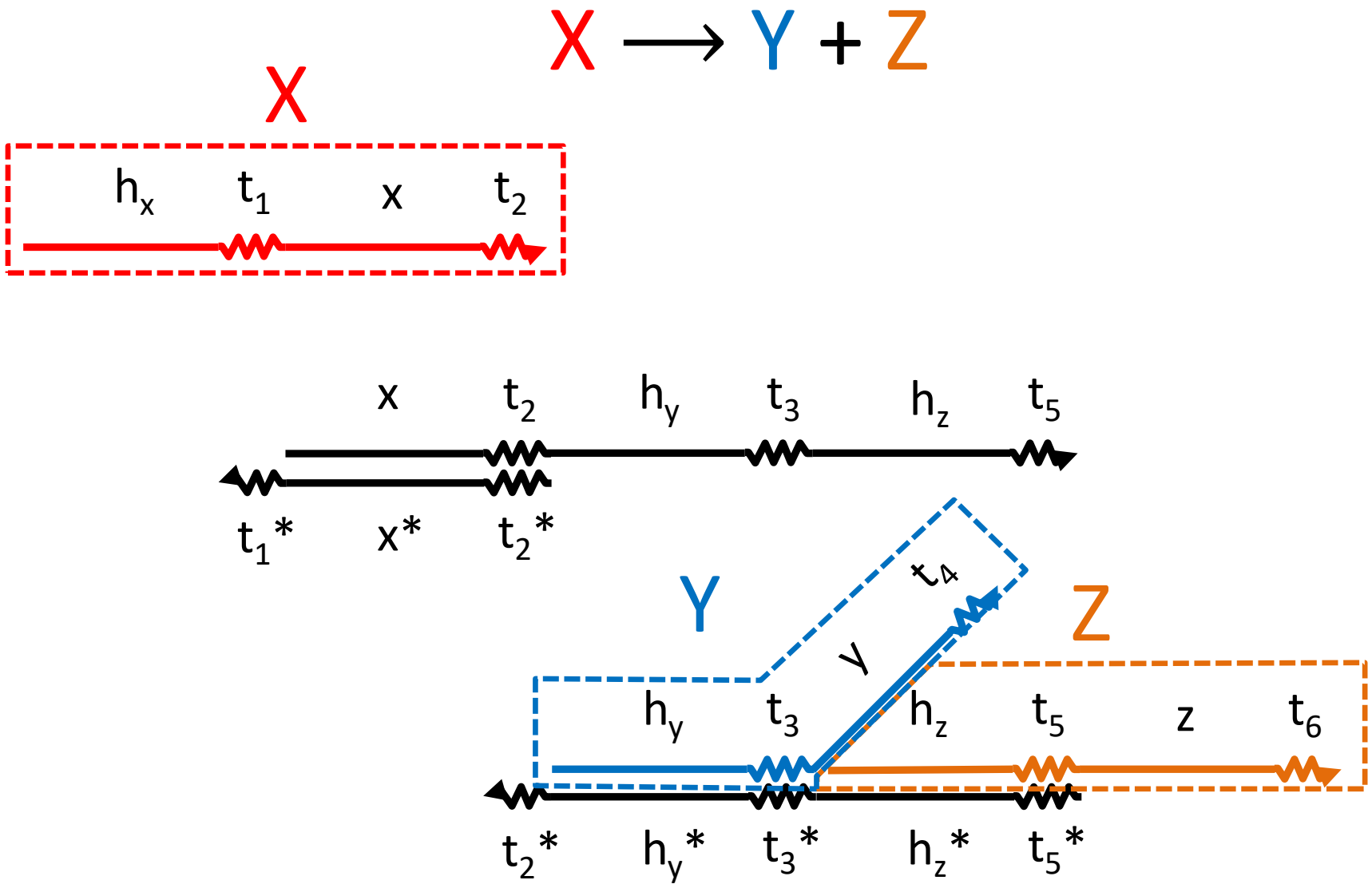
Release

$$X \longrightarrow Y + Z$$

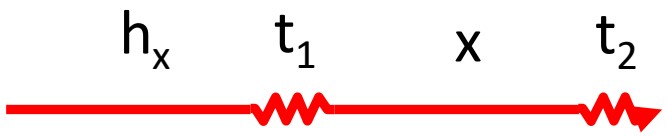


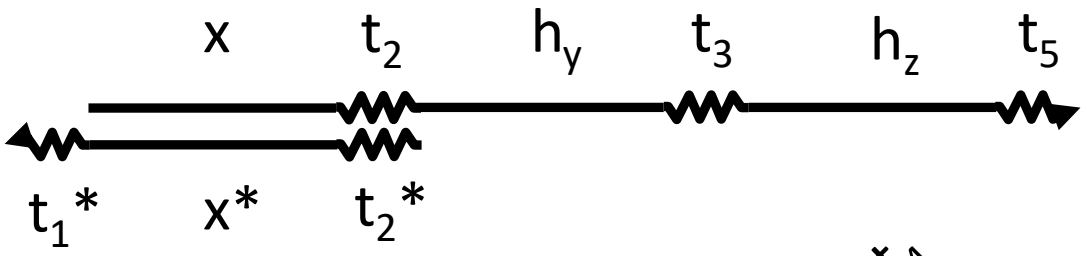


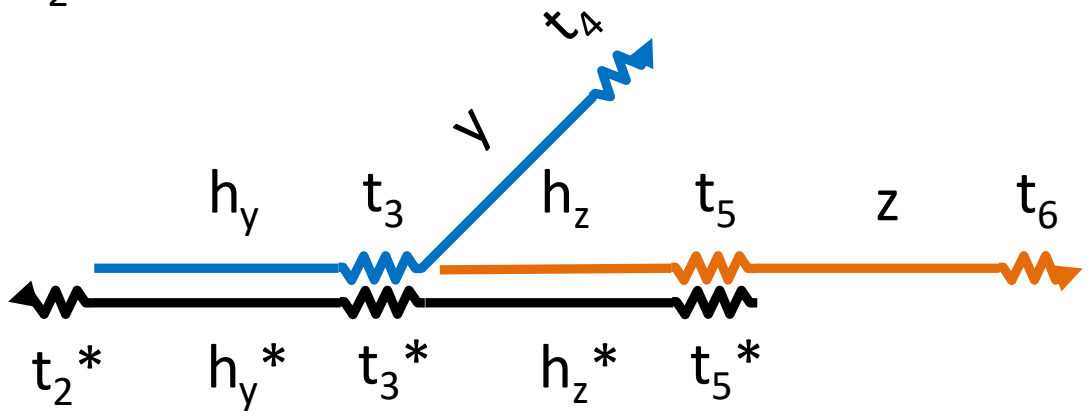



X ⟶ Y + Z
X
h_x t_1 x t_2
x t_2 h_y t_3 h_z t_5
t_1* x* t_2*
Y
Z
t_4
y
h_y t_3 h_z t_5 z t_6
t_2* h_y* t_3* h_z* t_5*

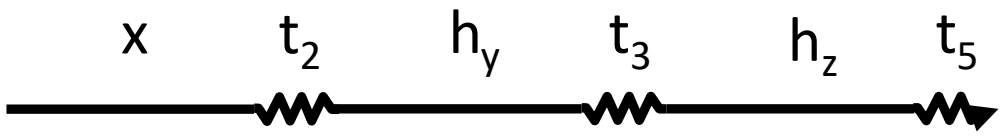
$$X \longrightarrow Y + Z$$

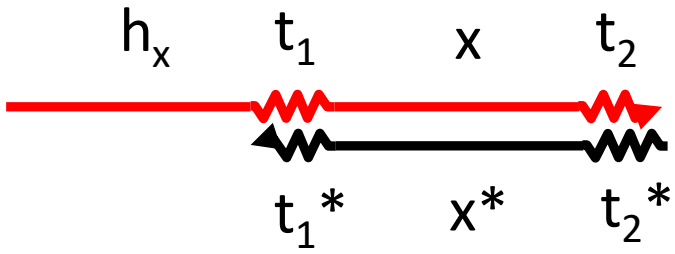


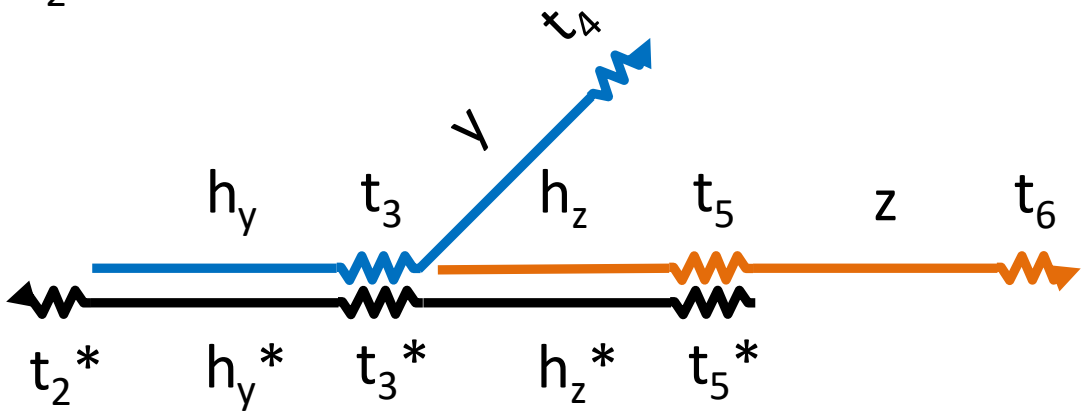




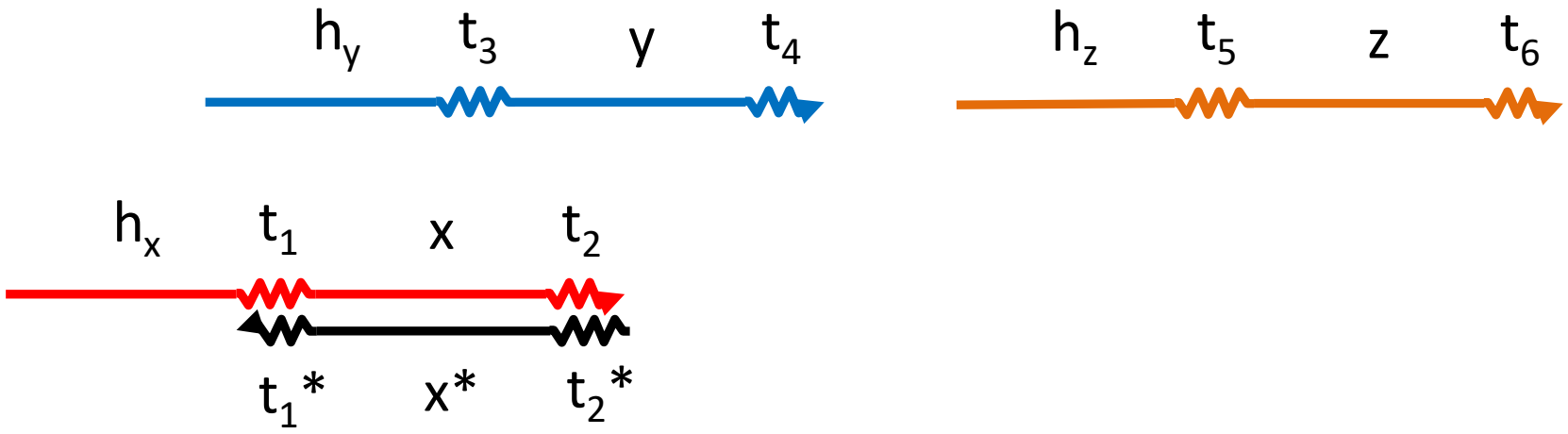
$$X \longrightarrow Y + Z$$

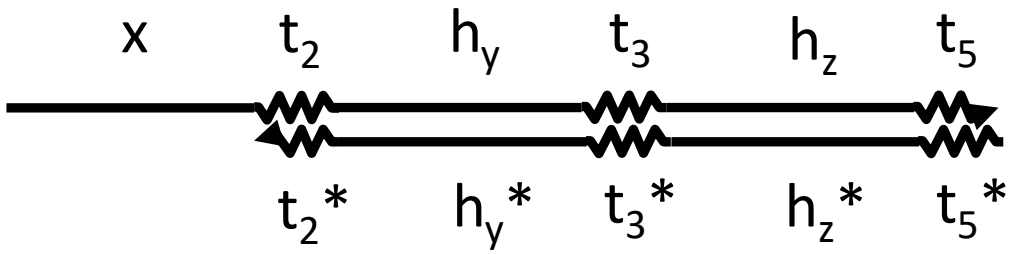




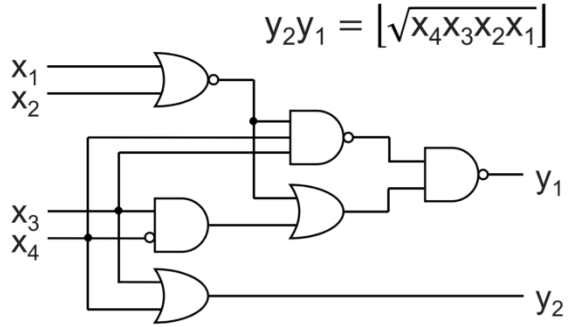


$$X \longrightarrow Y + Z$$



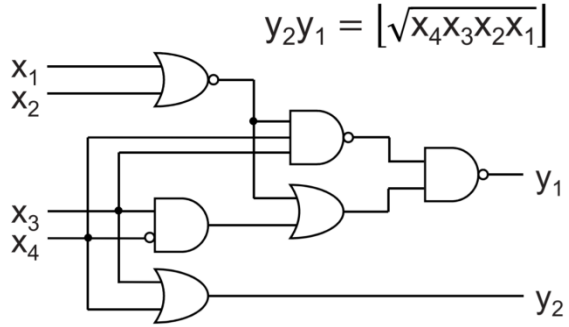


Leak in strand displacement experiments

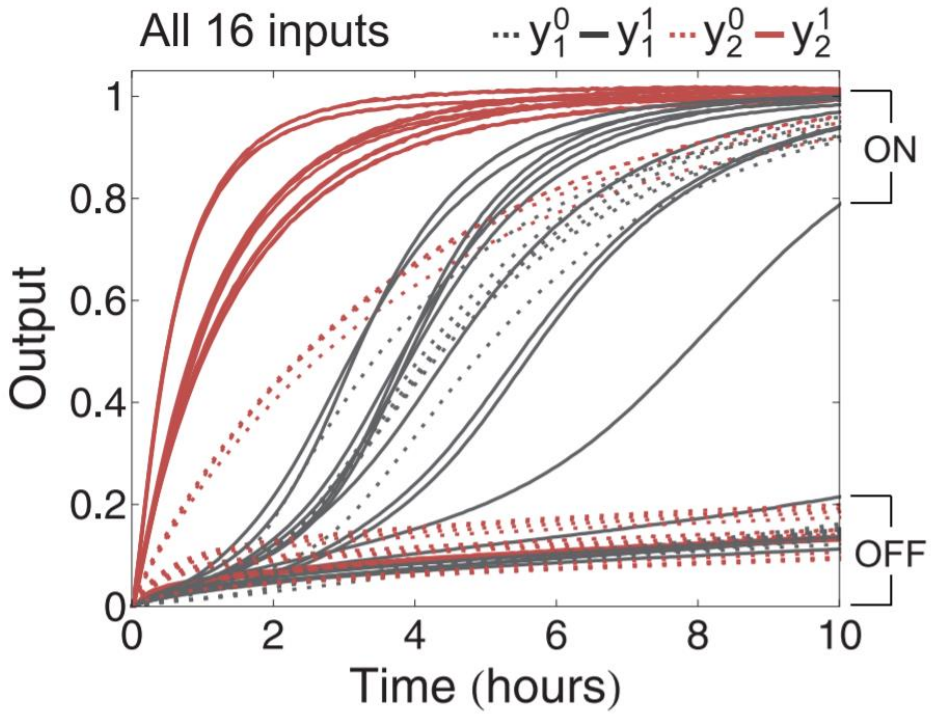


Source:
Lulu Qian, Erik Winfree.
Scaling Up Digital Circuit Computation
Science 332, 2011

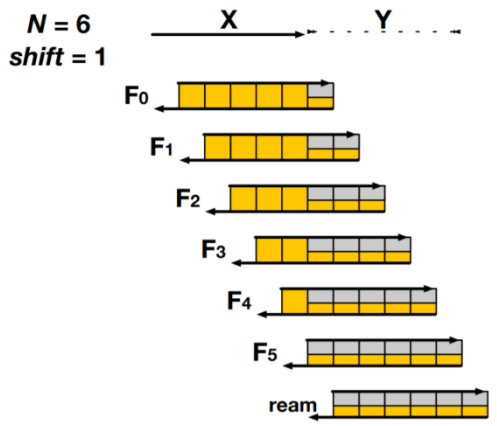
Leak in strand displacement experiments



Source:
Lulu Qian, Erik Winfree.
Scaling Up Digital Circuit Computation
Science 332, 2011

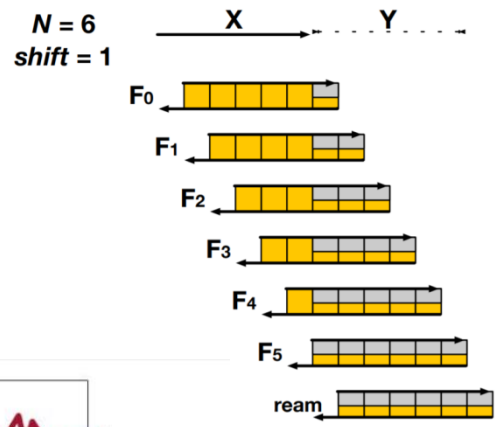


Reducing Leak



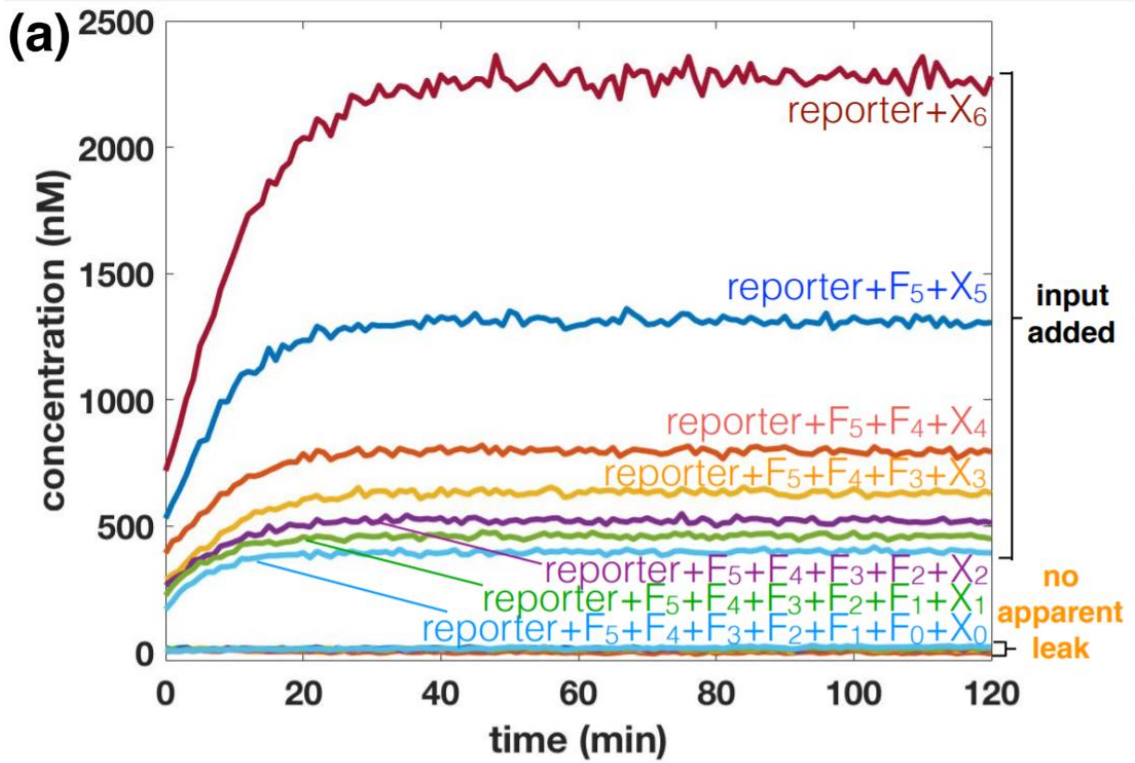
[Boya Wang, Chris Thachuk, Andrew Ellington, David Soloveichik. *The Design Space of Strand Displacement Cascades with Toehold-Size Clamps* DNA Computing Conference, 2017]

Reducing Leak

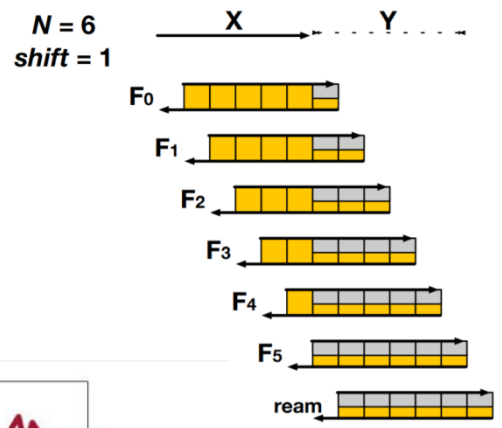


[Boya Wang, Chris Thachuk, Andrew Ellington, David Soloveichik. *The Design Space of Strand Displacement Cascades with Toehold-Size Clamps* DNA Computing Conference, 2017]

Intended:

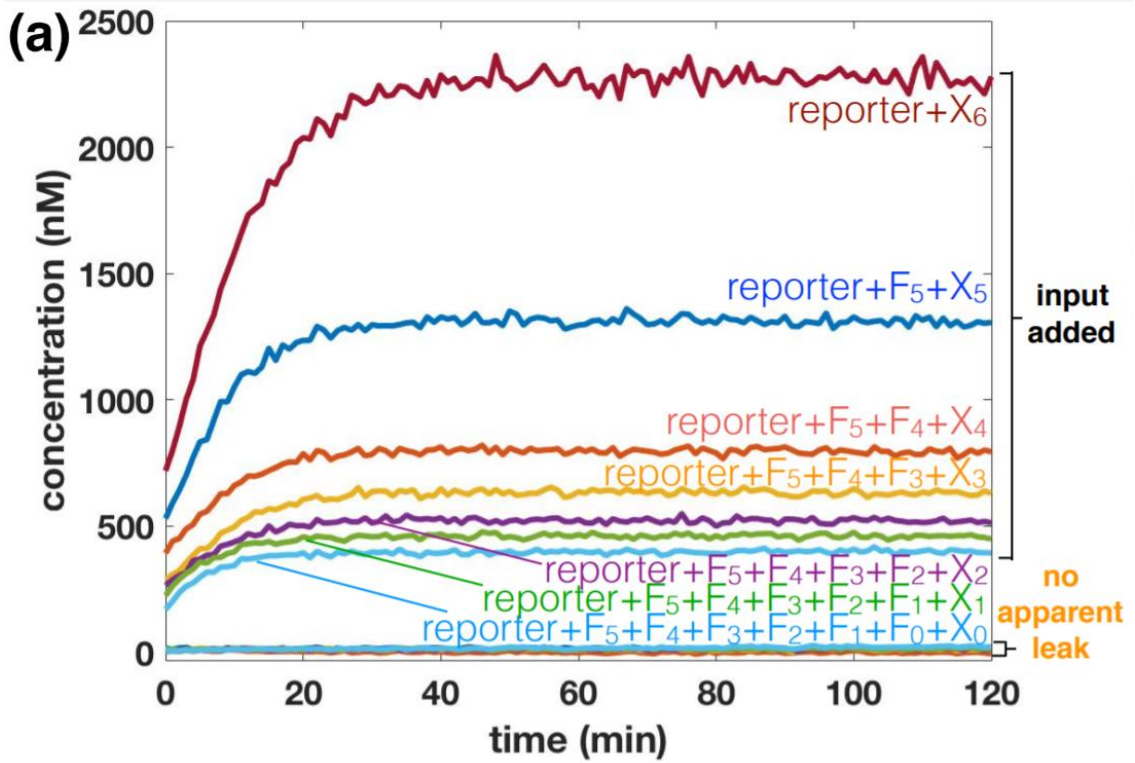


Reducing Leak

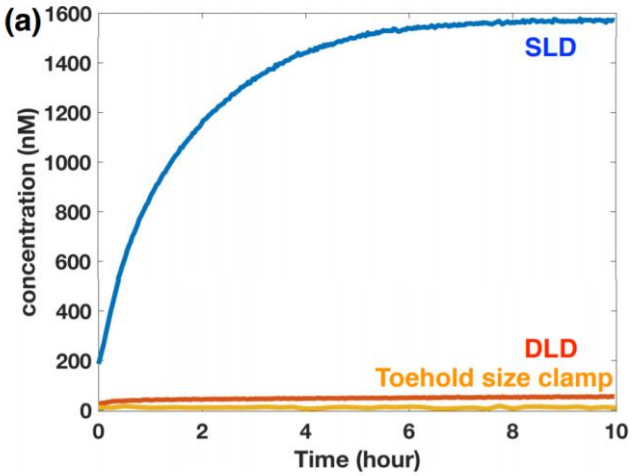


[Boya Wang, Chris Thachuk, Andrew Ellington, David Soloveichik. *The Design Space of Strand Displacement Cascades with Toehold-Size Clamps* DNA Computing Conference, 2017]

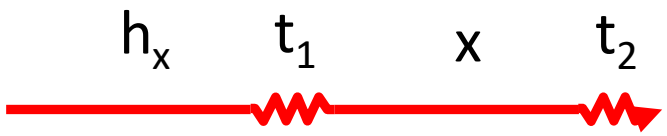
Intended:

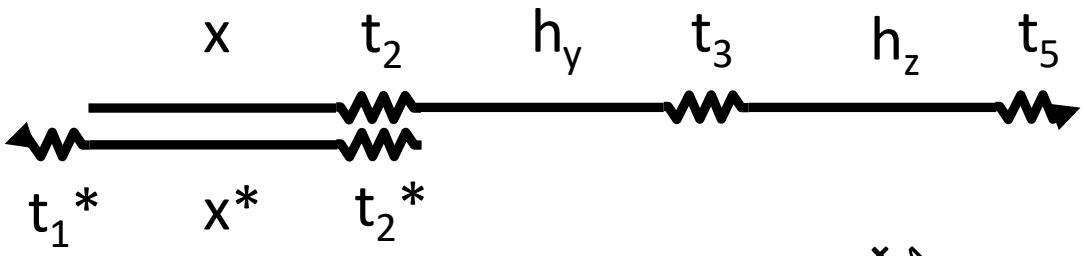


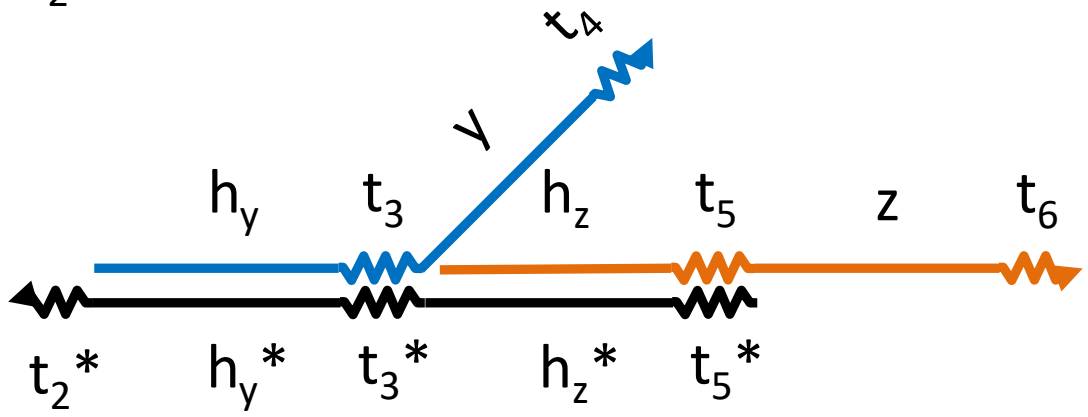
Leak:



$$X \longrightarrow Y + Z$$

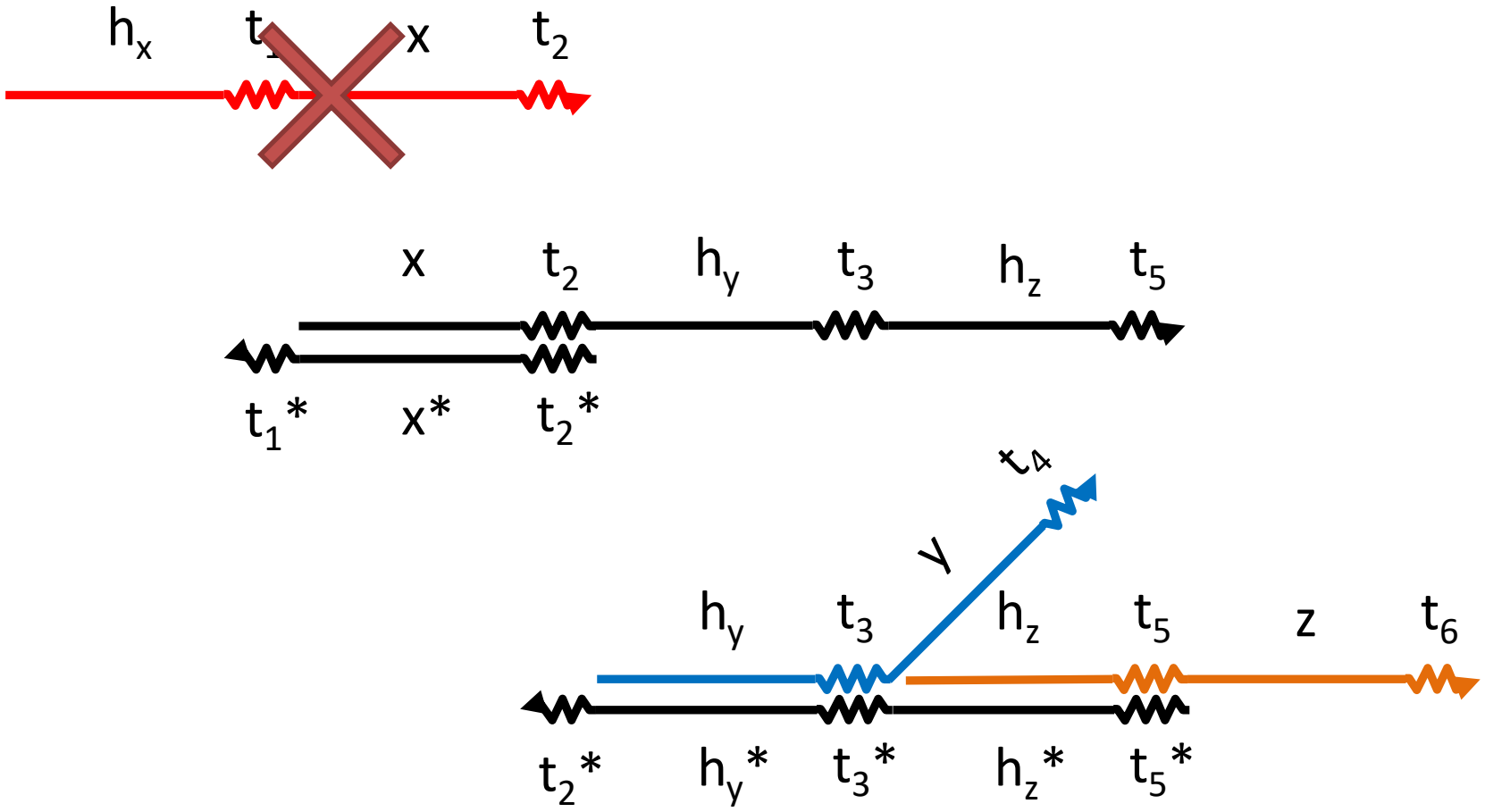






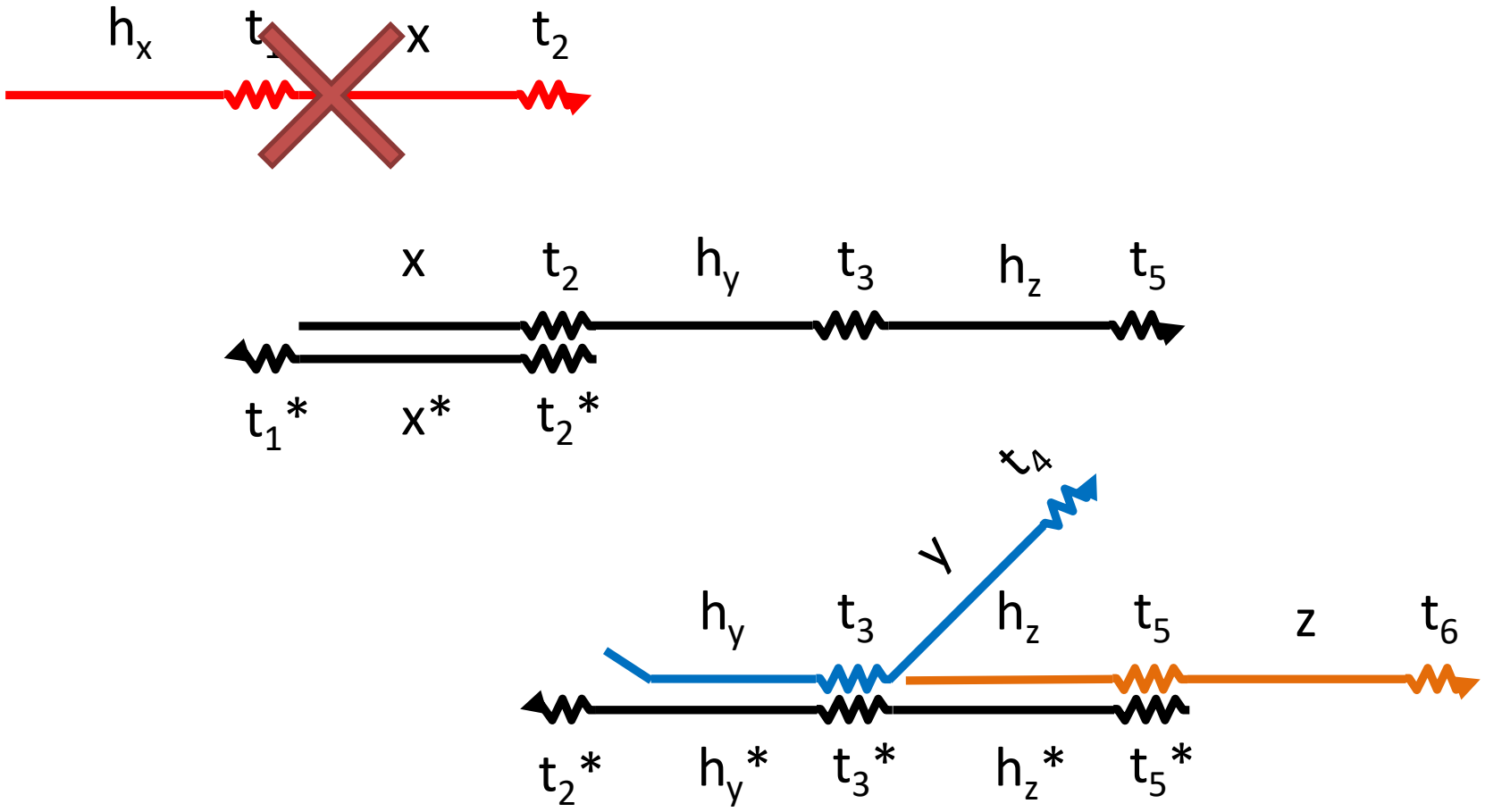
What causes leak “kinetically”?

$$\emptyset \longrightarrow Y + Z$$



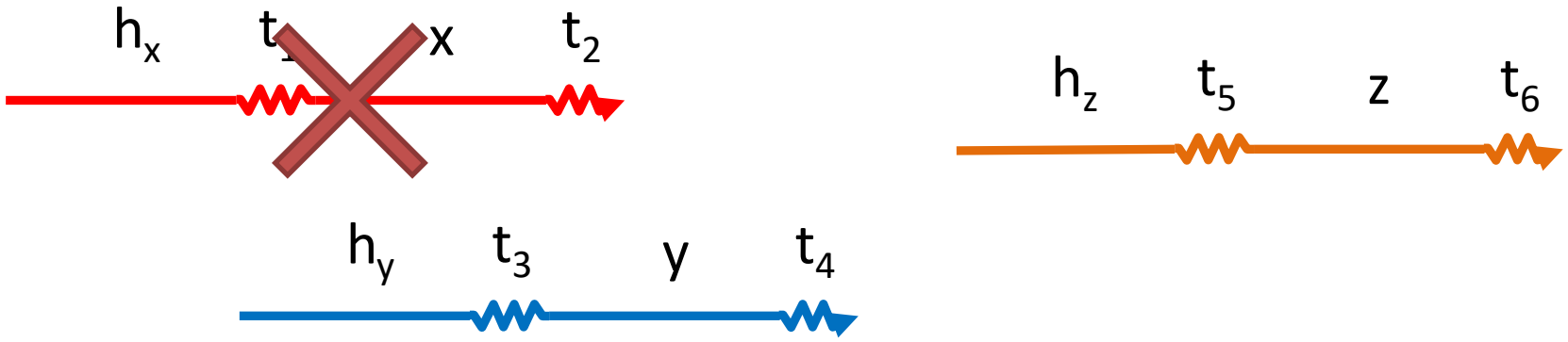
What causes leak “kinetically”?

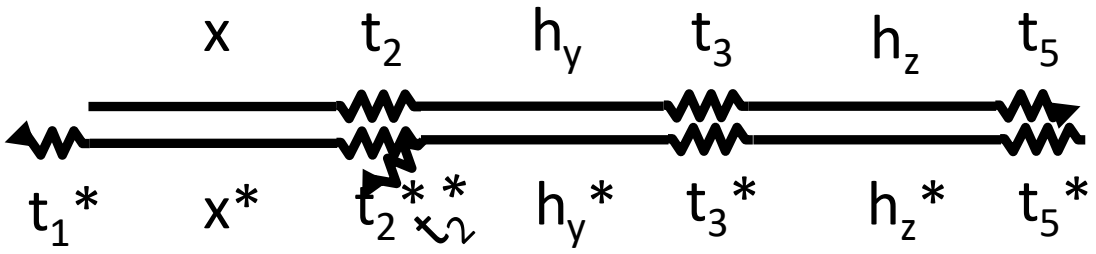
$$\emptyset \longrightarrow Y + Z$$



What causes leak "kinetically"?

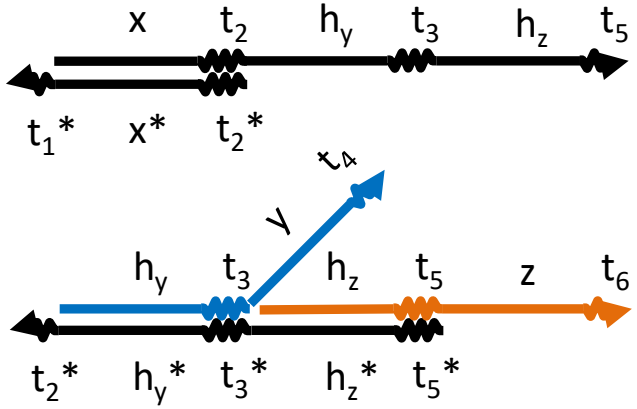
$$\emptyset \longrightarrow Y + Z$$



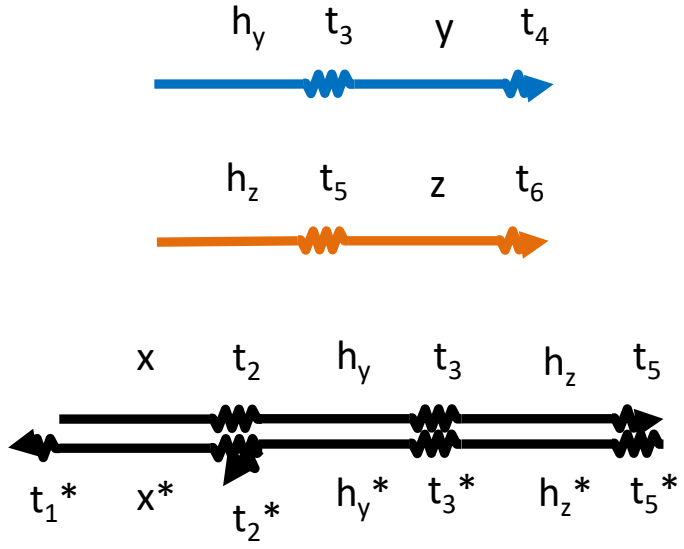


What causes leak “thermodynamically”?

Before:

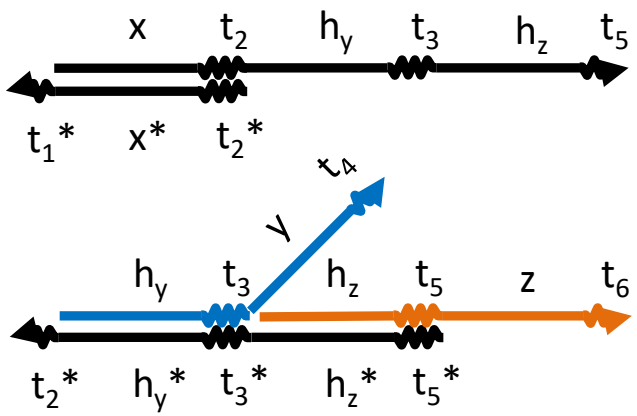


After:



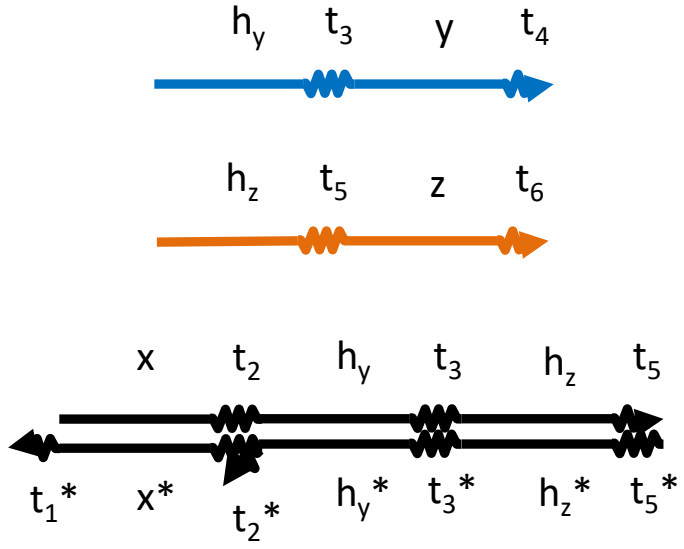
What causes leak “thermodynamically”?

Before:



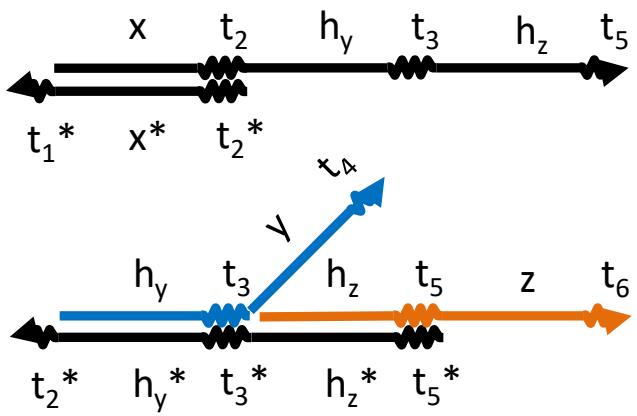
slow

After:



What causes leak “thermodynamically”?

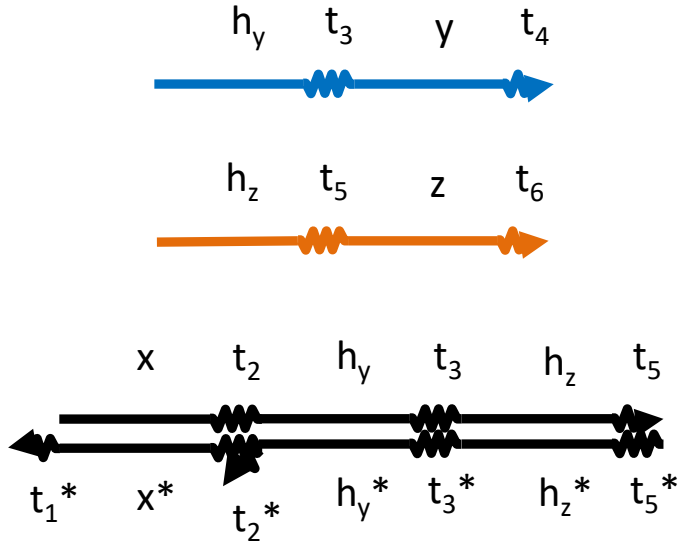
Before:



slow

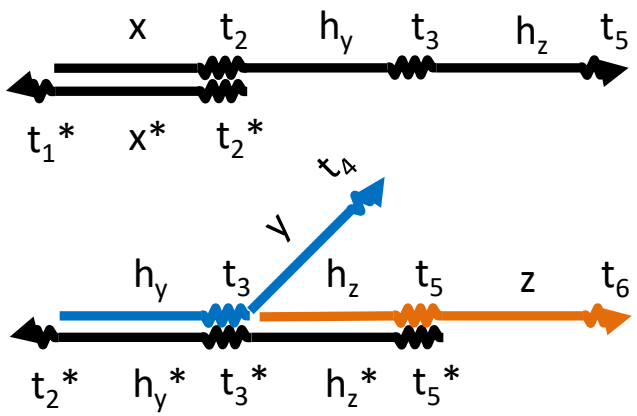
very slow

After:



What causes leak “thermodynamically”?

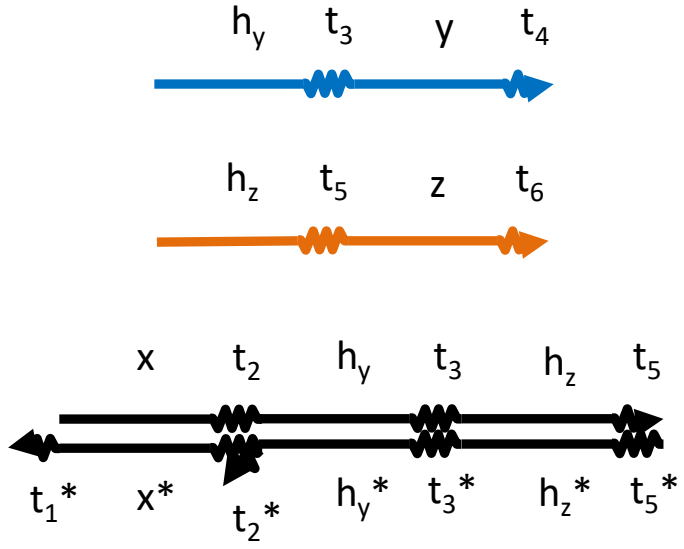
Before:



slow

very slow

After:



less favorable

more favorable

Need a *kinetic* binding network model

Need a *kinetic* binding network model

- Can we design pathways that maintain local stability?

Need a *kinetic* binding network model

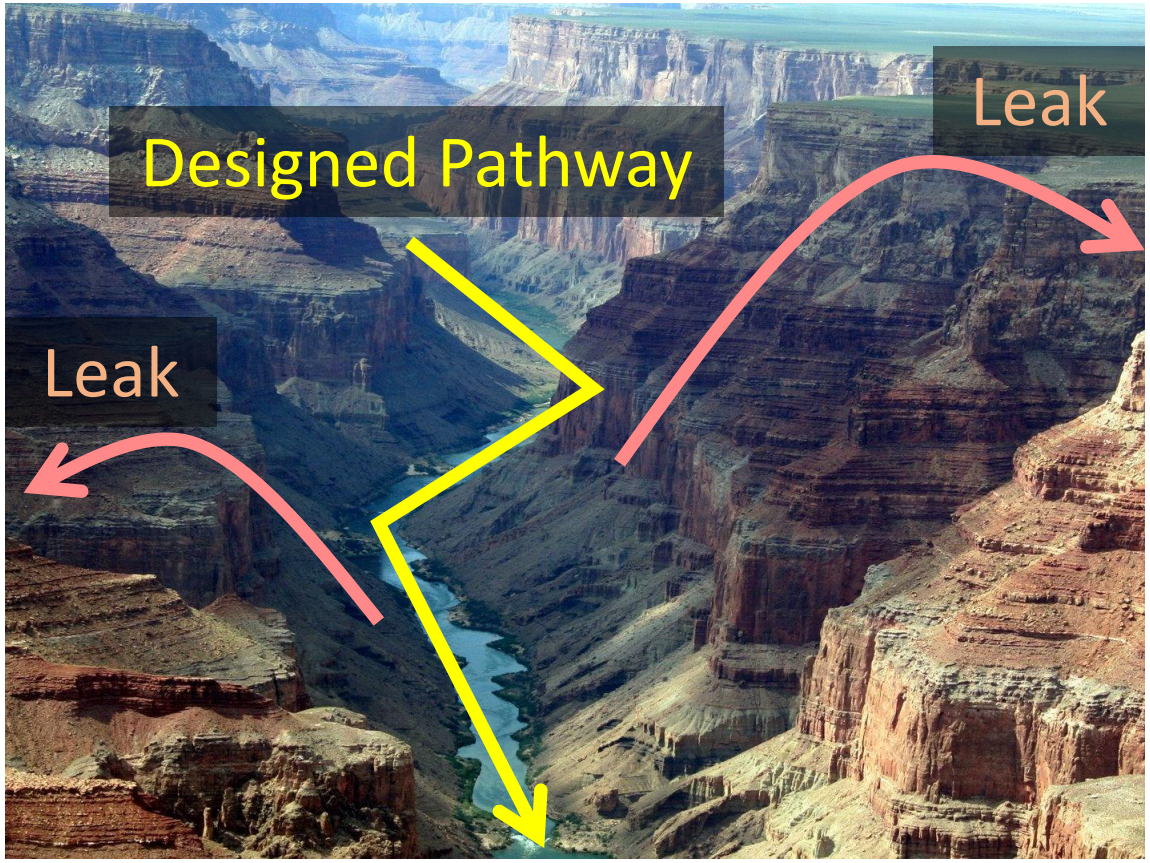
- Can we design pathways that maintain local stability?

Need a *kinetic* binding network model

- Can we design pathways that maintain local stability?

Need a *kinetic* binding network model

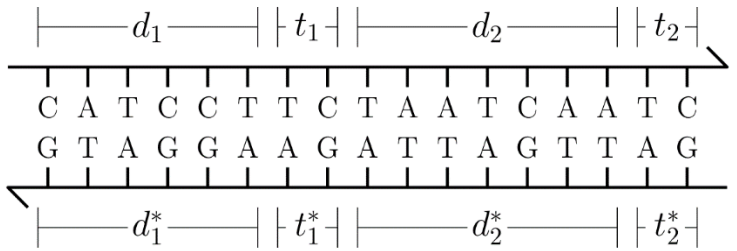
- Can we design pathways that maintain local stability?



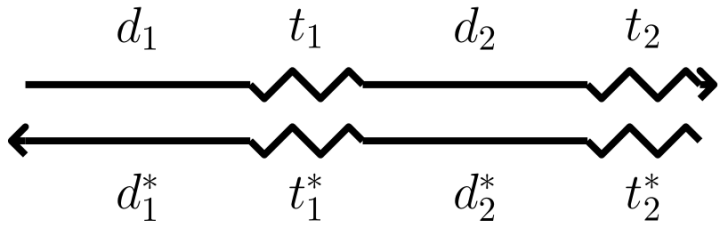
Levels of Abstraction

DNA

Base Pairs



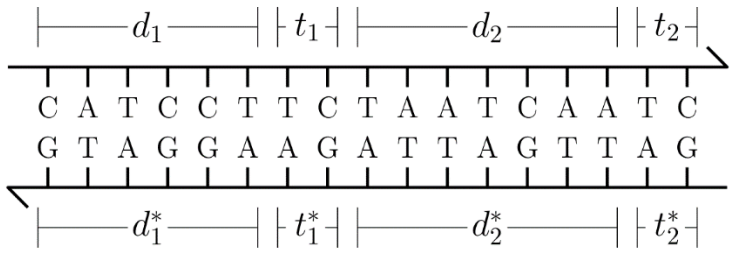
Strands



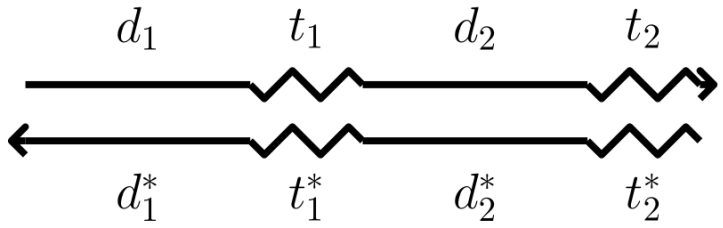
Levels of Abstraction

DNA

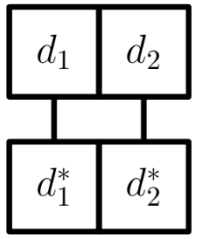
Base Pairs



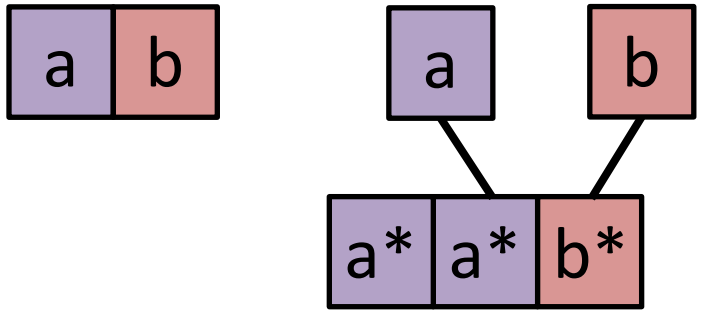
Strands



Thermodynamic Binding Network



Thermodynamic Binding Networks



Geometry-Free Model:

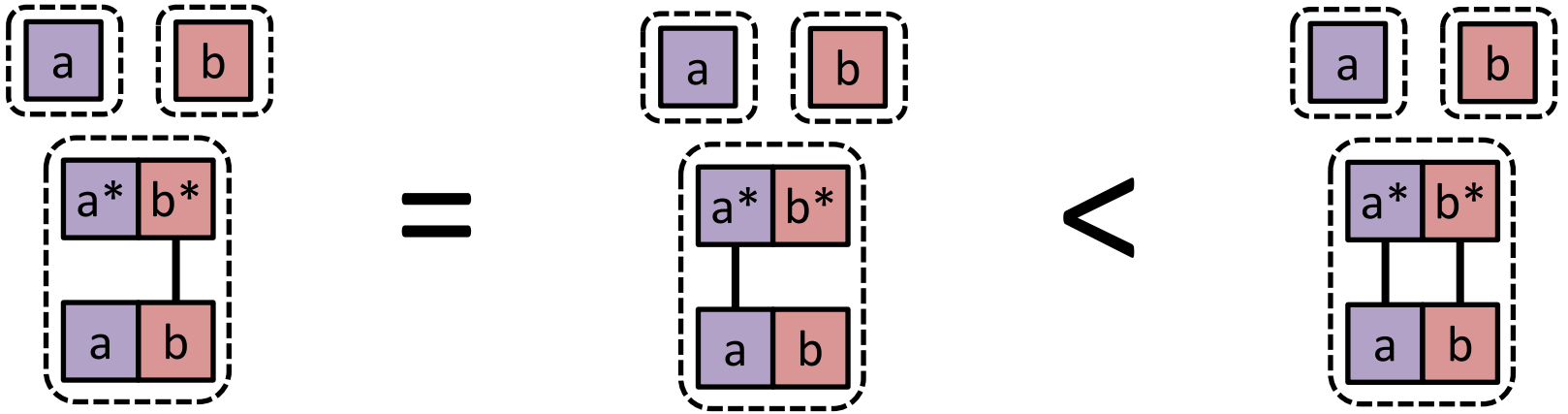
The domains within a monomer are unordered

Monomer = collection of domains

Configuration = how monomers are bound

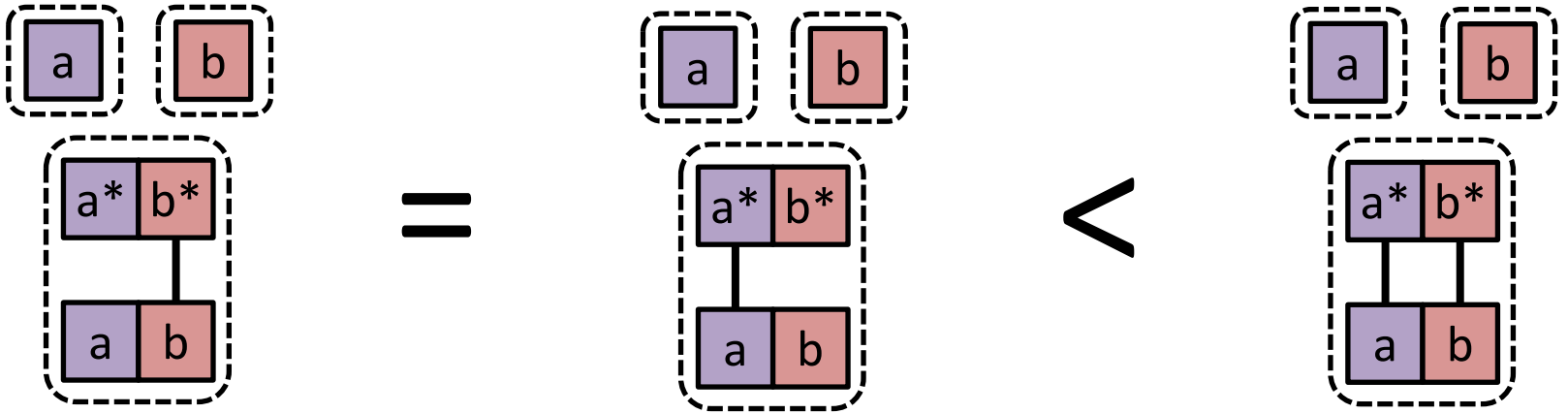
Energetic favorability: Bonds and complexes

all else equal,
more bonds
= more favorable

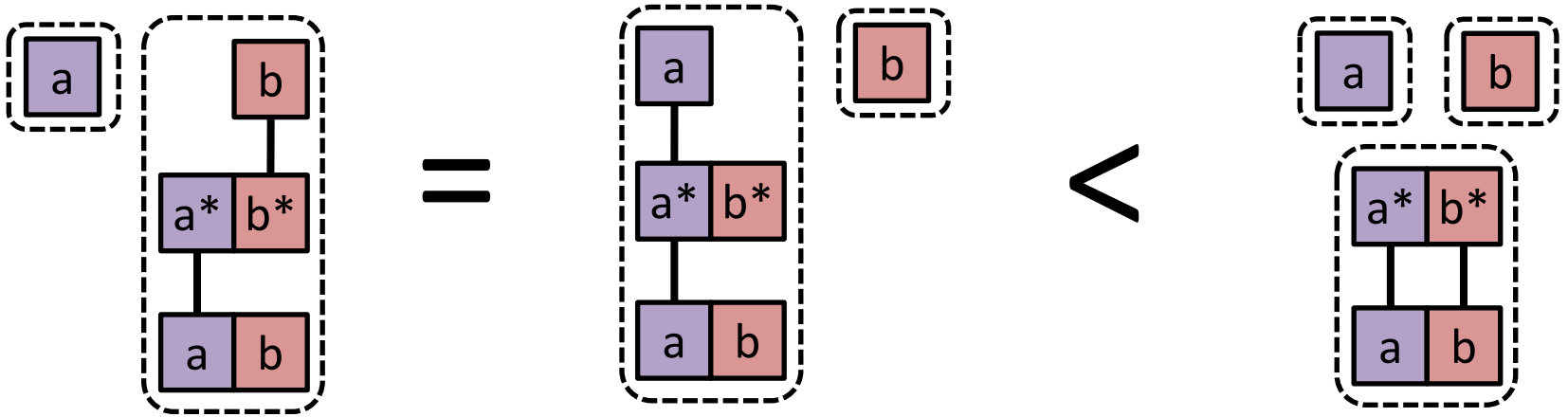


Energetic favorability: Bonds and complexes

all else equal,
more bonds
= more favorable



all else equal,
more complexes
= more favorable



Tradeoff between #bonds and #complexes

Tradeoff between #bonds and #complexes

- in general, there's some weight parameter *w*:

  energy = w*#bonds + #complexes

Tradeoff between #bonds and #complexes

- in general, there's some weight parameter *w*:

  energy = w*#bonds + #complexes

  (*physics notation*: $\Delta G = \Delta H - T \cdot \Delta S$)

Tradeoff between #bonds and #complexes

- in general, there's some weight parameter *w*:

  energy = w*#bonds + #complexes

  (*physics notation*: $\Delta G = \Delta H - T \cdot \Delta S$)

- We often consider a natural limiting case:
  - favoring # bonds <u>infinitely</u> over #complexes
  - require maximal #bonds formed; use #complexes only as tiebreaker

Tradeoff between #bonds and #complexes

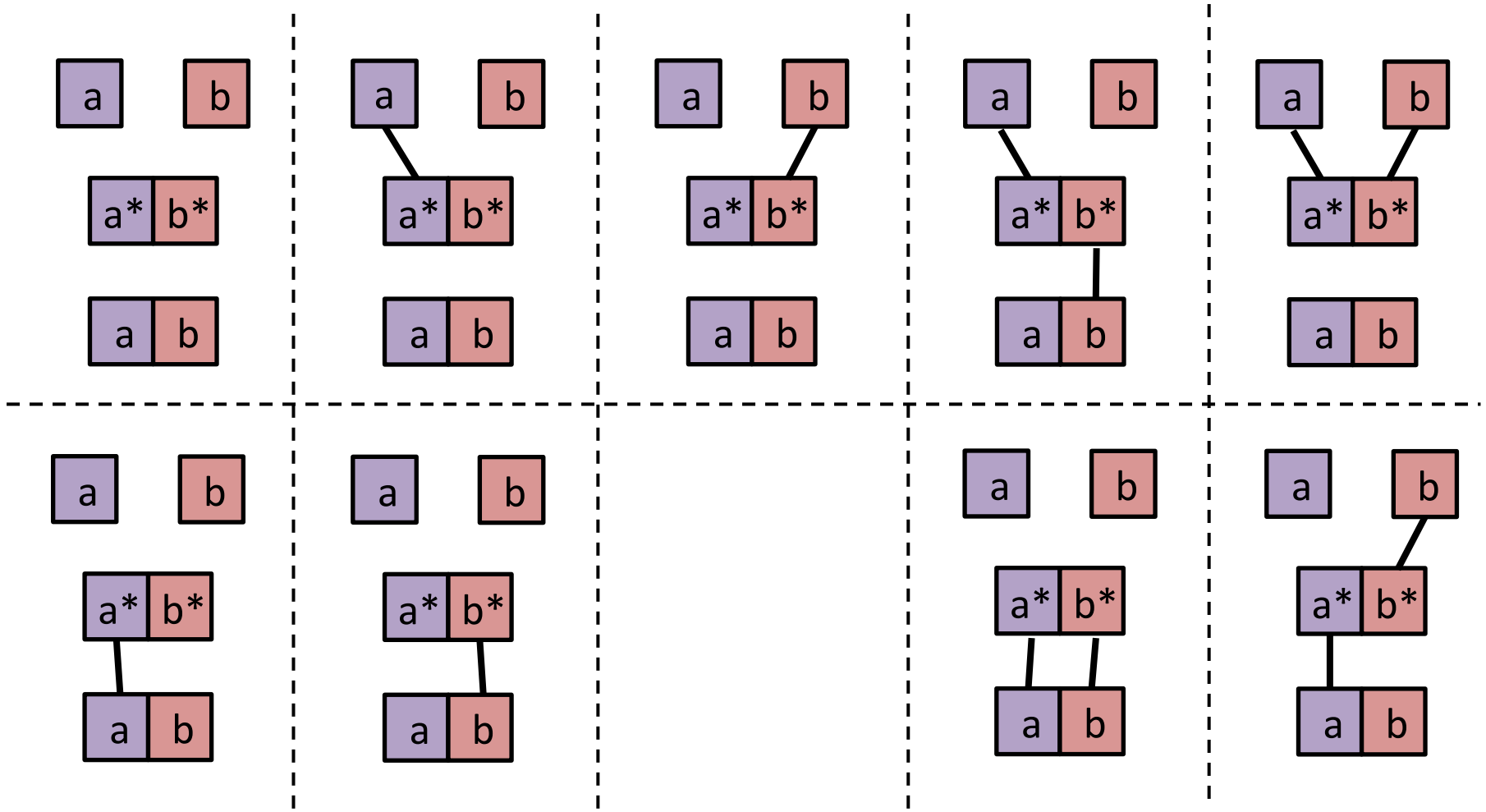
- in general, there's some weight parameter *w*:

  energy = w*#bonds + #complexes

  (*physics notation*: $\Delta G = \Delta H - T \cdot \Delta S$)

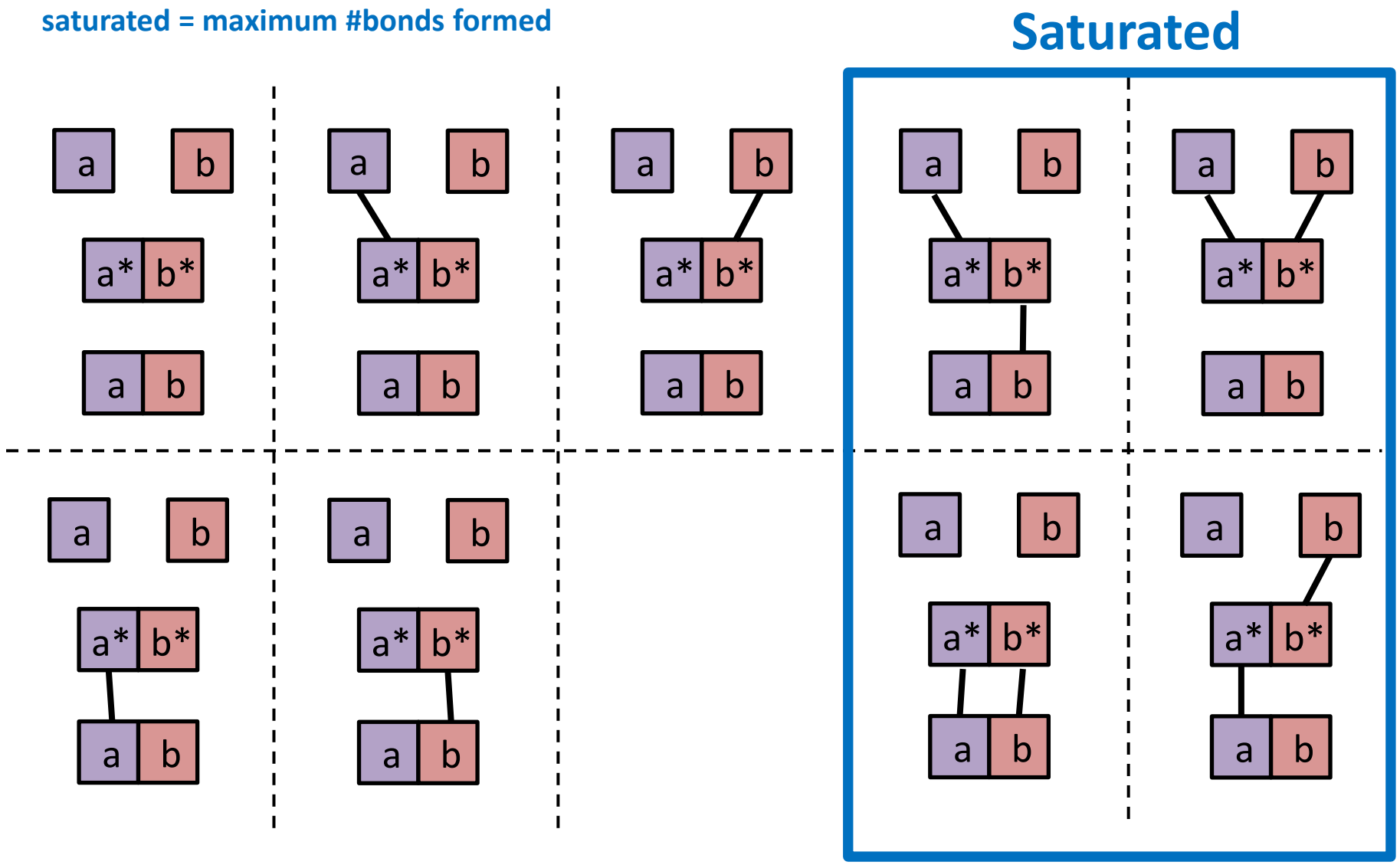
- We often consider a natural limiting case:
  - favoring # bonds infinitely over #complexes
  - require maximal #bonds formed; use #complexes only as tiebreaker
  - Corresponds to bonds that are so strong they cannot spontaneously dissociate, but can exchange with each other to find configurations with more complexes

Thermodynamic Binding Networks



Thermodynamic Binding Networks

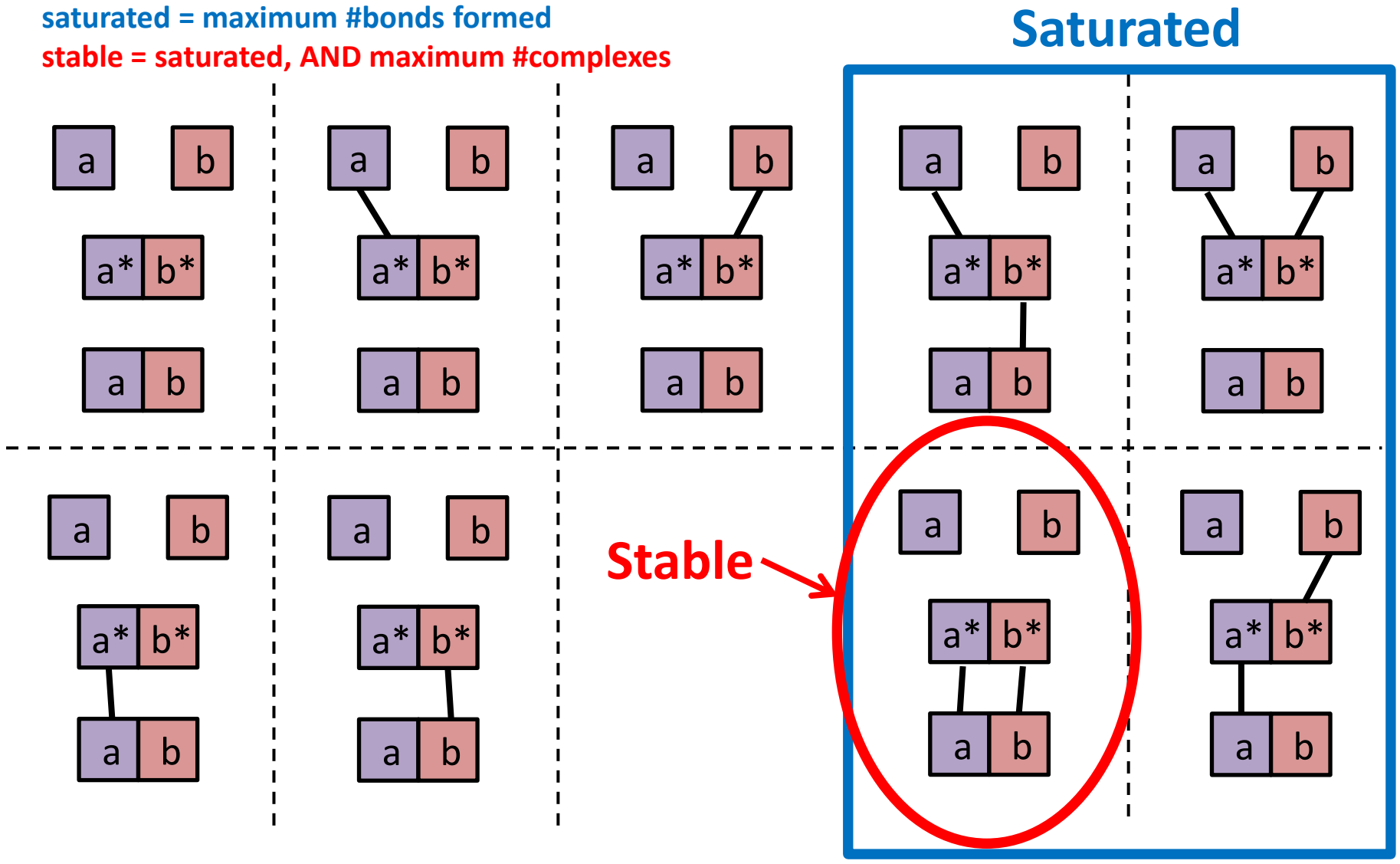
saturated = maximum #bonds formed



Thermodynamic Binding Networks

saturated = maximum #bonds formed

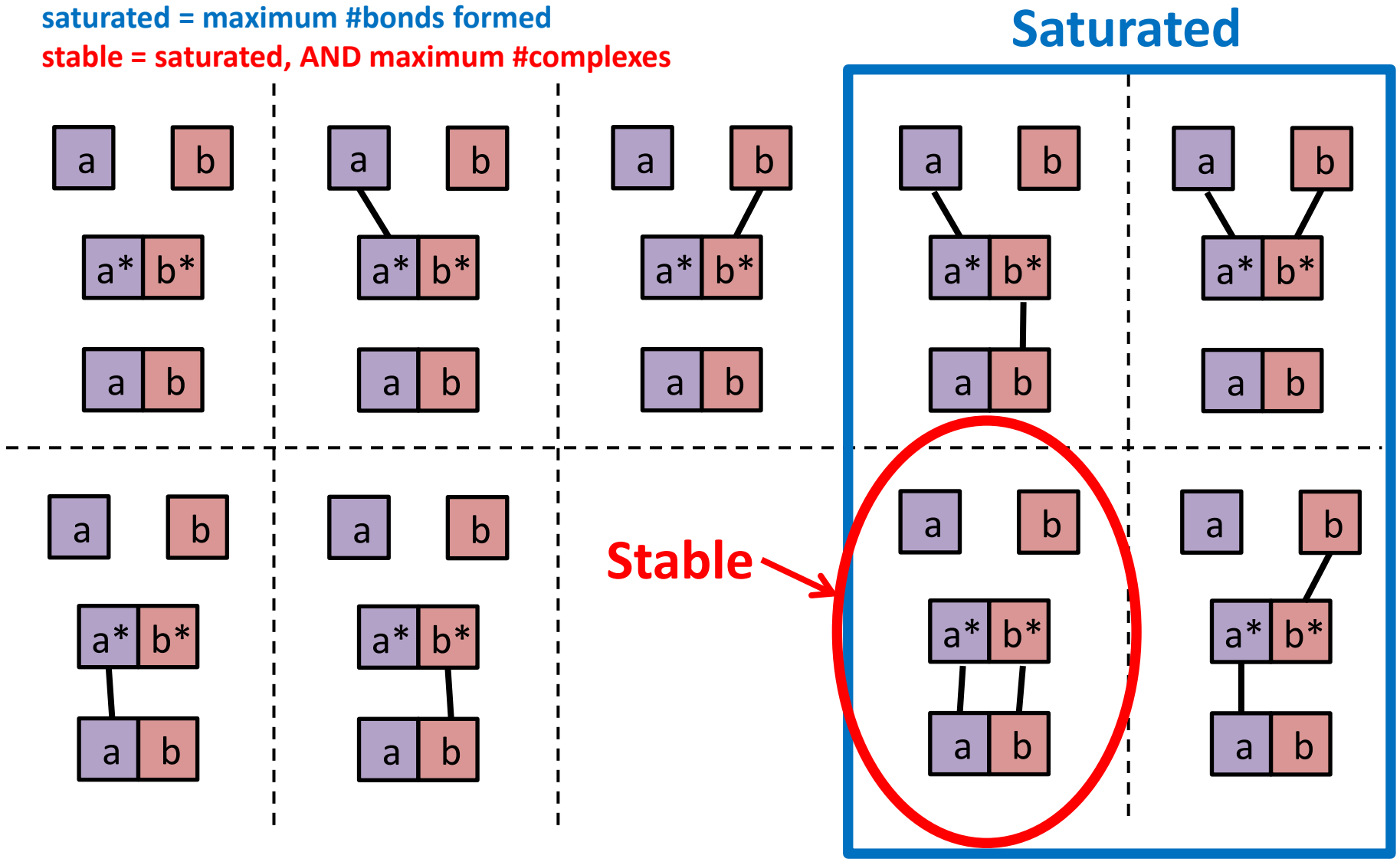
stable = saturated, AND maximum #complexes



Thermodynamic Binding Networks

saturated = maximum #bonds formed

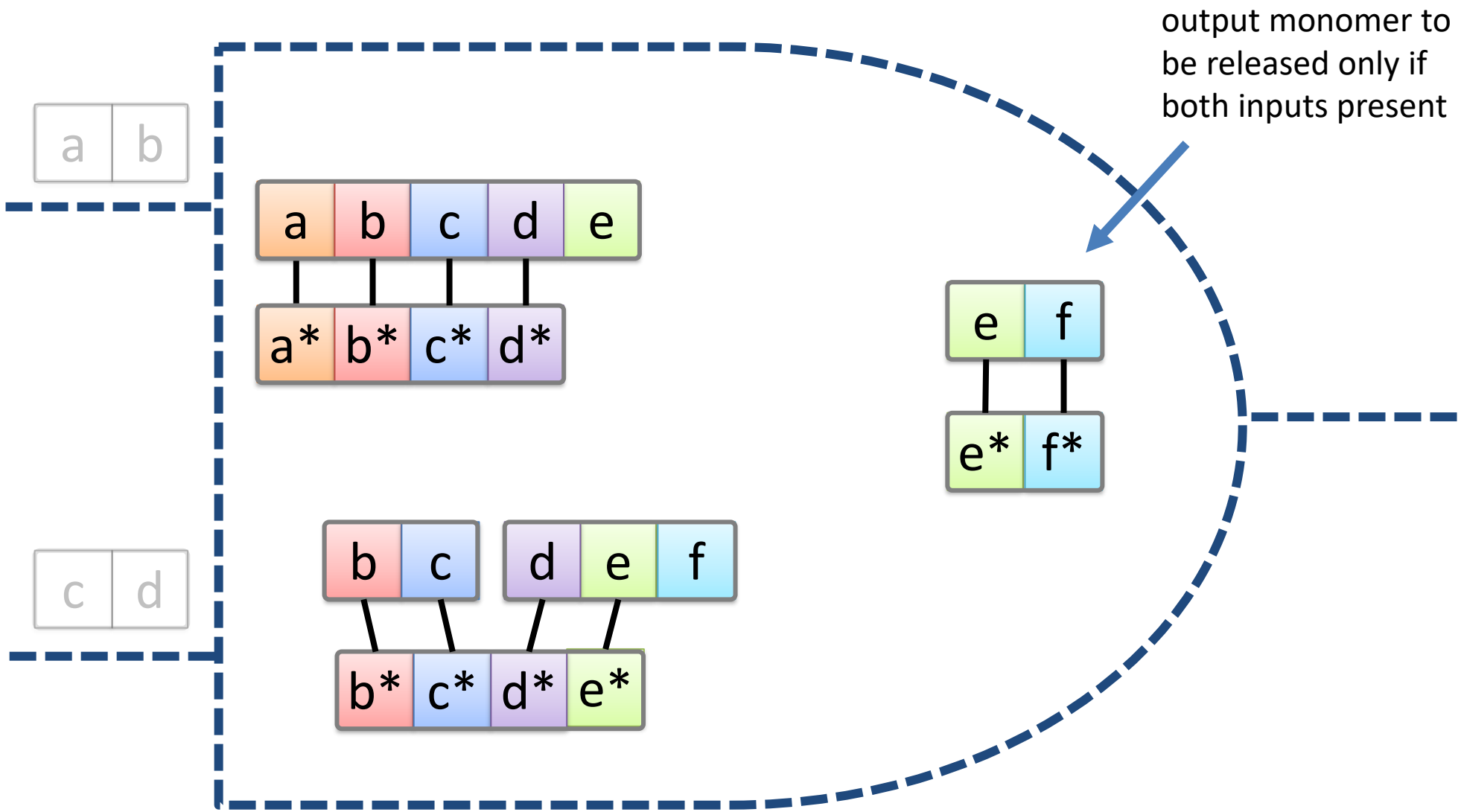
stable = saturated, AND maximum #complexes



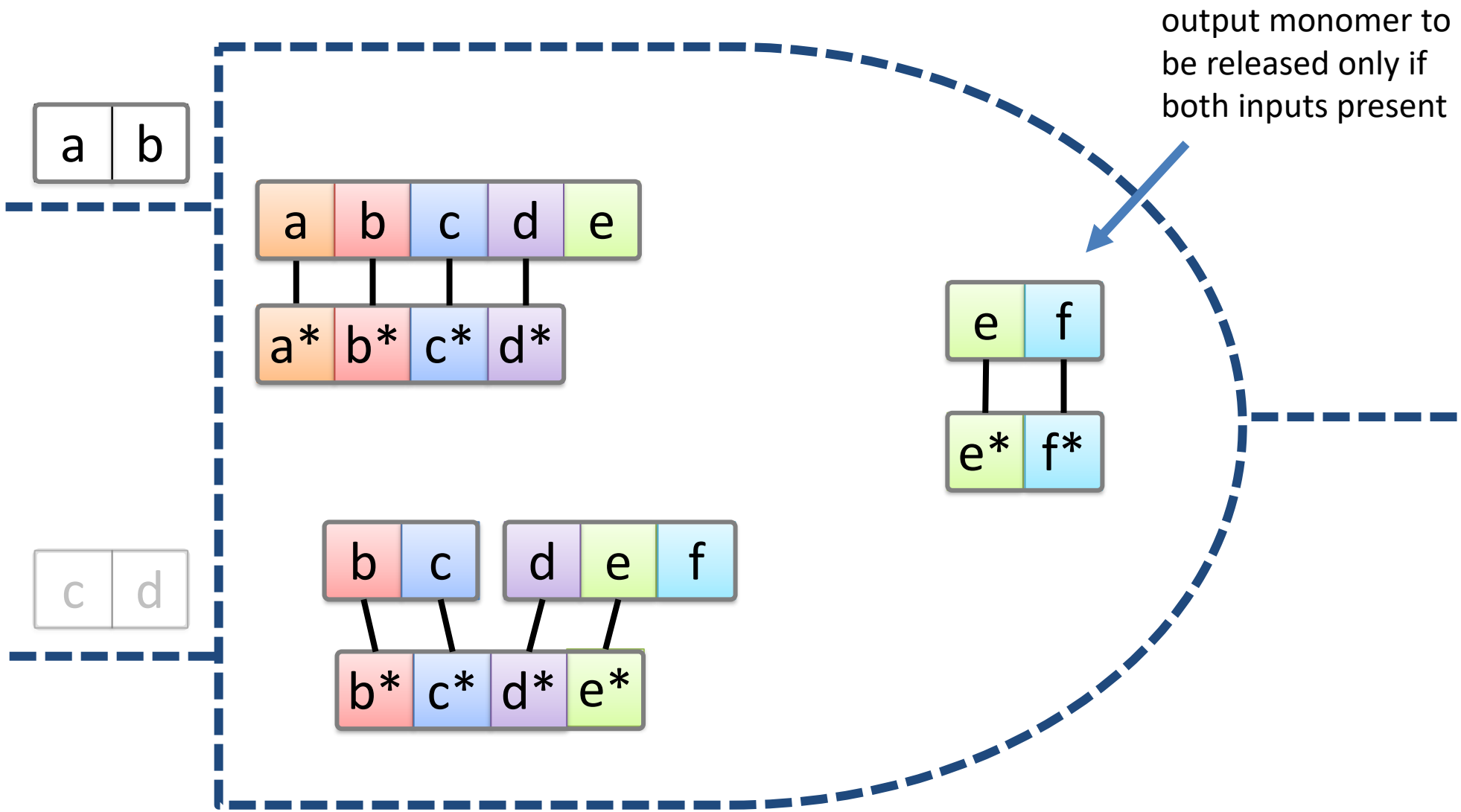
If we're careful to make starred binding sites limiting, then *saturated = all starred sites are bound*

Computing via Thermodynamic Equilibrium

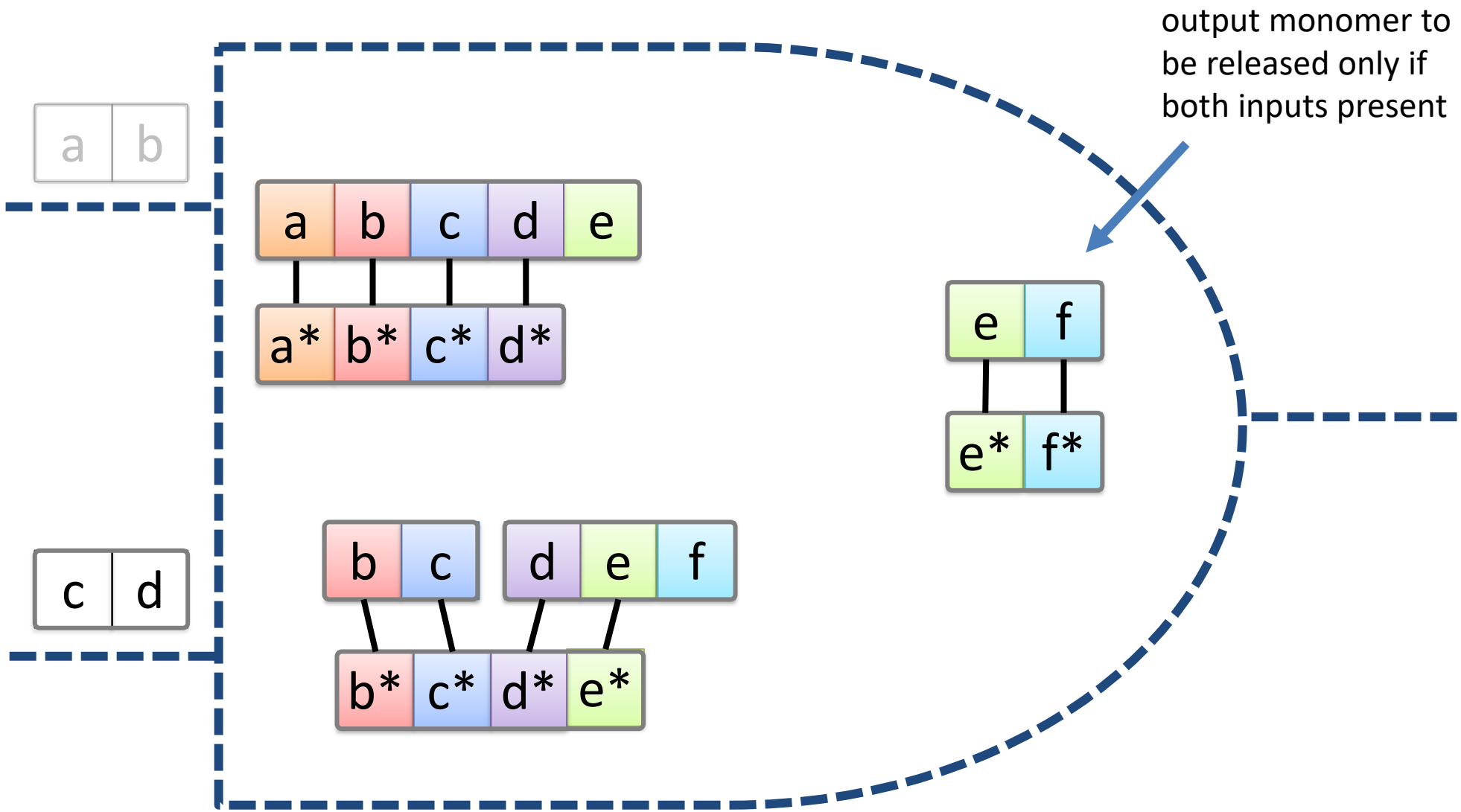
AND gate


output monomer to be released only if both inputs present
a
b
c
d
a
b
c
d
e
a*
b*
c*
d*
e
f
e*
f*
b
c
d
e
f
b*
c*
d*
e*

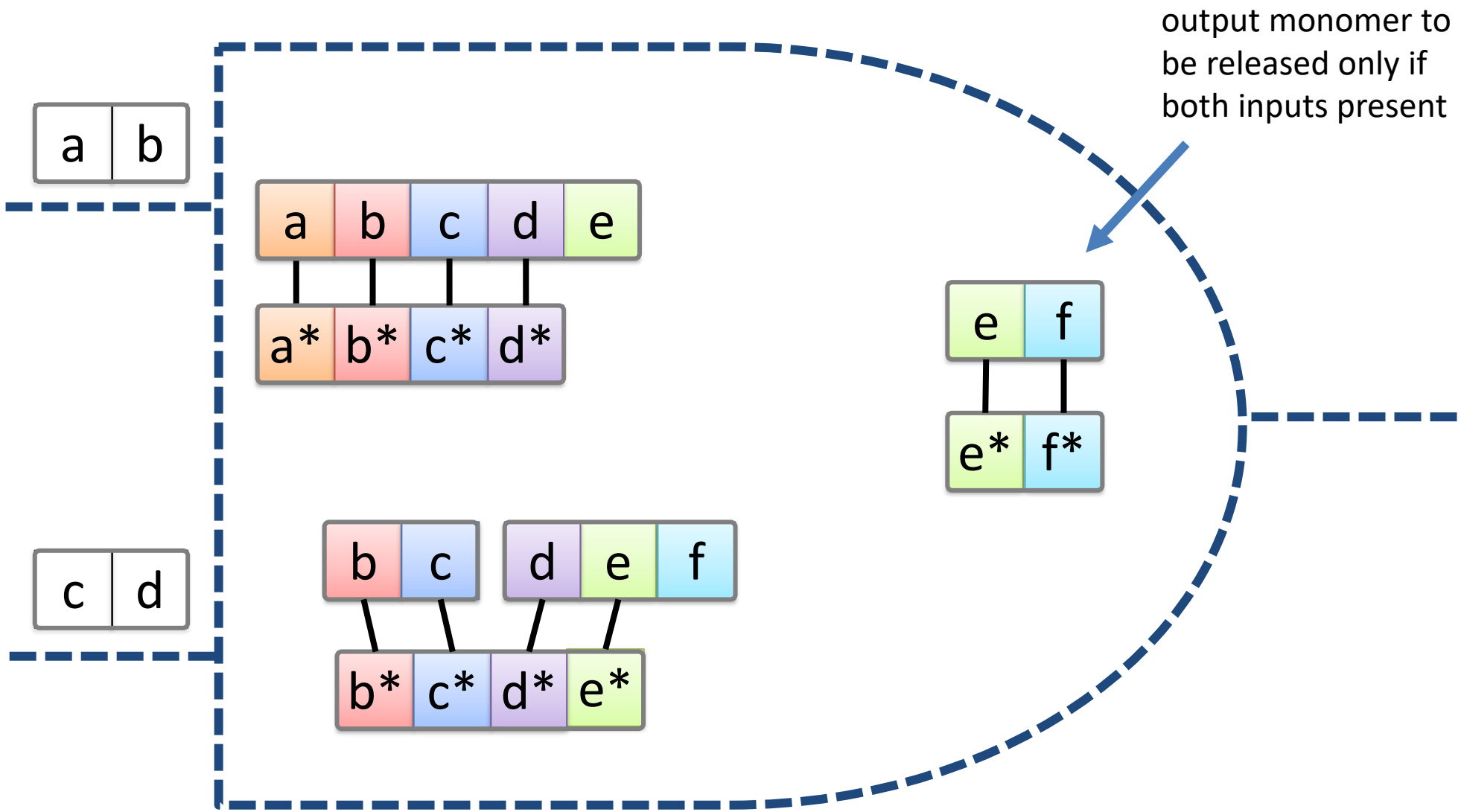
AND gate


output monomer to be released only if both inputs present
a
b
a
b
c
d
e
a*
b*
c*
d*
e
f
e*
f*
c
d
b
c
d
e
f
b*
c*
d*
e*

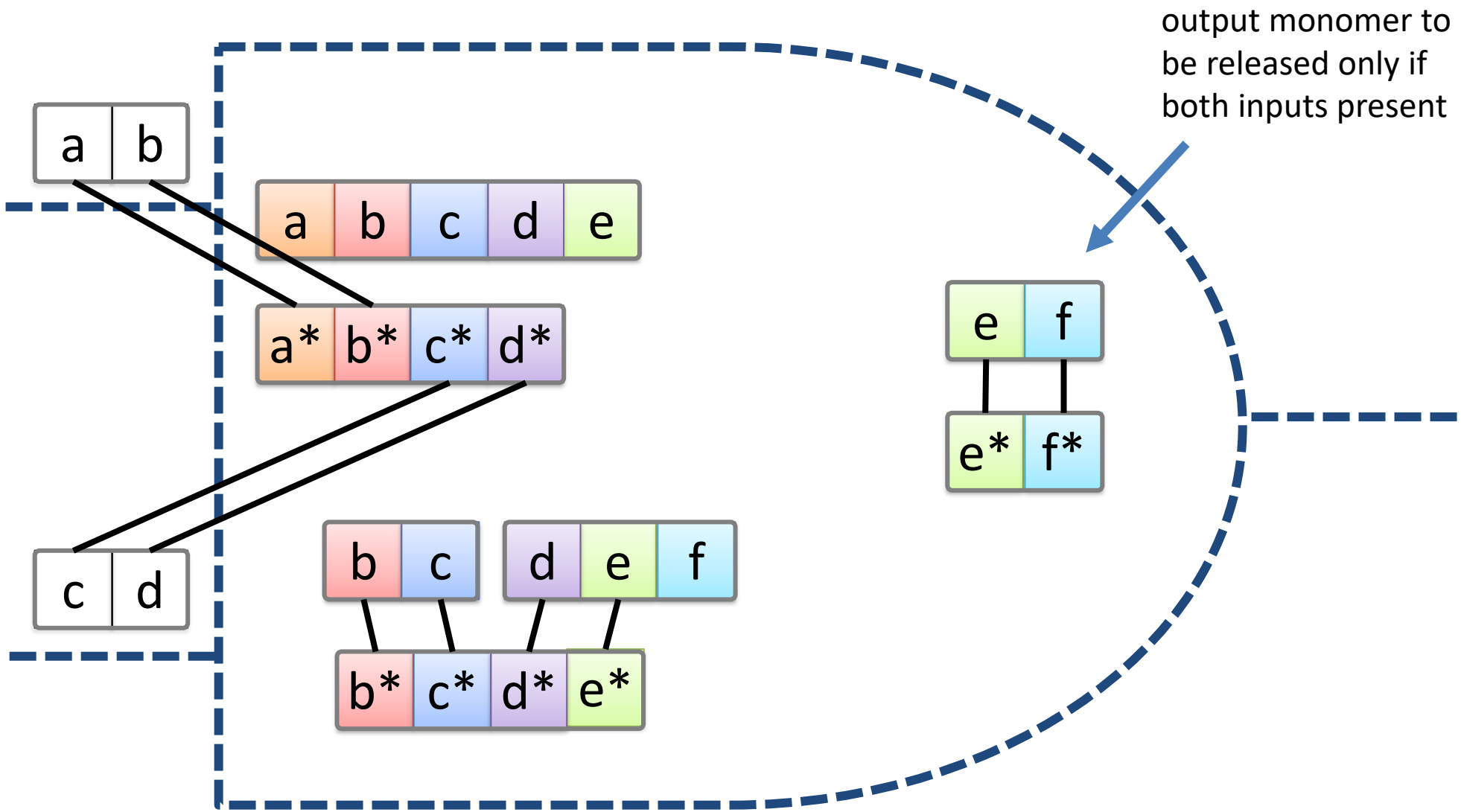
AND gate


output monomer to be released only if both inputs present
a
b
c
d
a
b
c
d
e
a*
b*
c*
d*
e
f
e*
f*
b
c
d
e
f
b*
c*
d*
e*

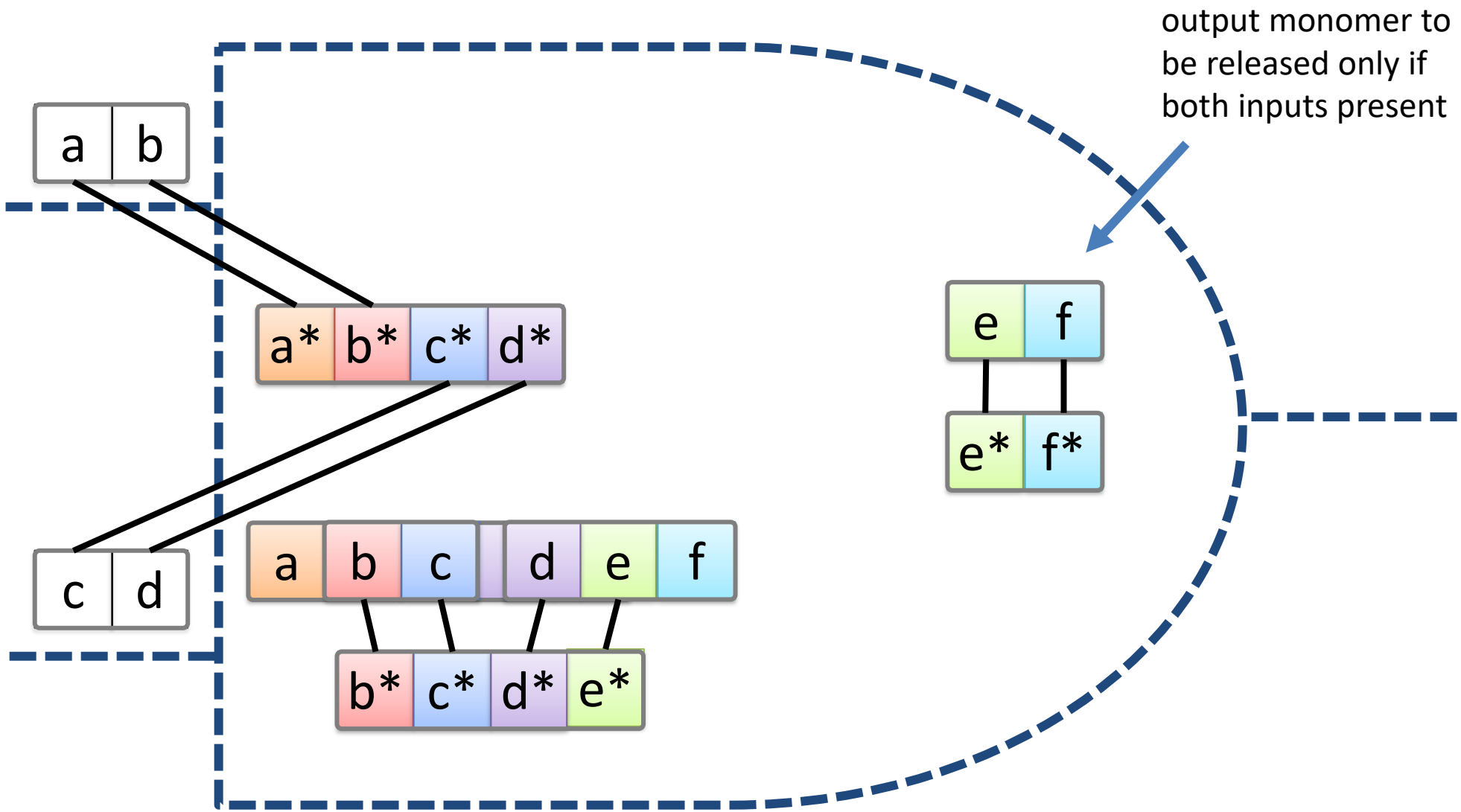
AND gate


output monomer to be released only if both inputs present
a
b
c
d
a
b
c
d
e
a*
b*
c*
d*
e
f
e*
f*
b
c
d
e
f
b*
c*
d*
e*

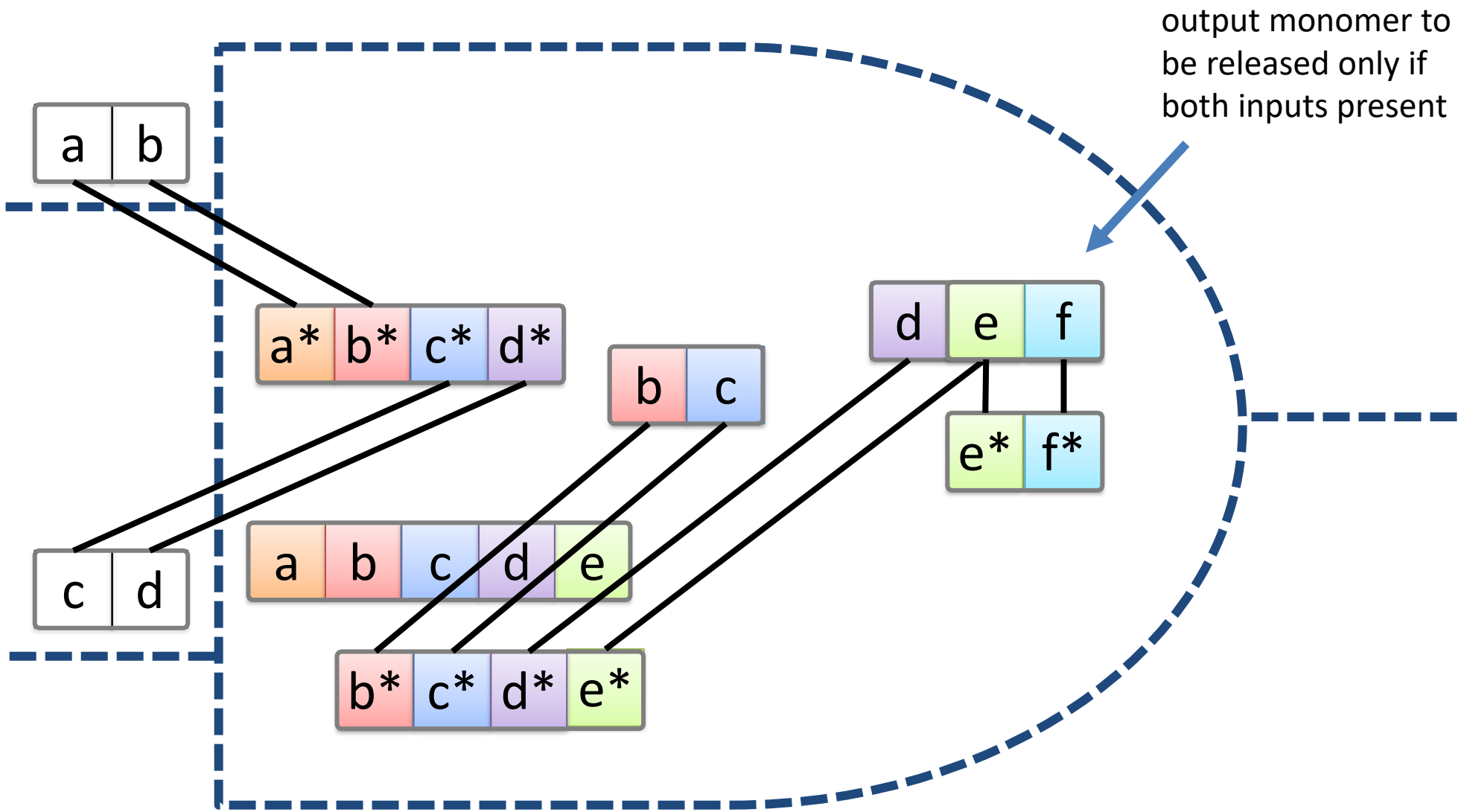
AND gate


output monomer to be released only if both inputs present
a
b
a b c d e
a* b* c* d*
c
d
b c
d e f
b* c* d* e*
e
f
e*
f*

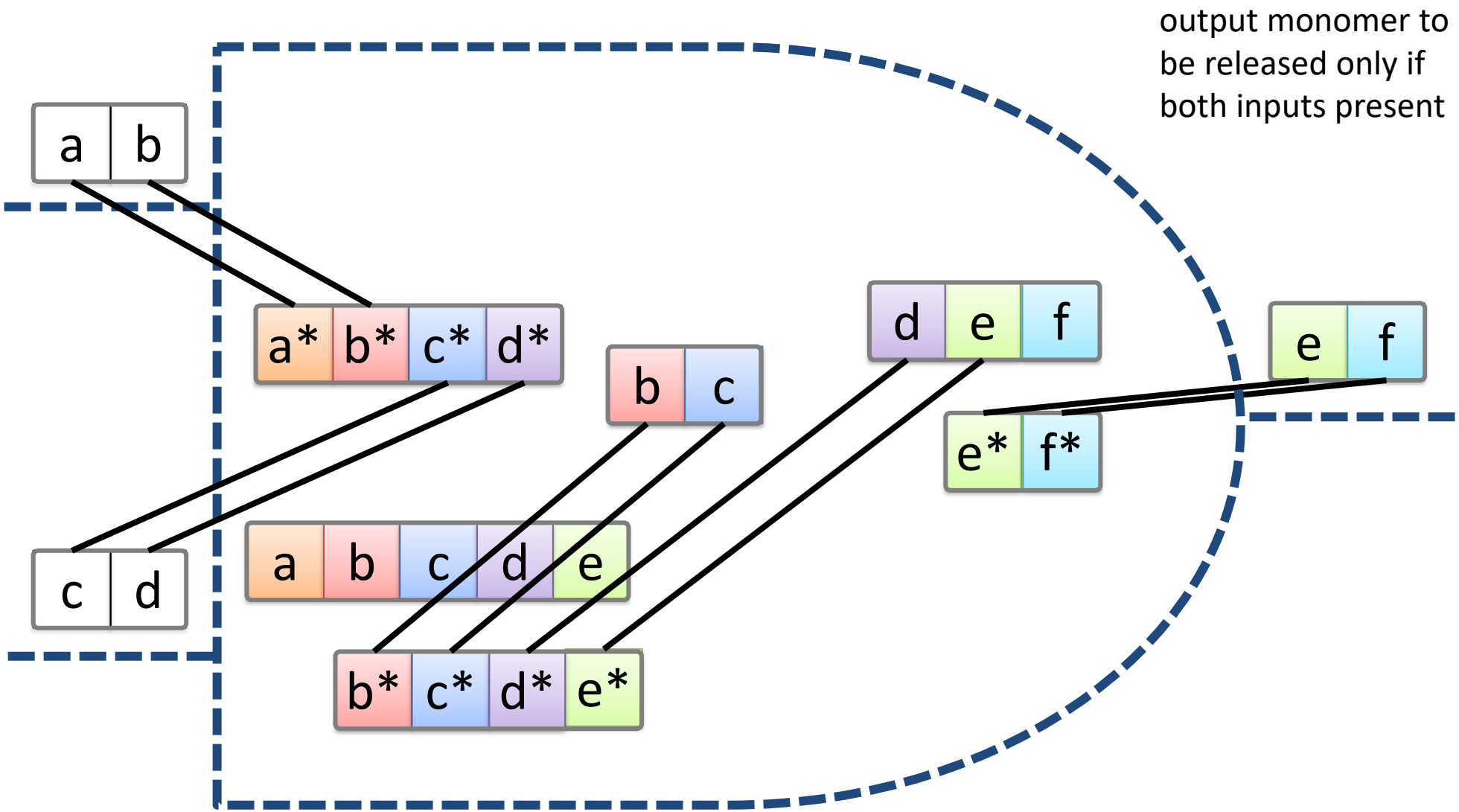
AND gate


output monomer to be released only if both inputs present
a
b
a*
b*
c*
d*
c
d
a
b
c
d
e
f
b*
c*
d*
e*
e
f
e*
f*

AND gate


output monomer to be released only if both inputs present
a
b
a*
b*
c*
d*
b
c
d
e
f
e*
f*
c
d
a
b
c
d
e
b*
c*
d*
e*

AND gate


output monomer to be released only if both inputs present
a
b
a*
b*
c*
d*
b
c
d
e
f
e
f
e*
f*
c
d
a
b
c
d
e
b*
c*
d*
e*

Issues with Boolean logic

Issues with Boolean logic

- How to compose?
  - We don’t know how to prove the previous gate is composable, and used a more complex design in the paper

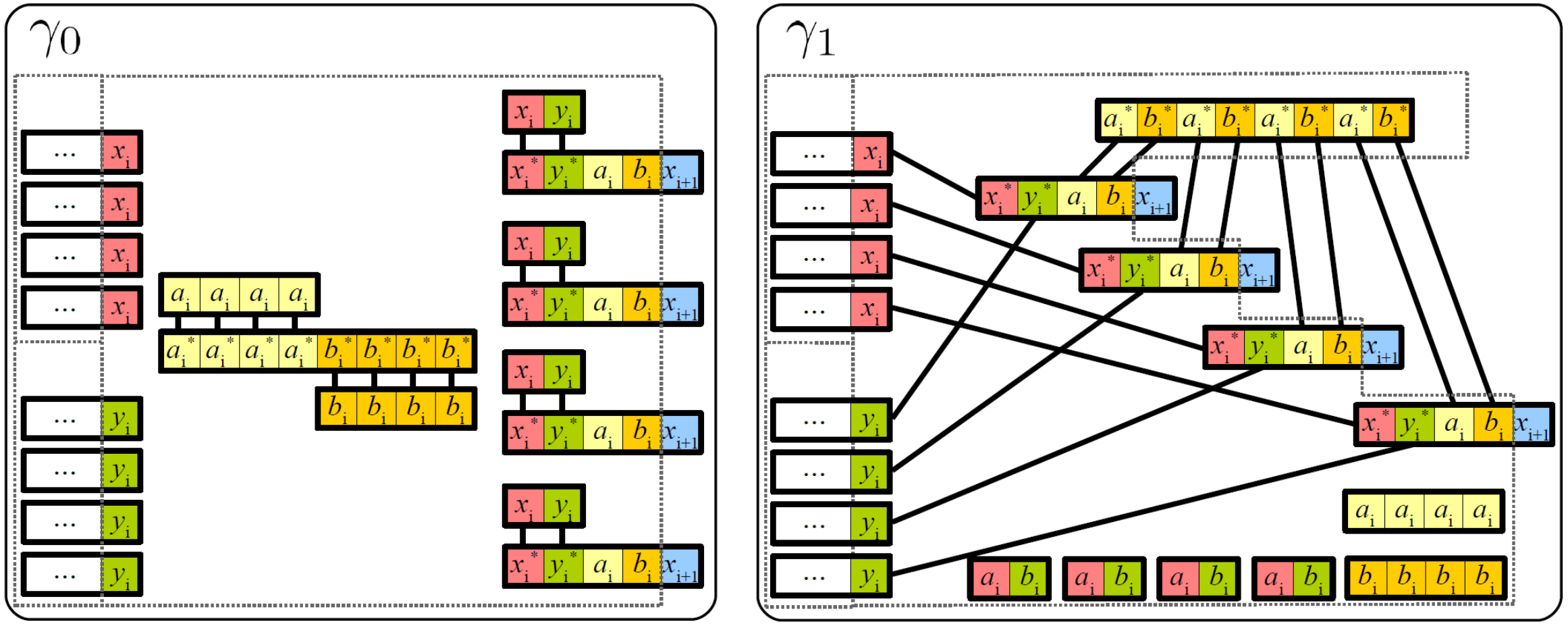
Issues with Boolean logic

- How to compose?
  - We don't know how to prove the previous gate is composable, and used a more complex design in the paper
- Want "entropy gap":
  - Need not merely that unwanted configurations are unstable (i.e., if saturated, they have lower entropy), but more strongly that they have much lower entropy.
  - We can use $O(n)$ domain/monomer types to achieve an entropy gap of n.

Issues with Boolean logic

- How to compose?
  - We don't know how to prove the previous gate is composable, and used a more complex design in the paper
- Want "entropy gap":
  - Need not merely that unwanted configurations are unstable (i.e., if saturated, they have lower entropy), but more strongly that they have much lower entropy.
  - We can use $O(n)$ domain/monomer types to achieve an entropy gap of n.
- Output convention?
  - Obvious one: "there's a unique stable configuration with the correct output"
  - It's problematic, so we have a one-sided convention:
    - if correct output is 0, unique stable configuration with correct answer
    - if correct output is 1, then both the "output=1" and "output=0" configurations are stable

Composable AND gate with entropy gap 3



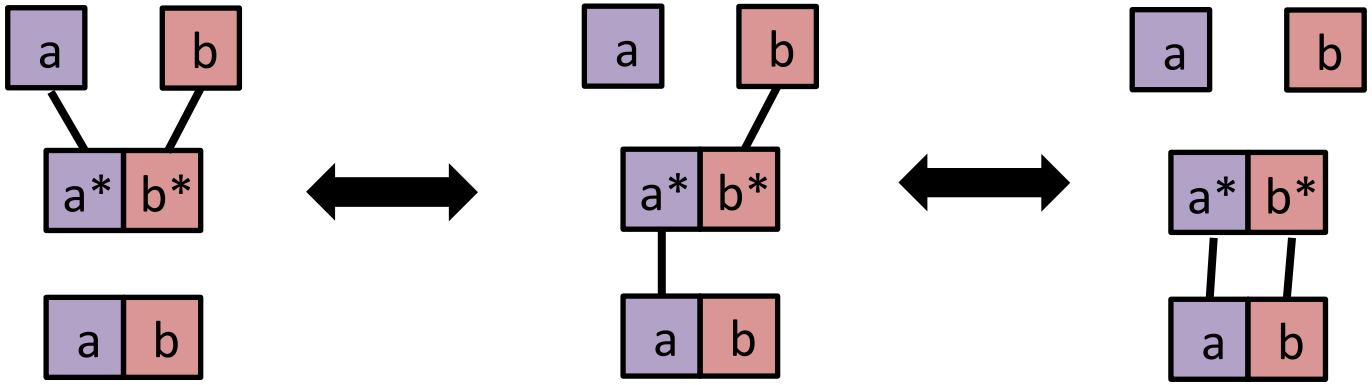
Rather than release a single output monomer, it suffices to gather all output domains on one complex.

Kinetic pathways and energy barriers

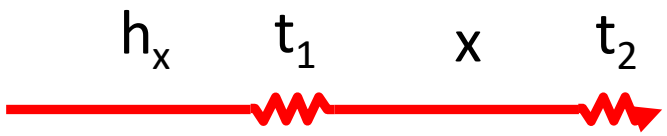
Pathways

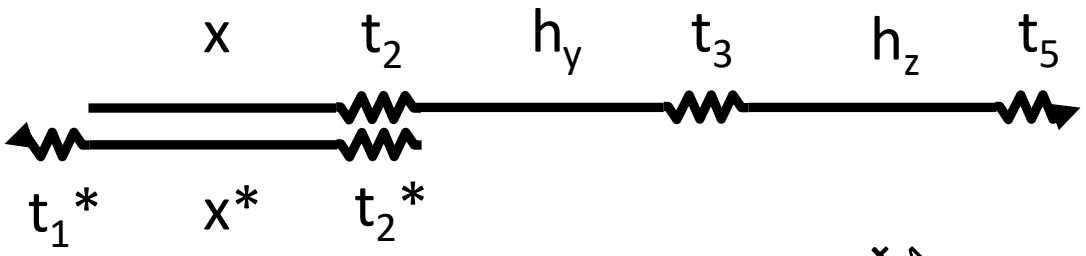
Thermodynamics: Which configurations are energetically favorable

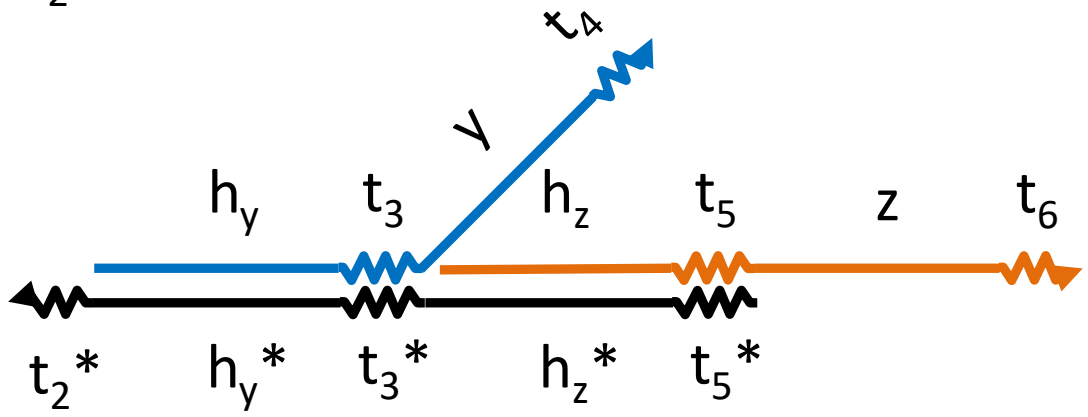
Kinetics: How a system moves between configurations over time



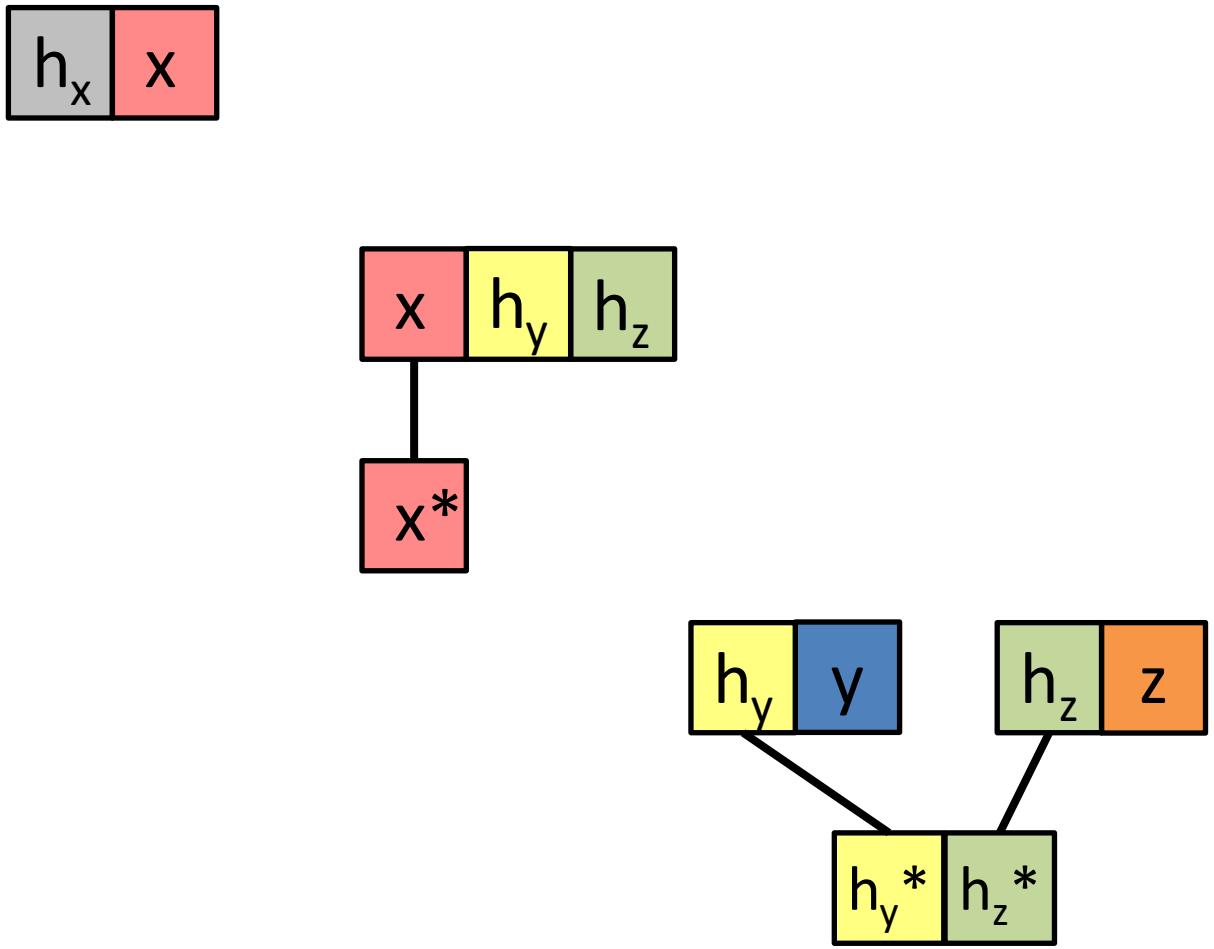
$$X \longrightarrow Y + Z$$

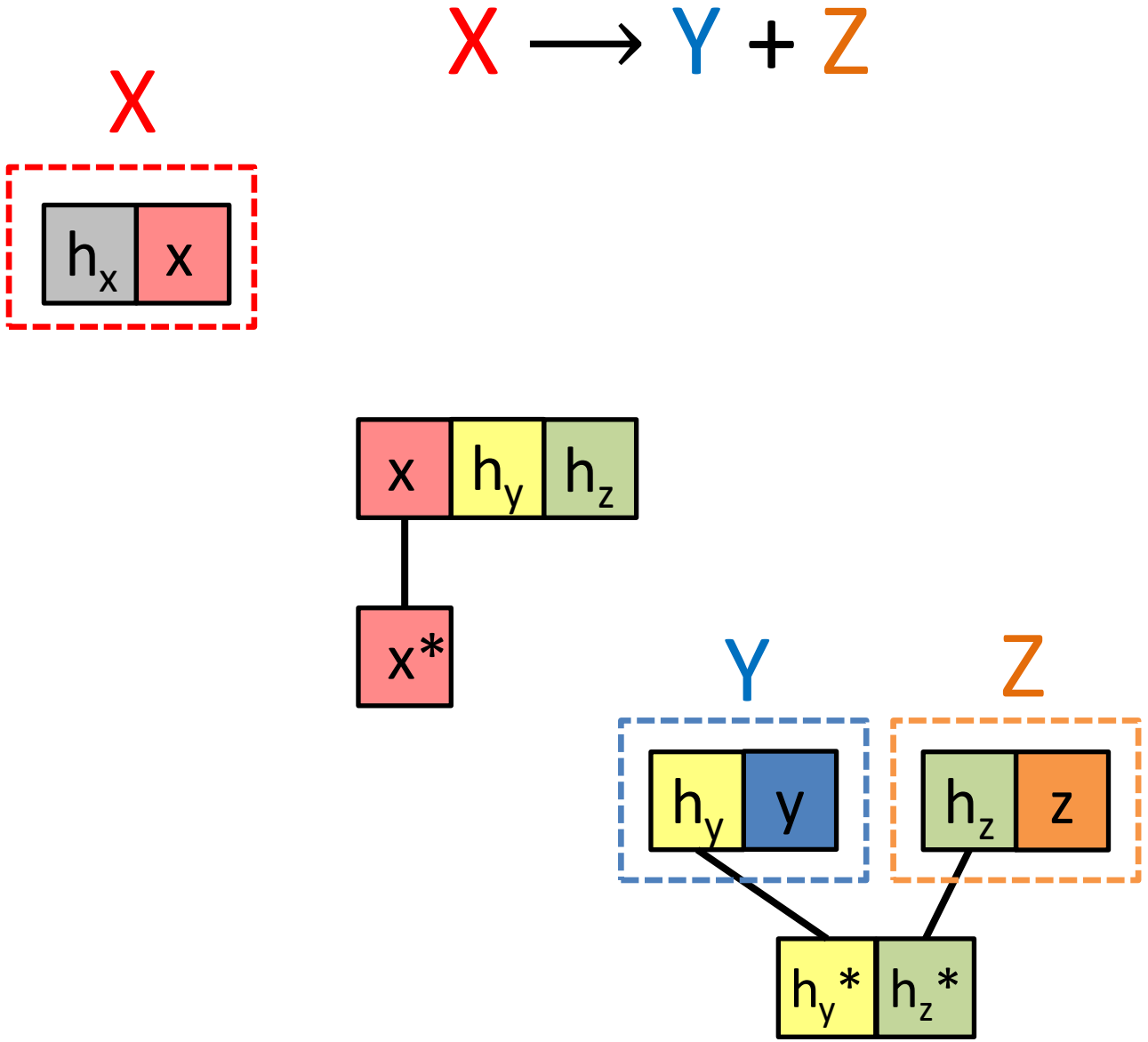




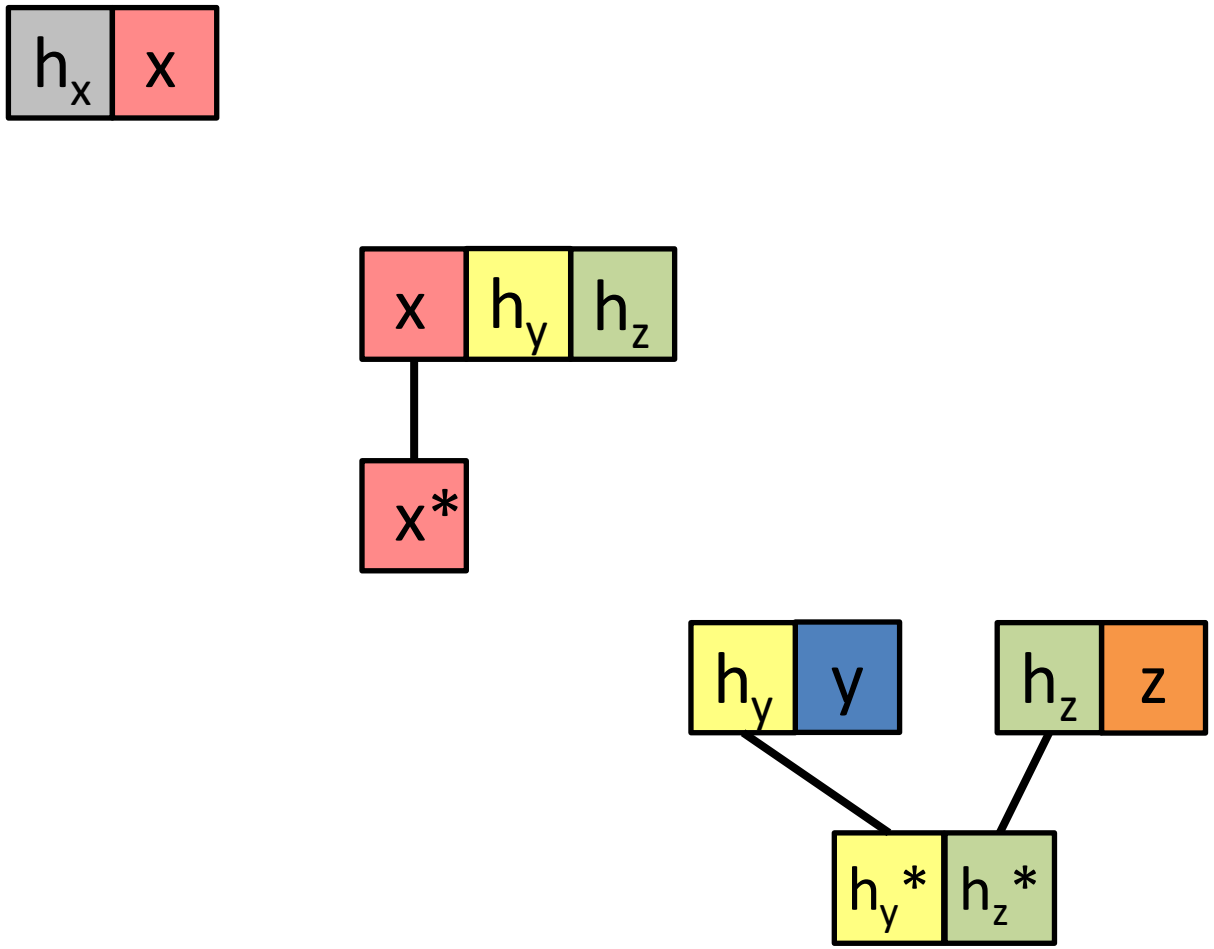


$$X \longrightarrow Y + Z$$

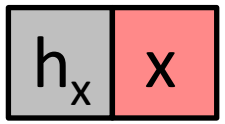


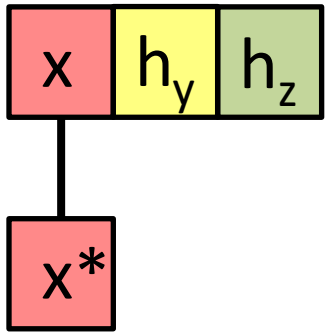

X ⟶ Y + Z
X
h_x
x
x
h_y
h_z
x*
Y
Z
h_y
y
h_z
z
h_y*
h_z*

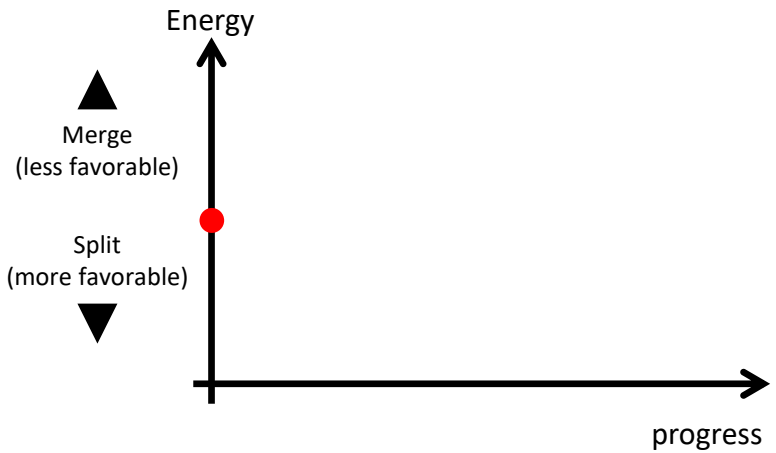
$$X \longrightarrow Y + Z$$

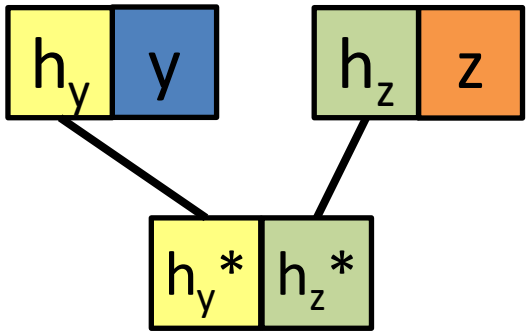


$$X \longrightarrow Y + Z$$

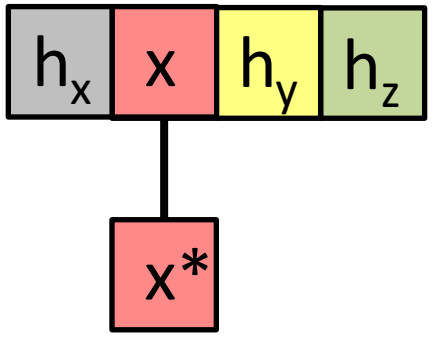


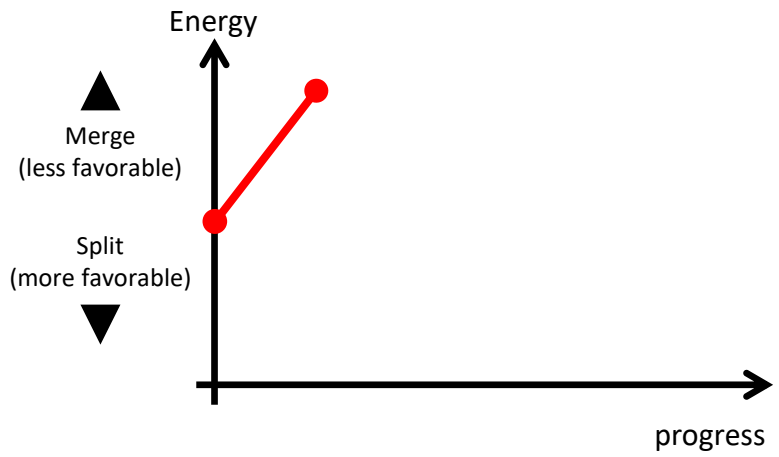


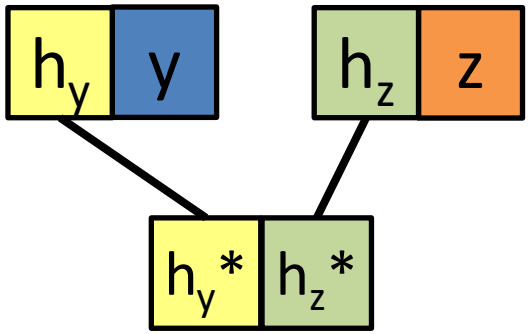




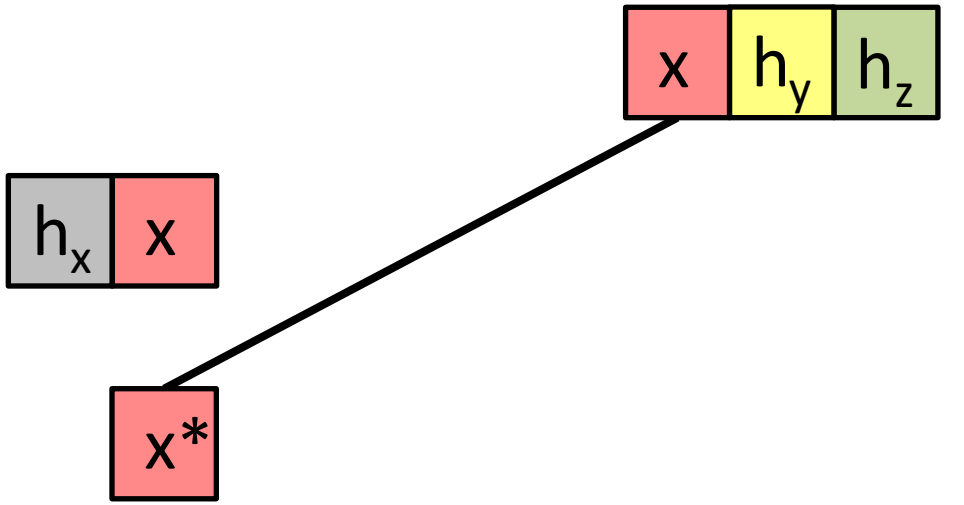
$$X \longrightarrow Y + Z$$

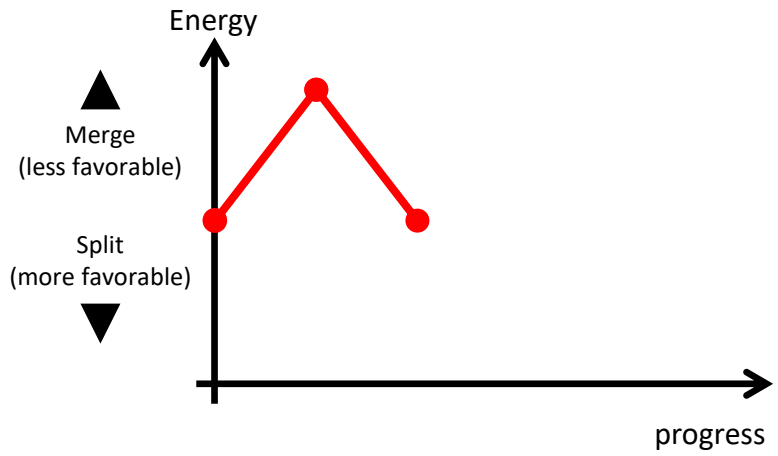


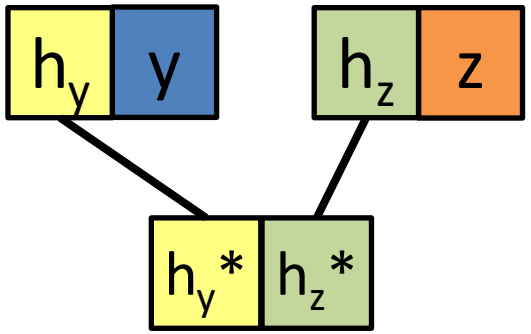




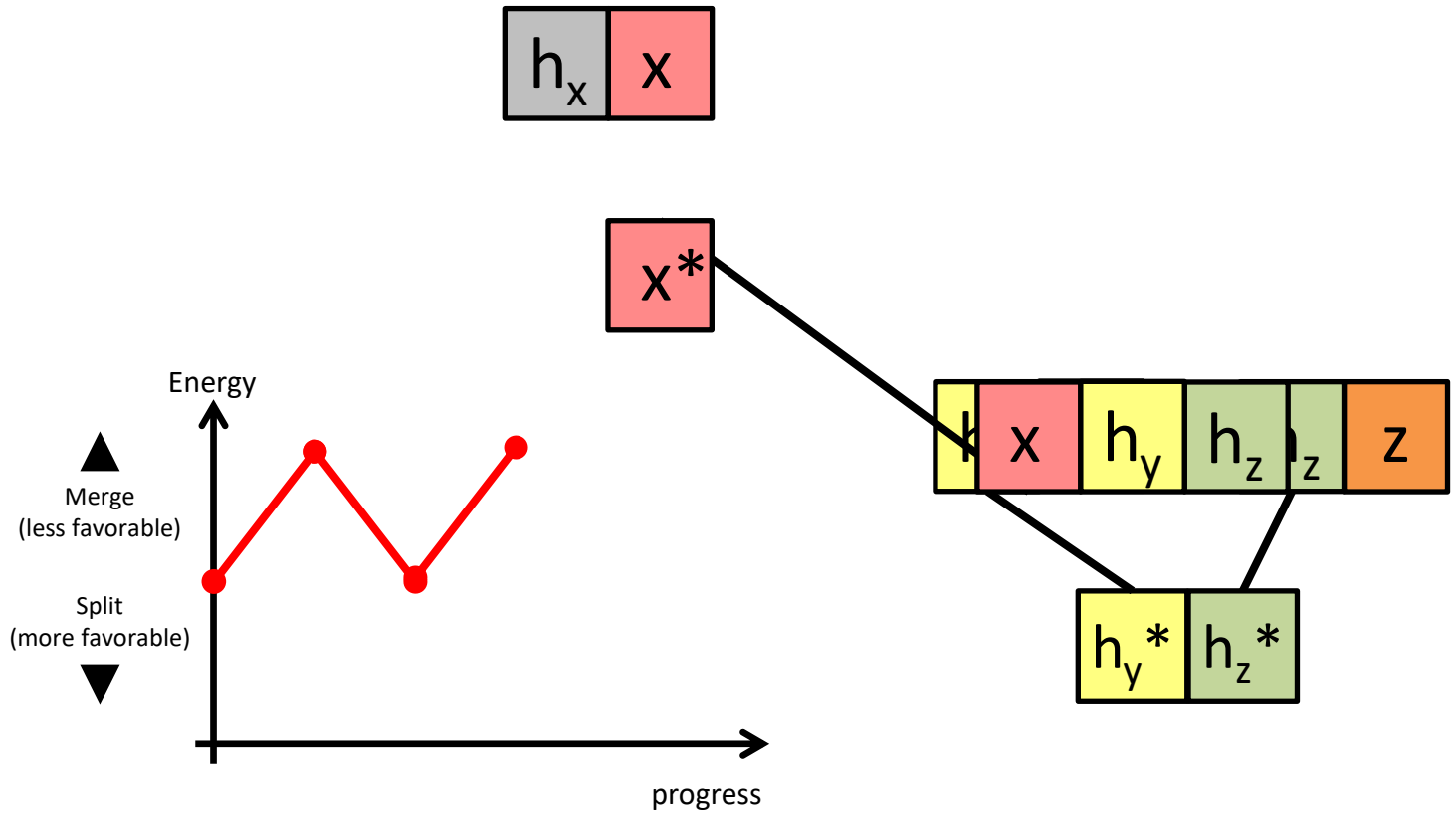
$$X \longrightarrow Y + Z$$



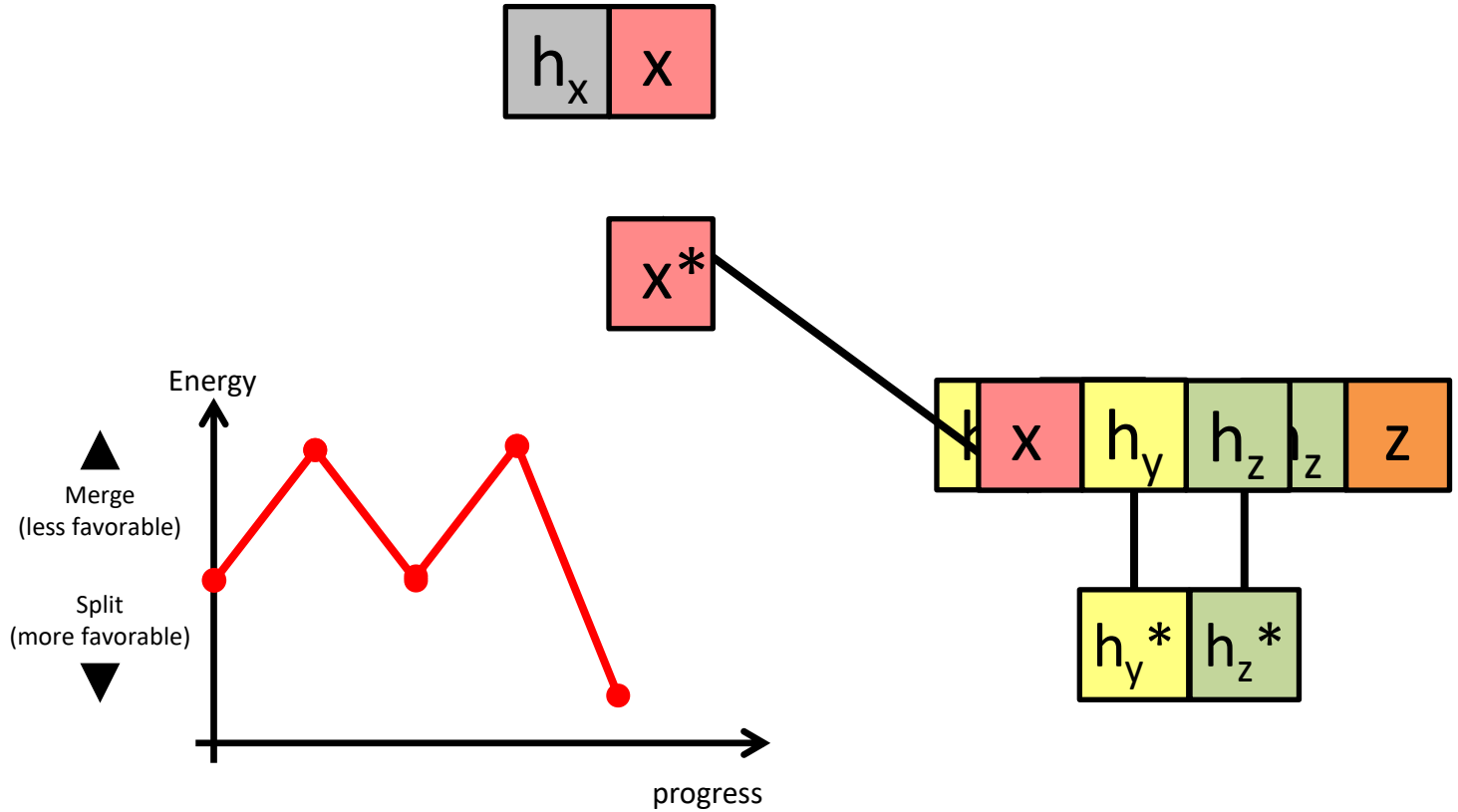




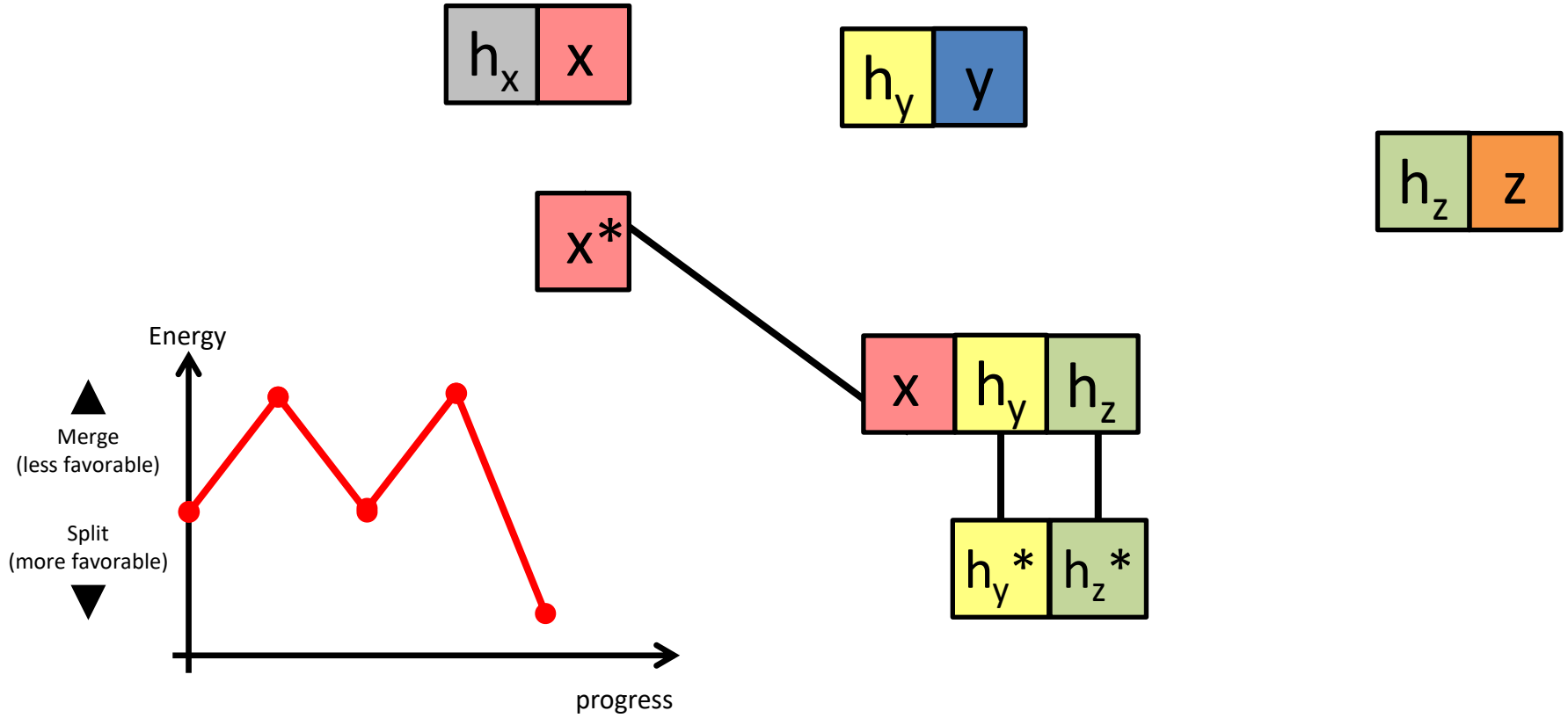
$$X \longrightarrow Y + Z$$



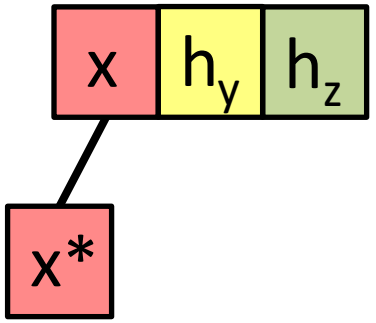
$$X \longrightarrow Y + Z$$

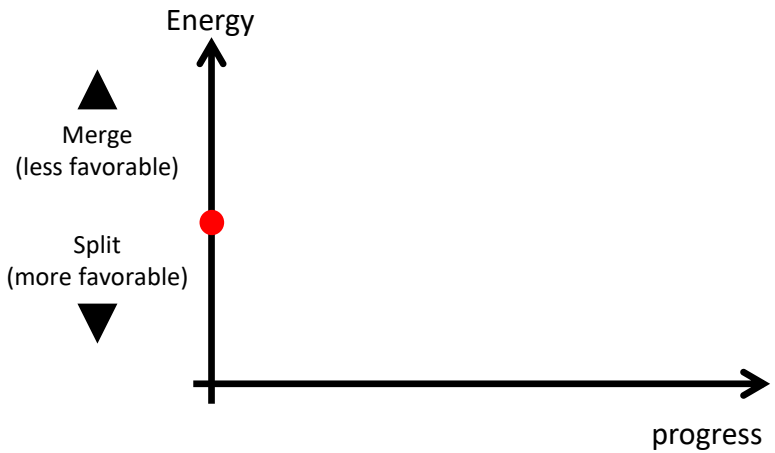


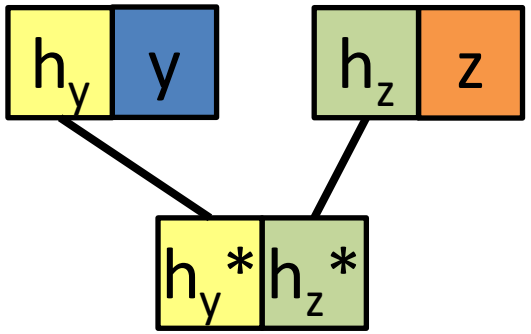
$$X \longrightarrow Y + Z$$



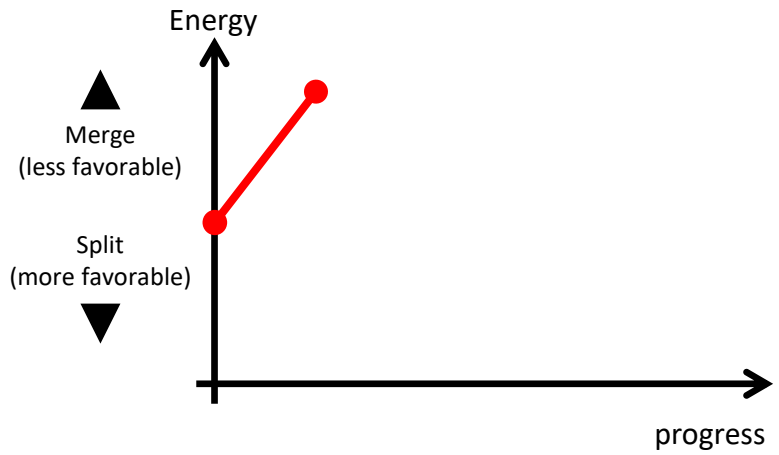
$$\emptyset \longrightarrow Y + Z$$

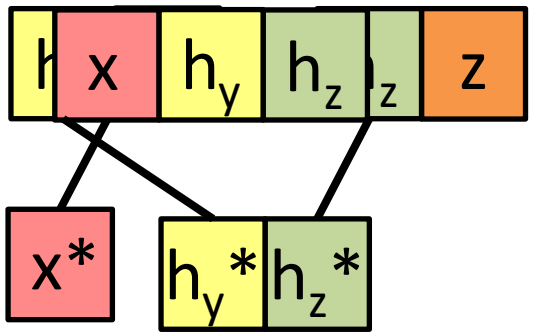




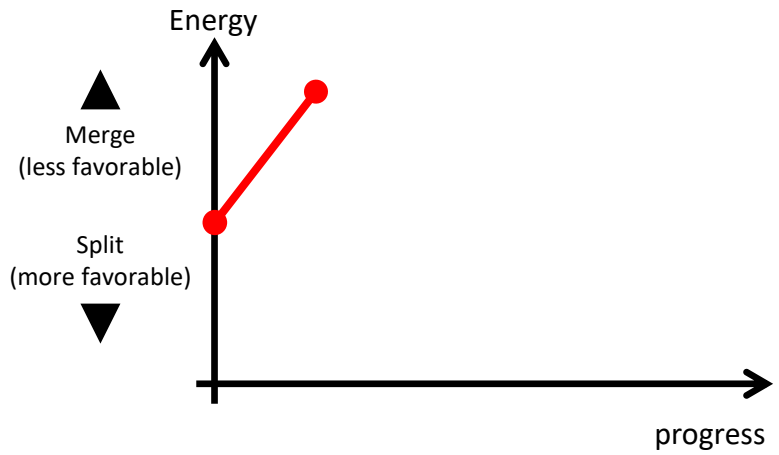


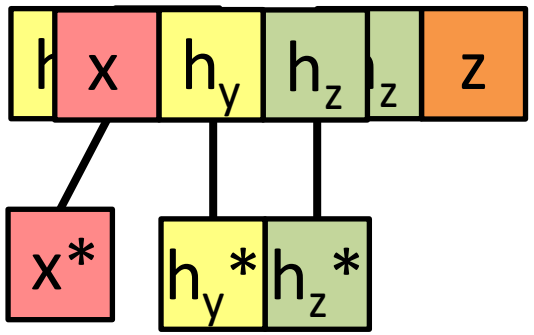
$$\emptyset \longrightarrow Y + Z$$





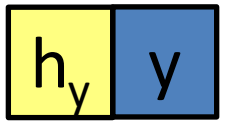
$$\emptyset \longrightarrow Y + Z$$

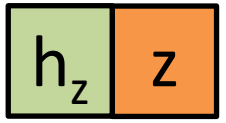


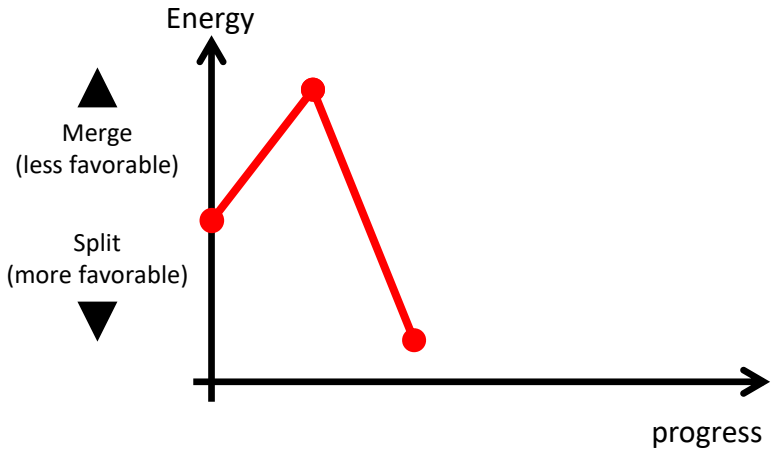


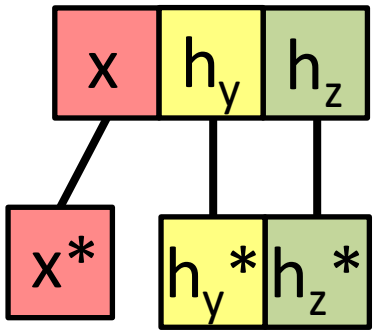
What causes leak "kinetically"?

$$\emptyset \longrightarrow Y + Z$$









Kinetic Binding Networks

Kinetic Binding Networks

- Favorability is a combination of bond count and complex count

Kinetic Binding Networks

- Favorability is a combination of bond count and complex count

<u>Weighted average:</u>

Energy := $-w_H(\#\text{ bonds}) - (\#\text{ complexes})$

Kinetic Binding Networks

- Favorability is a combination of bond count and complex count

Weighted average:

$$\text{Energy} := -w_H(\#\text{ bonds}) - (\#\text{ complexes})$$

- Define pathways to consist of merges and splits

Kinetic Binding Networks

- Favorability is a combination of bond count and complex count

Weighted average:

$$\text{Energy} := -w_H(\#\text{ bonds}) - (\#\text{ complexes})$$

- Define pathways to consist of merges and splits
- But for $w_H \geq 2$, only saturated pathways need be considered

[Keenan Breik, Cameron Chalk, David Doty, David Haley, David Soloveichik. *Programming Substrate-Independent Kinetic Barriers with Thermodynamic Binding Networks*. Computational Methods in Systems Biology 2018]

Kinetic Binding Networks

- Favorability is a combination of bond count and complex count

Weighted average:

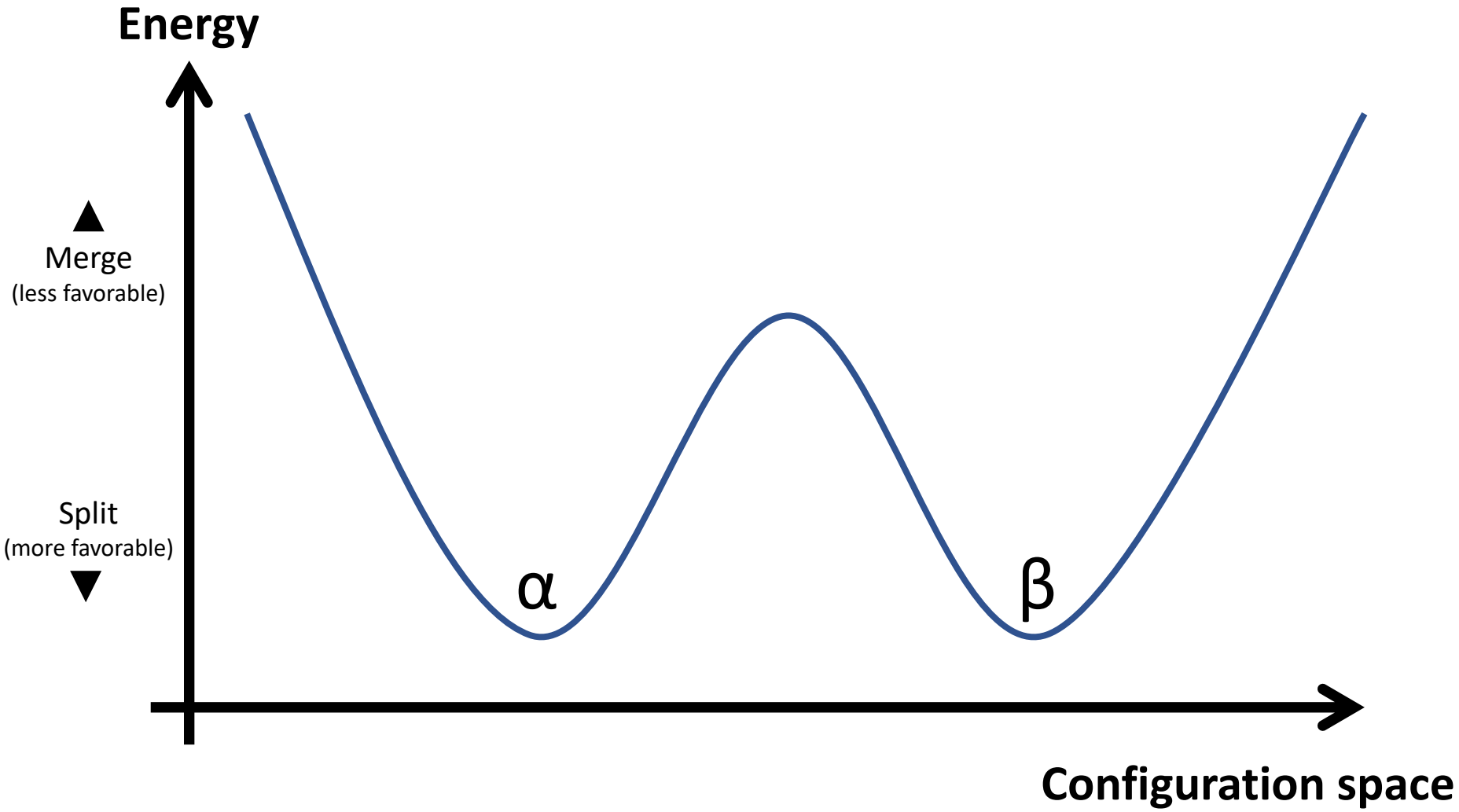
$$\text{Energy} := -w_H(\#\text{ bonds}) - (\#\text{ complexes})$$

- Define pathways to consist of merges and splits
- But for $w_H \geq 2$, only saturated pathways need be considered

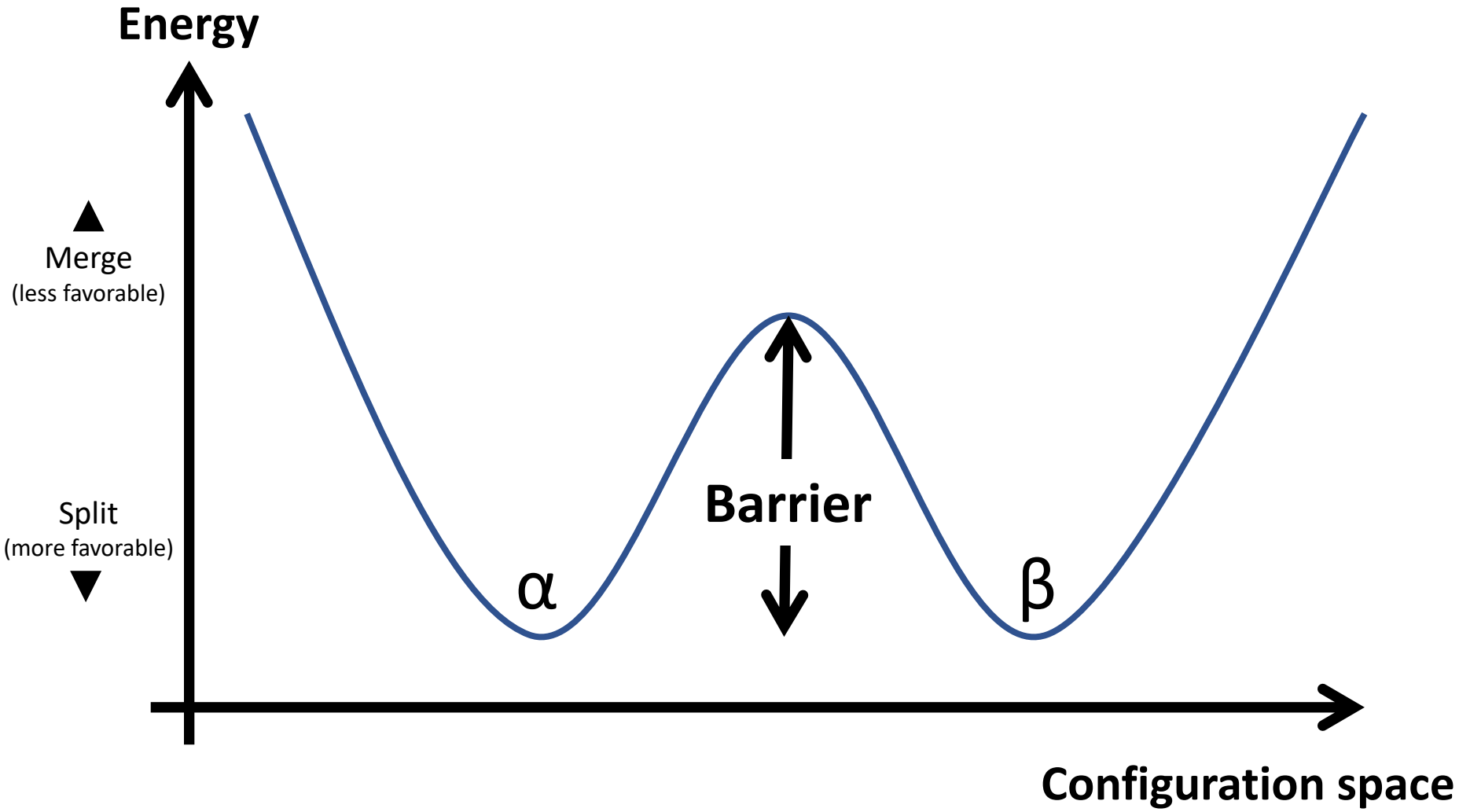
Since all saturated configurations have an equal number of bonds, we can focus solely on the number of complexes

[Keenan Breik, Cameron Chalk, David Doty, David Haley, David Soloveichik. *Programming Substrate-Independent Kinetic Barriers with Thermodynamic Binding Networks*. Computational Methods in Systems Biology 2018]

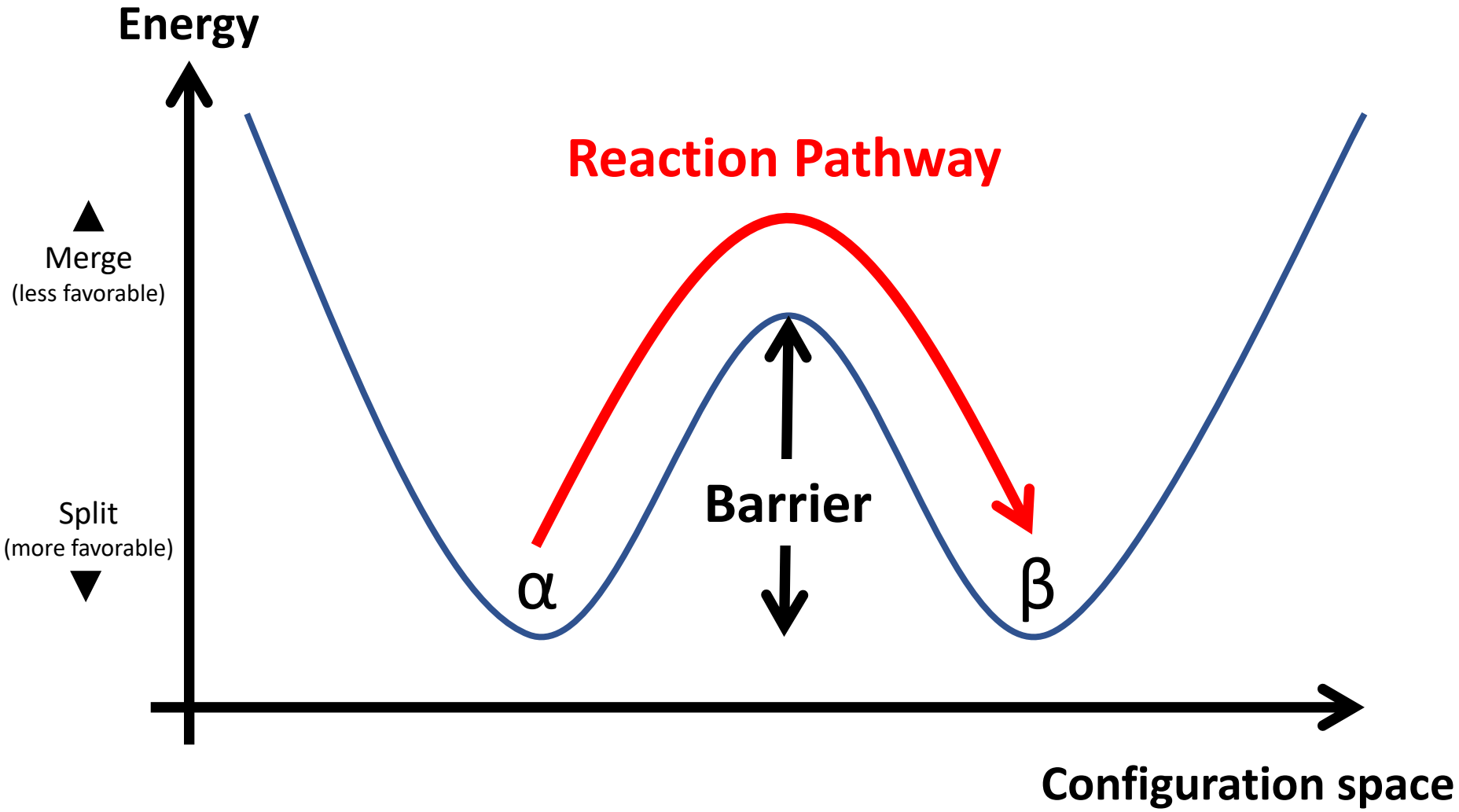
Large Energy Barriers



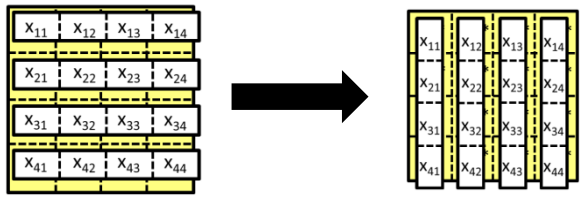
Large Energy Barriers

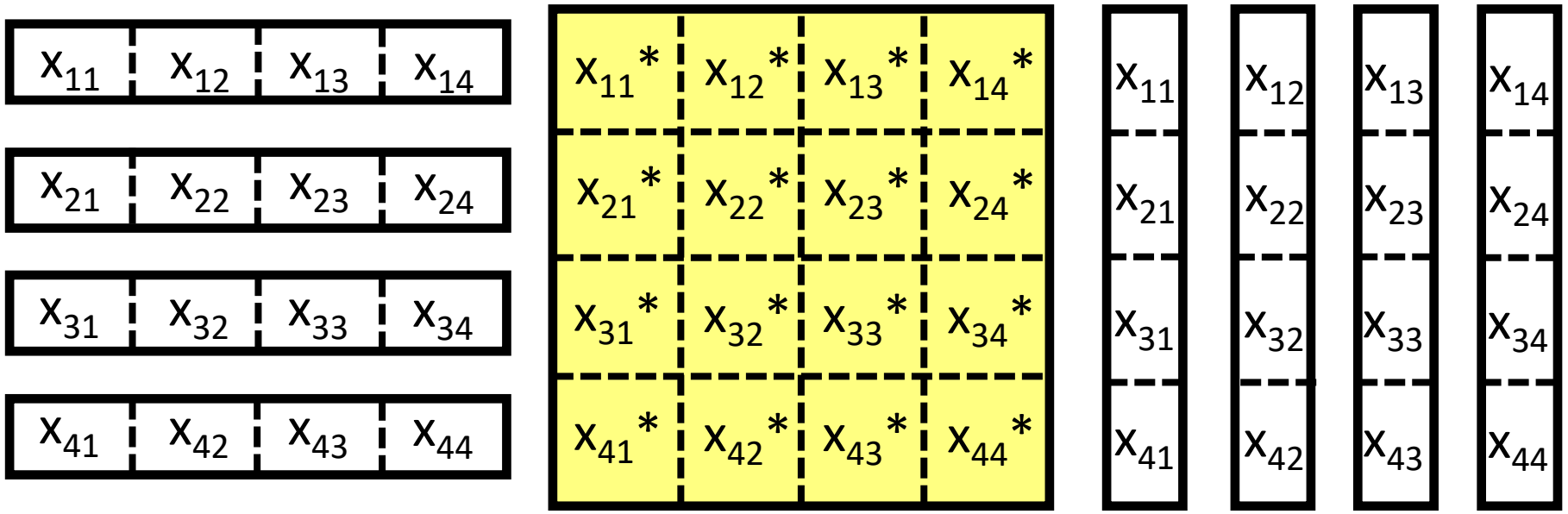


Large Energy Barriers

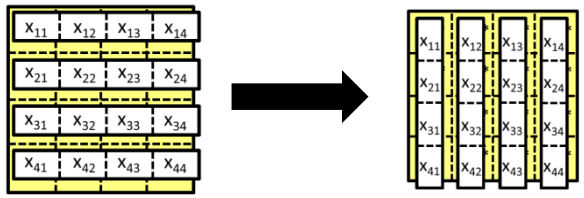


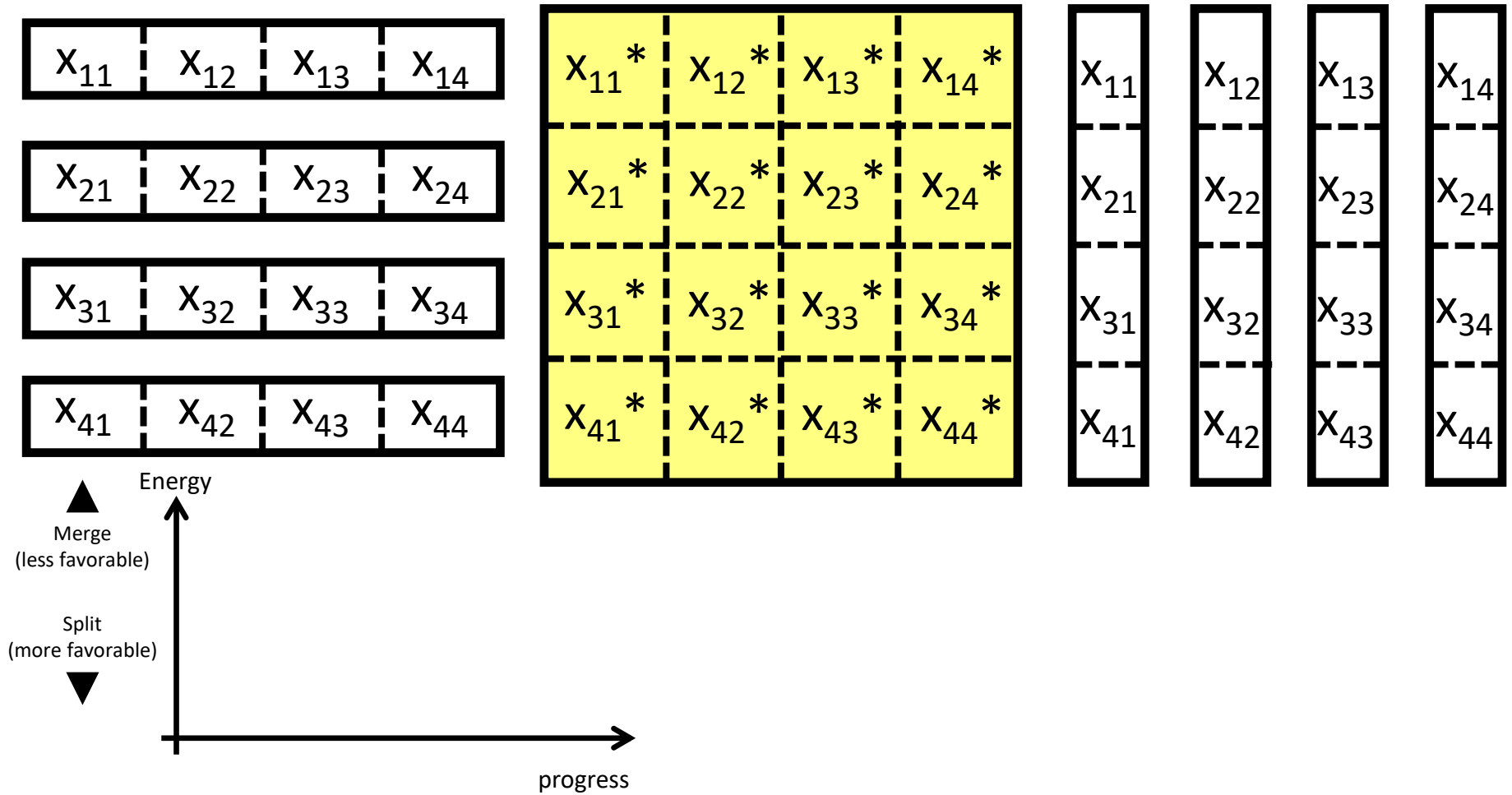
A Network with a Programmable Energy Barrier



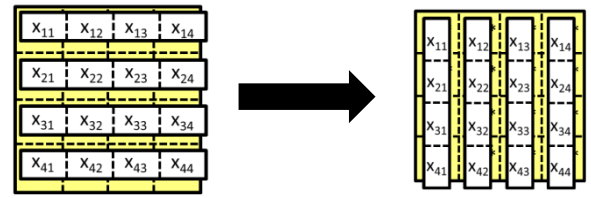


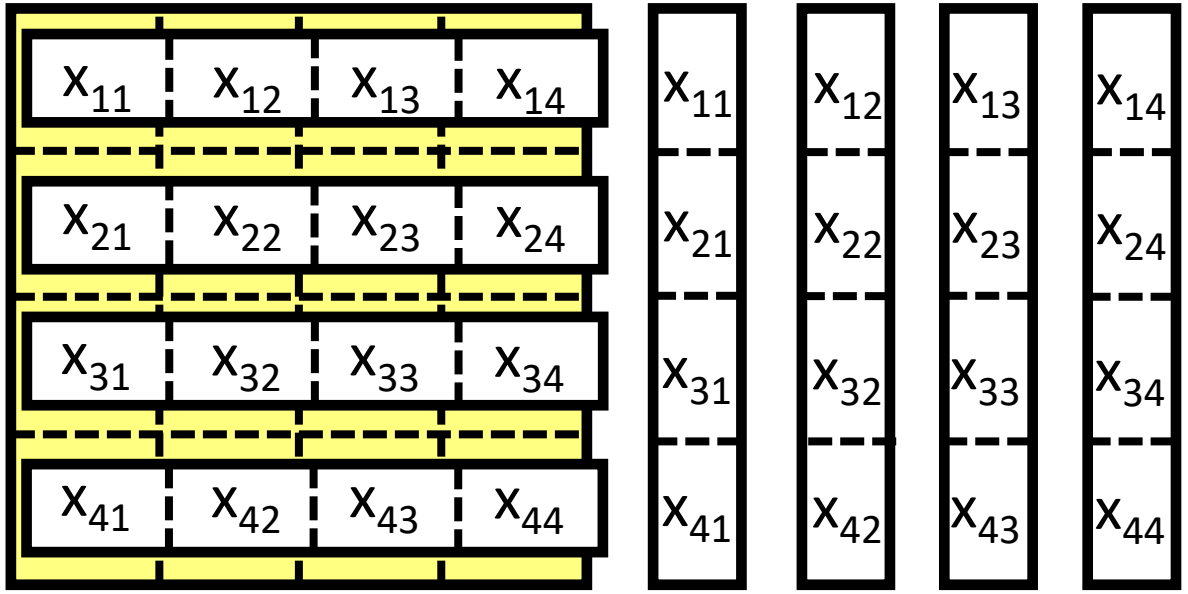
A Network with a Programmable Energy Barrier

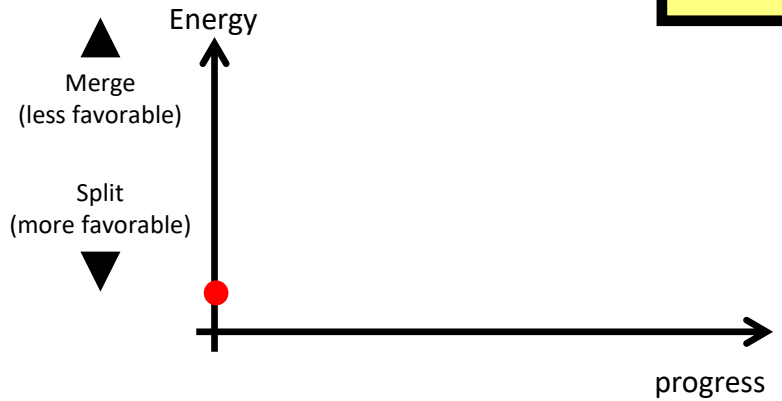




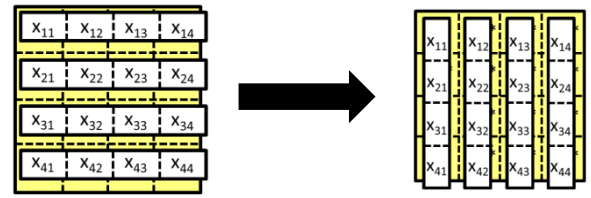
A Network with a Programmable Energy Barrier

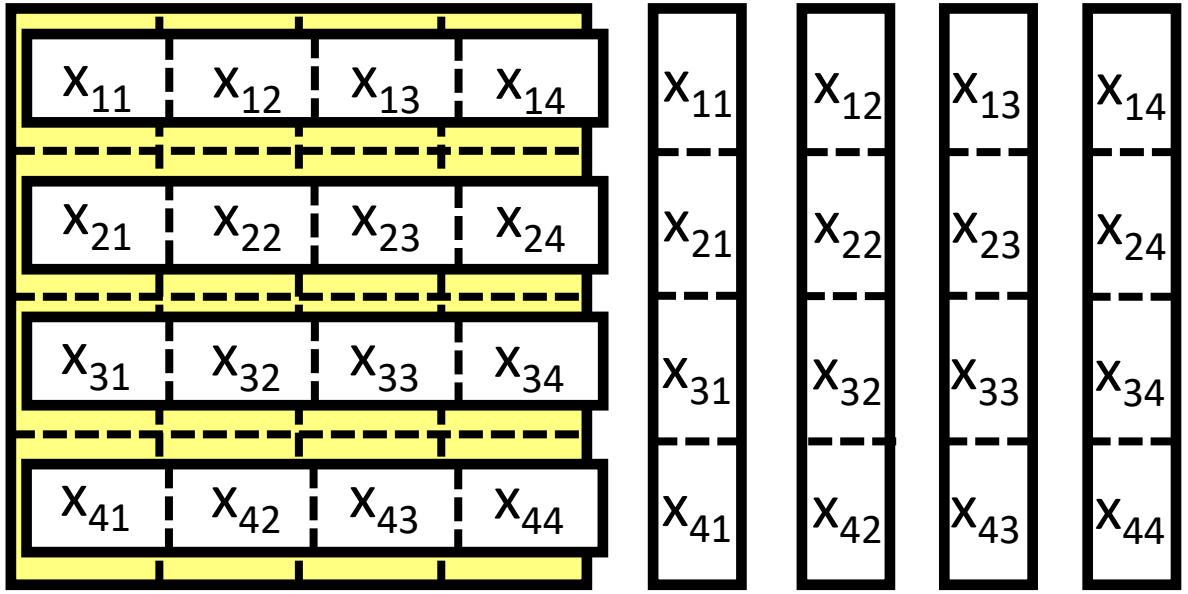


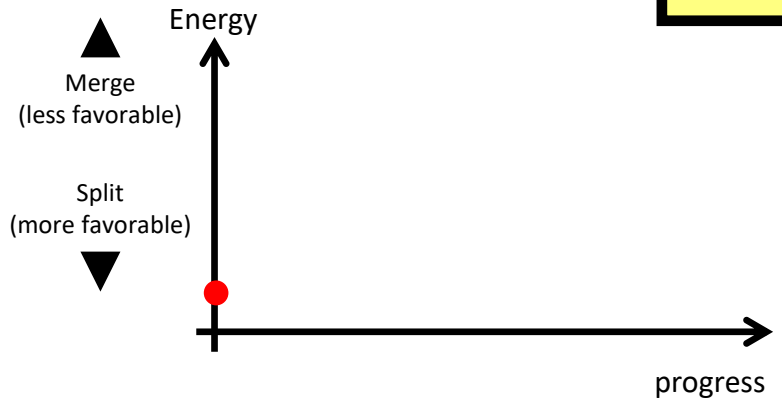




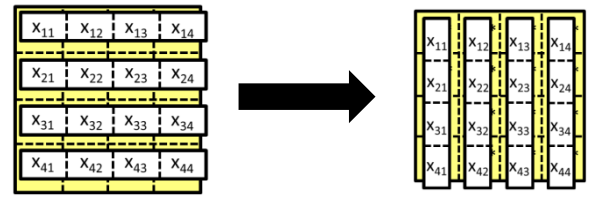
A Network with a Programmable Energy Barrier

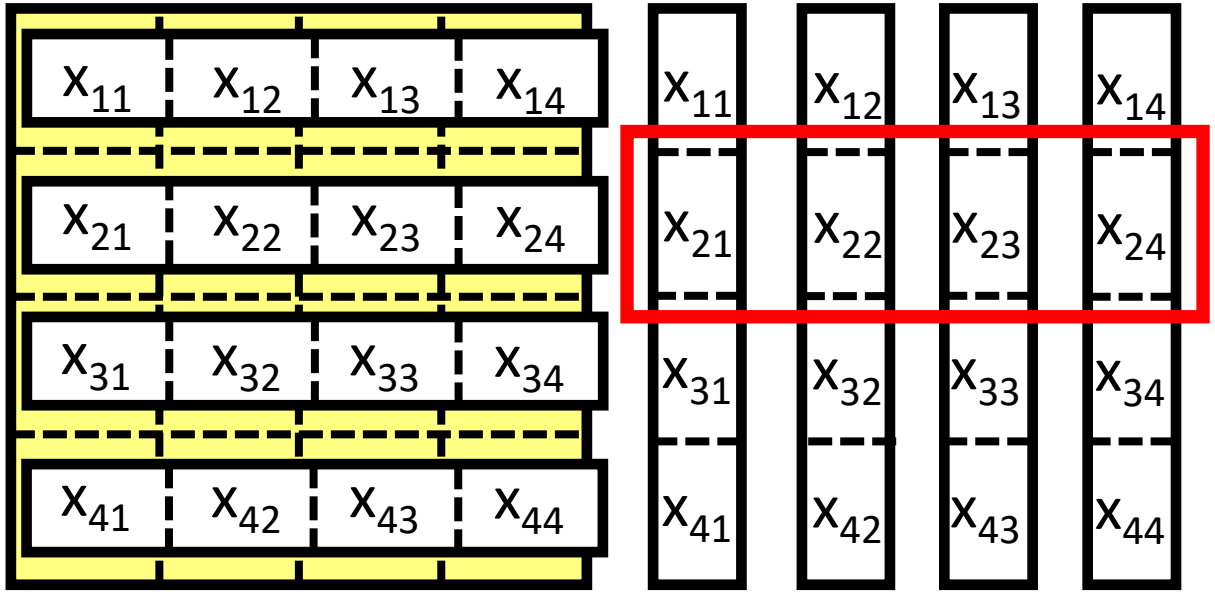


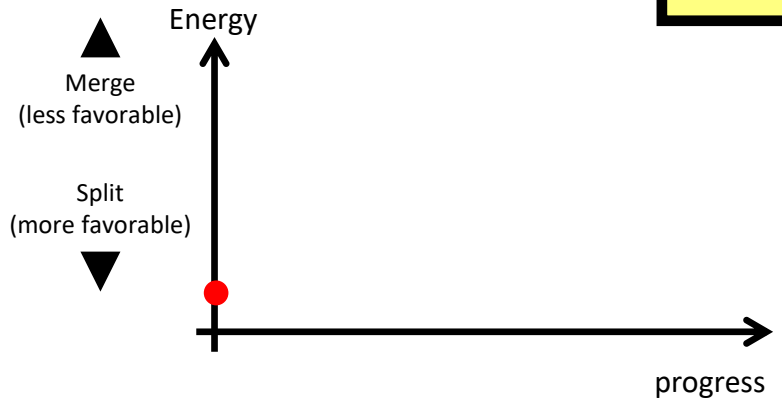




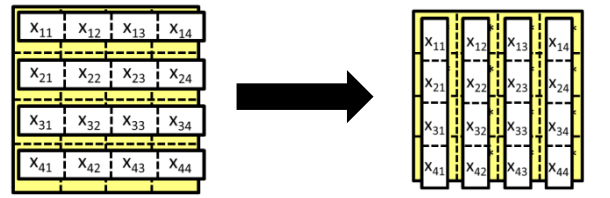
A Network with a Programmable Energy Barrier

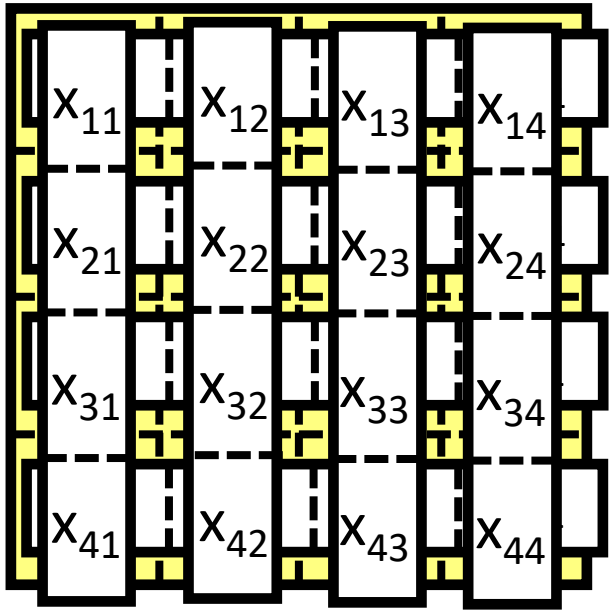


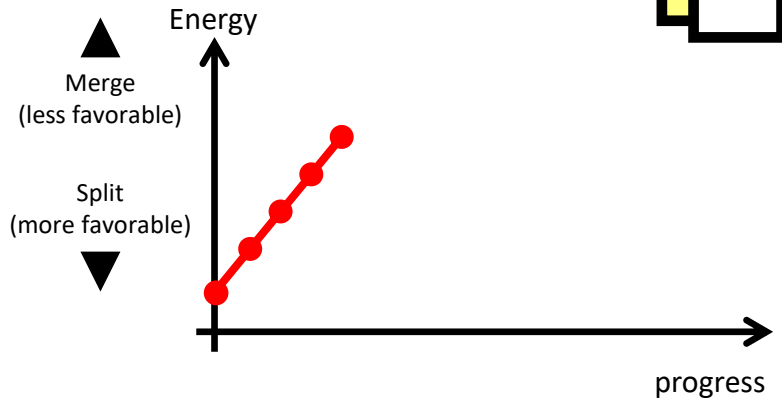




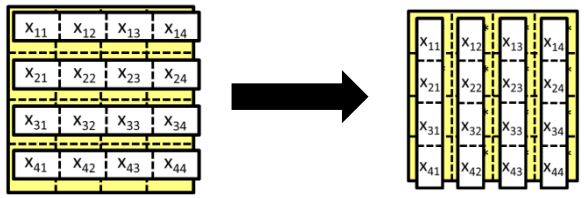
A Network with a Programmable Energy Barrier

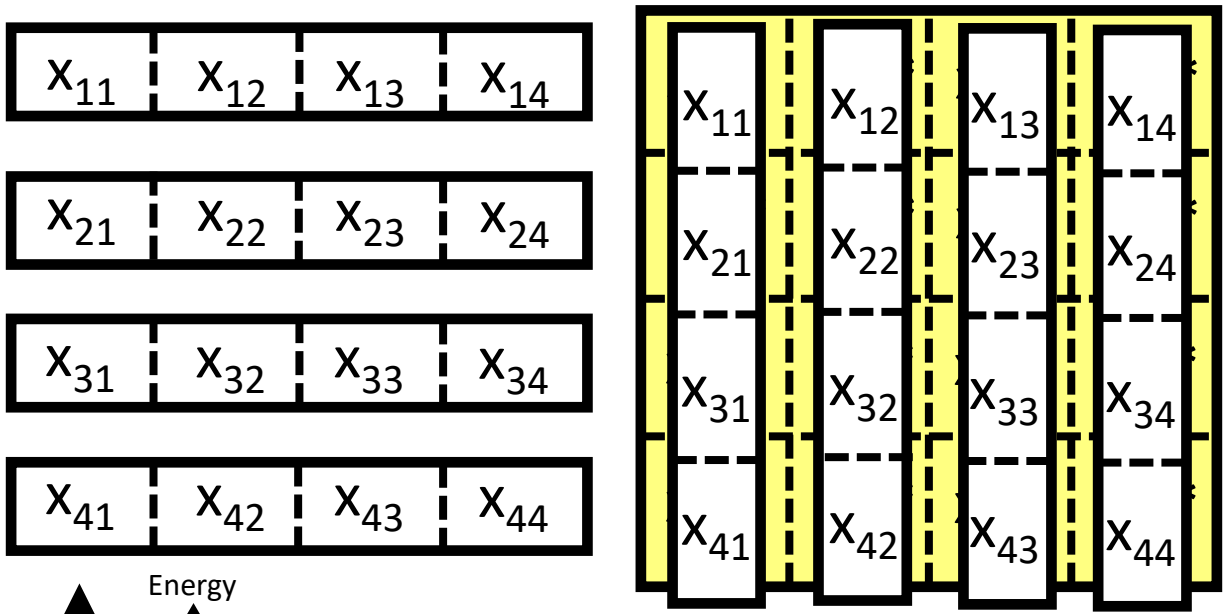




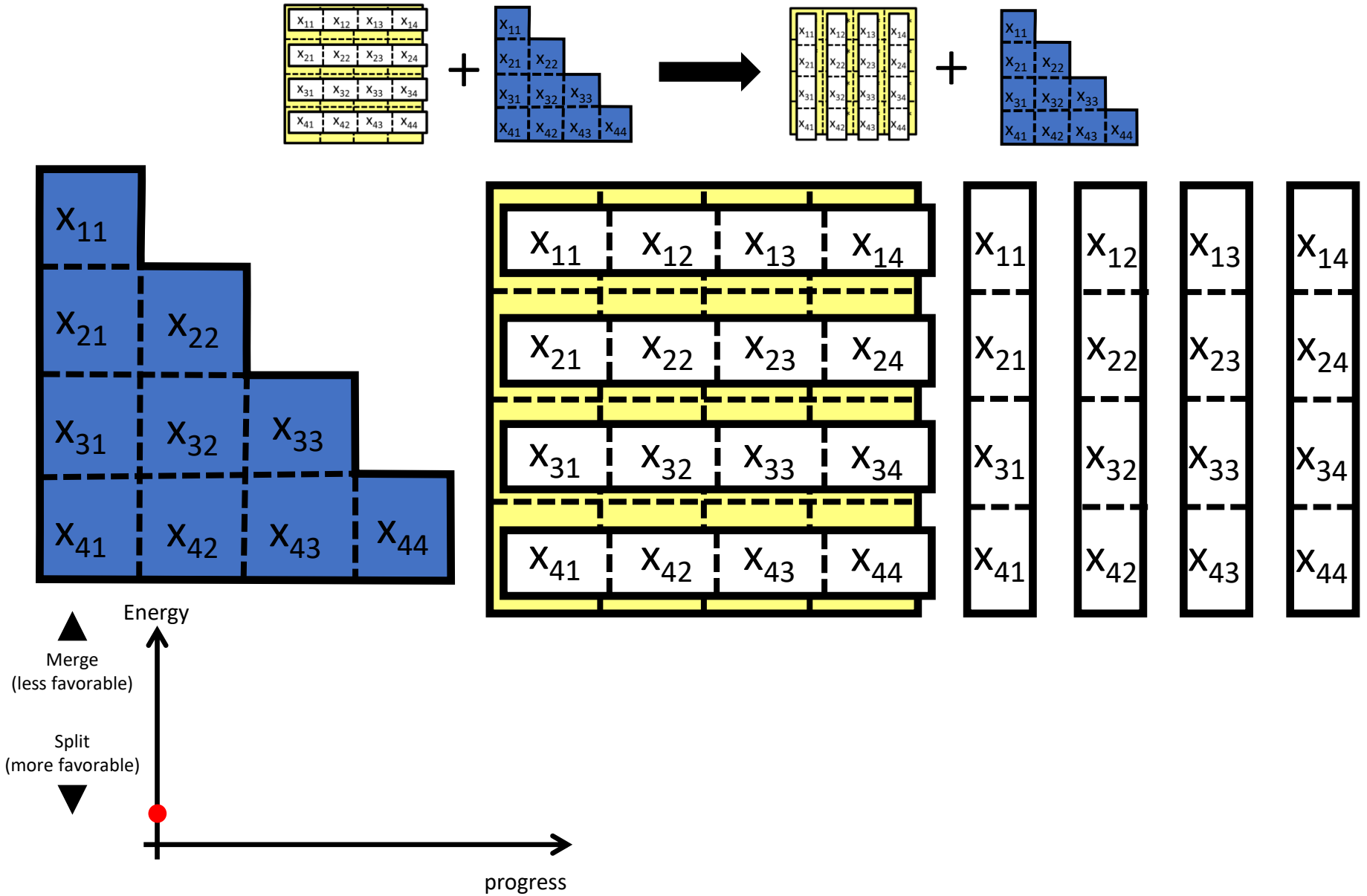


A Network with a Programmable Energy Barrier

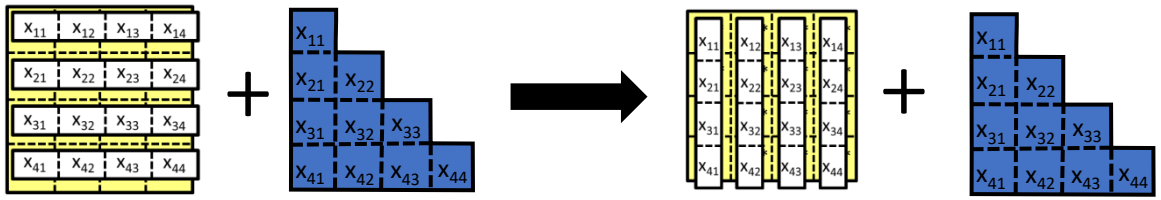


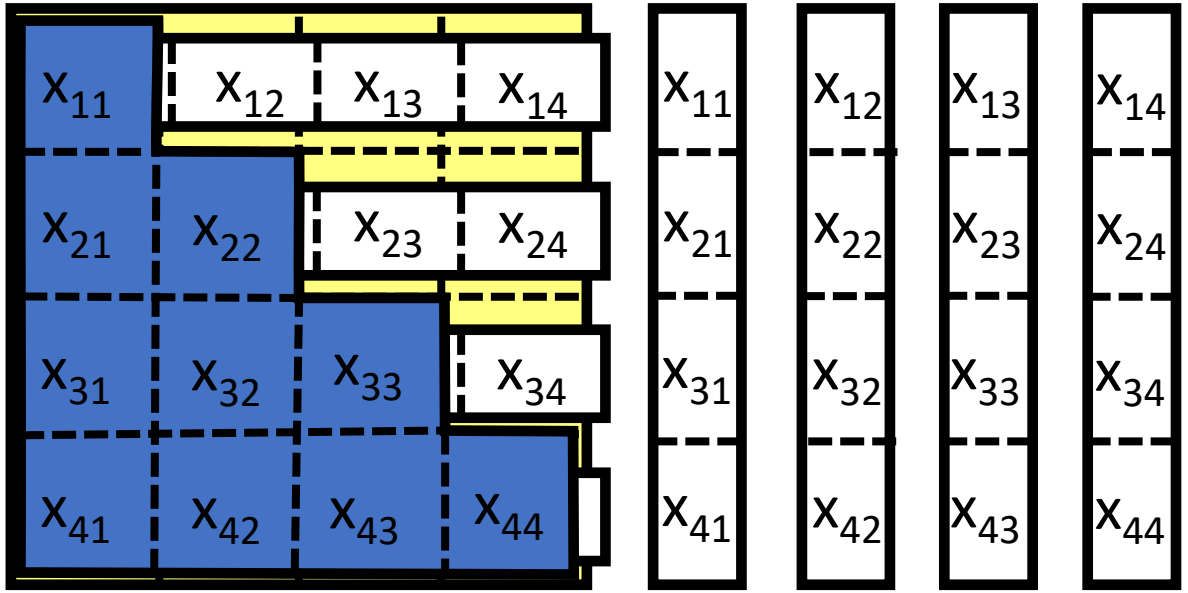


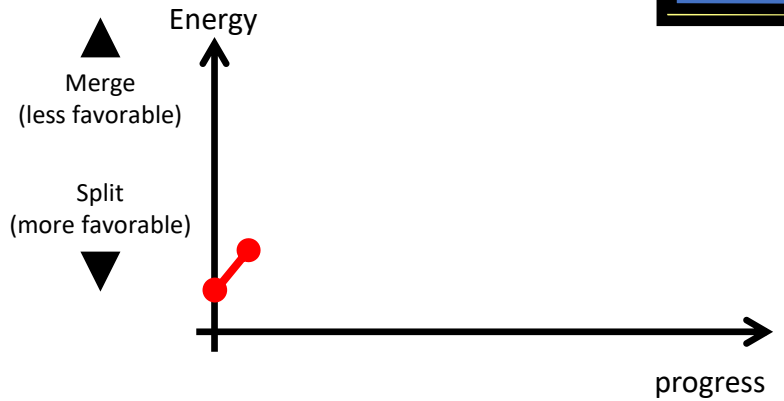
Catalysis


Energy
Merge
(less favorable)
Split
(more favorable)
progress

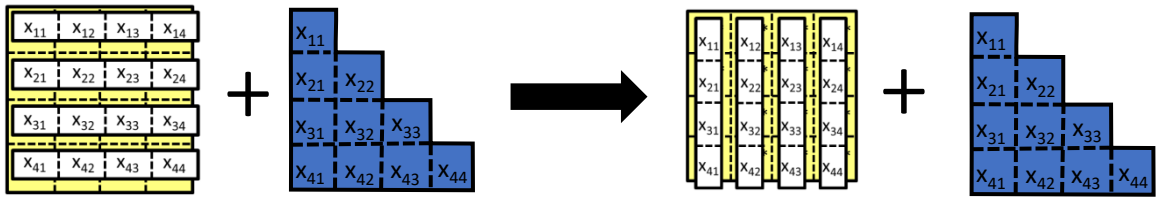
Catalysis

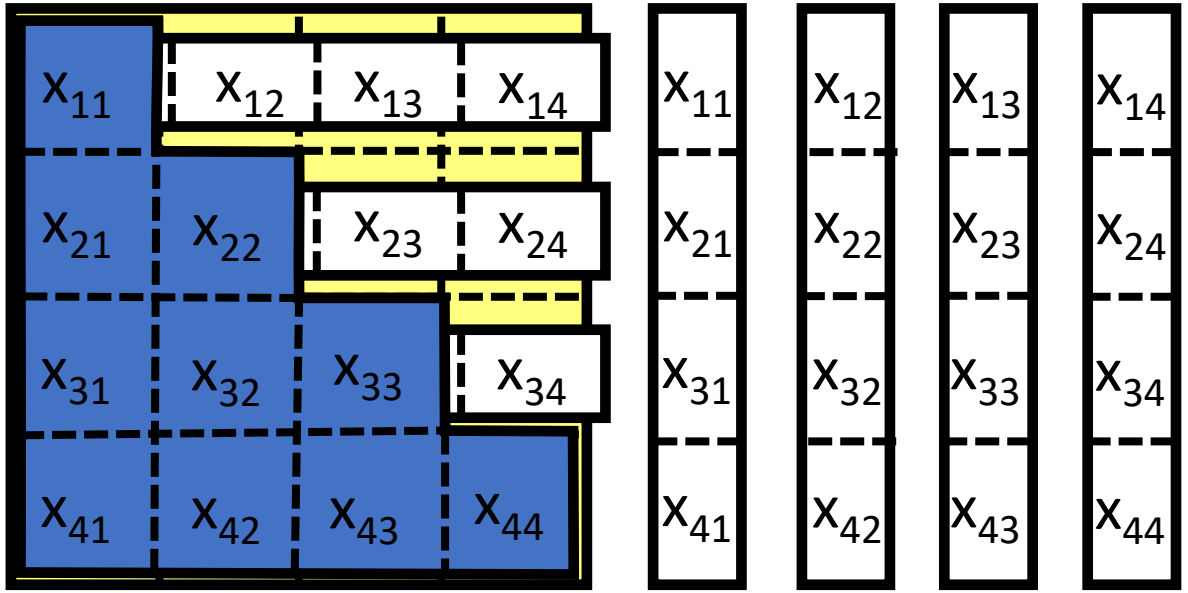

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

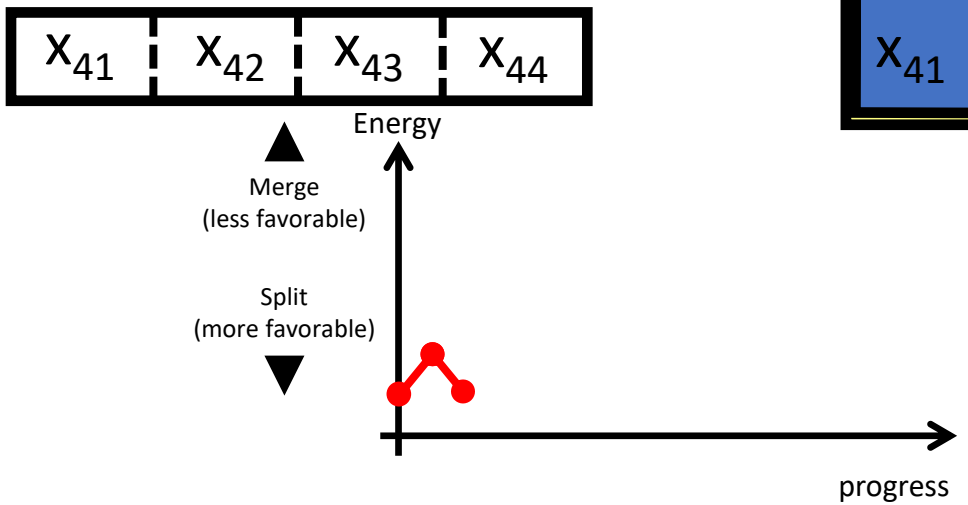

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44


Energy
Merge
(less favorable)
Split
(more favorable)
progress

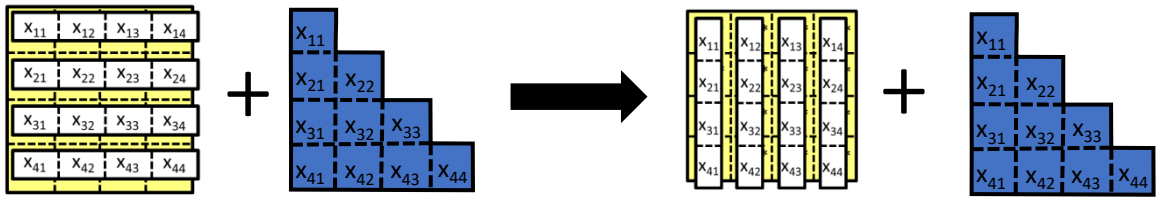
Catalysis

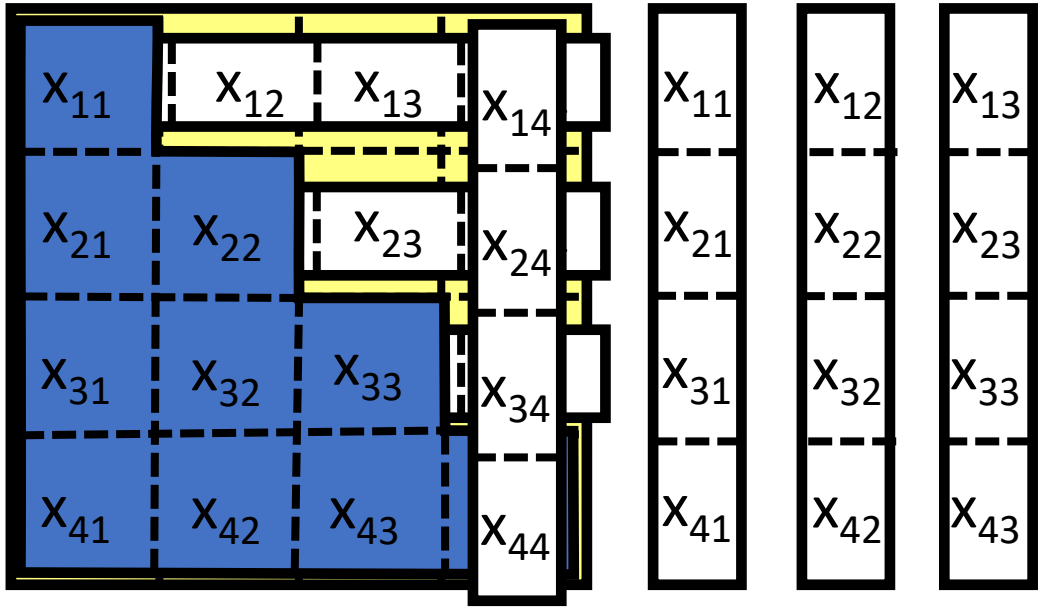

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

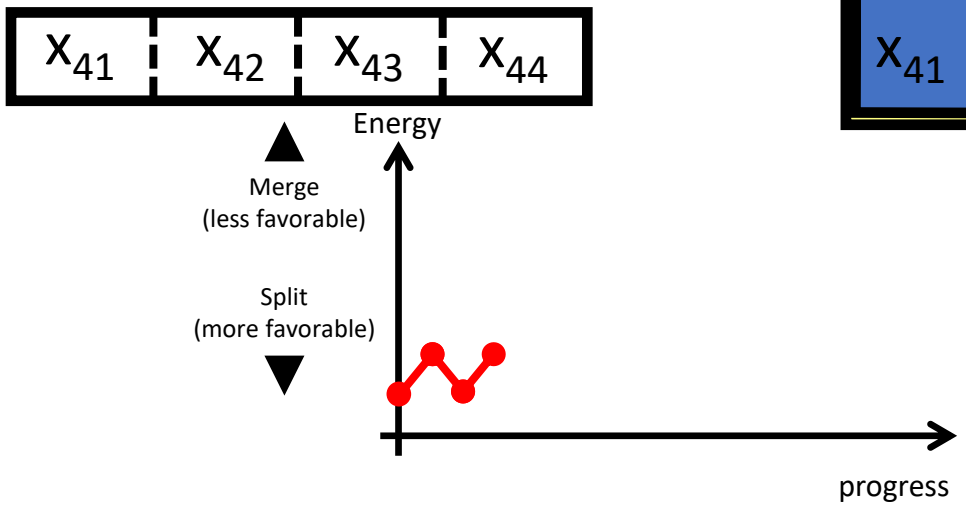

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44


x41 x42 x43 x44
Energy
Merge
(less favorable)
Split
(more favorable)
progress

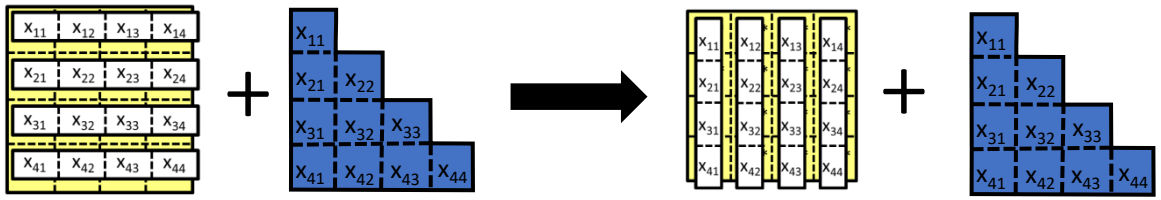
Catalysis

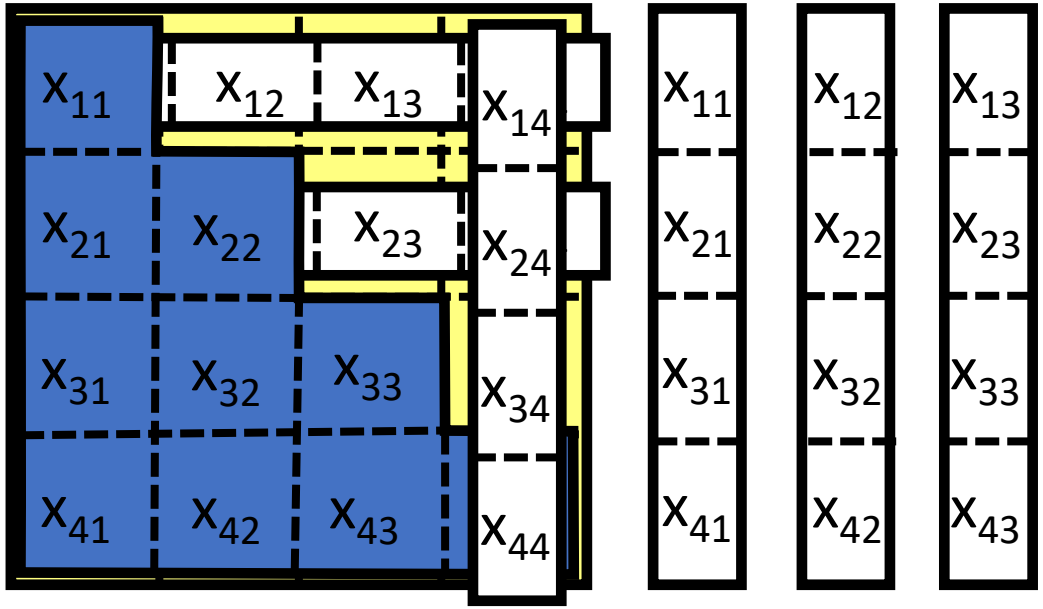

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11
x21 x22
x31 x32 x33
x41 x42 x43 x44
x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11
x21 x22
x31 x32 x33
x41 x42 x43 x44

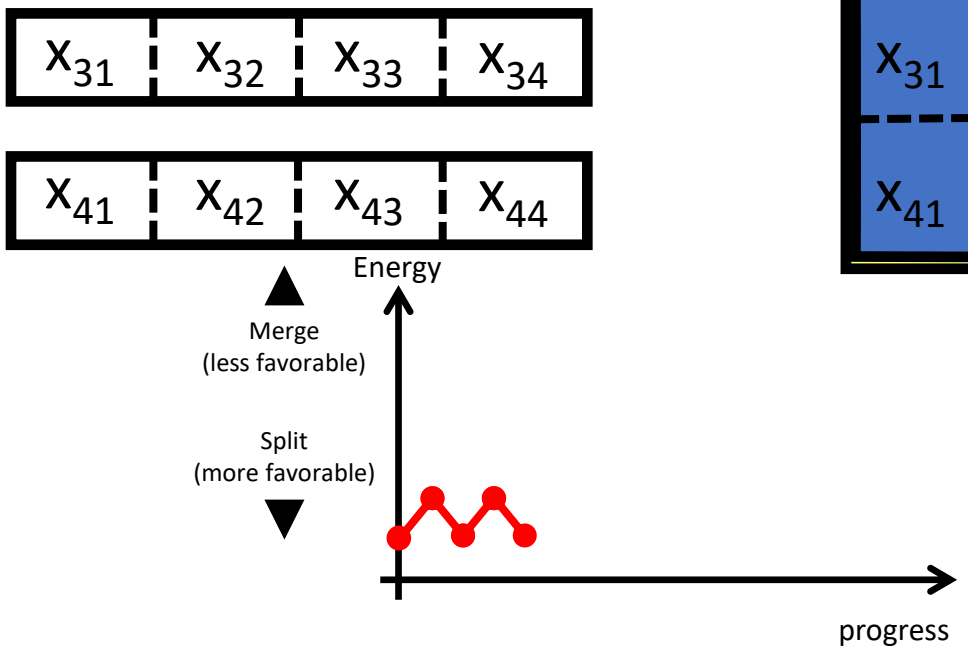

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11 x12 x13
x21 x22 x23
x31 x32 x33
x41 x42 x43
x14 x24 x34 x44
x11 x21 x31 x41
x12 x22 x32 x42
x13 x23 x33 x43


x41 x42 x43 x44
Energy
Merge
(less favorable)
Split
(more favorable)
progress

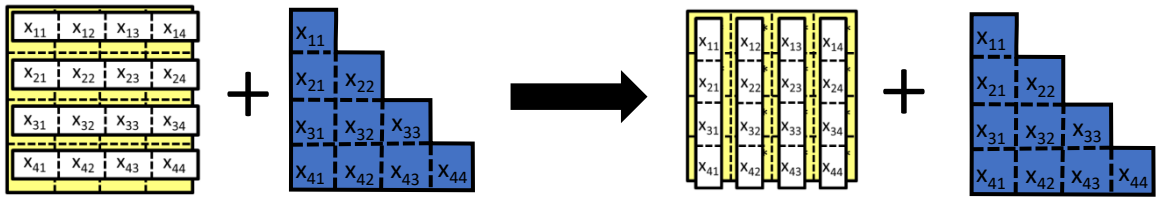
Catalysis

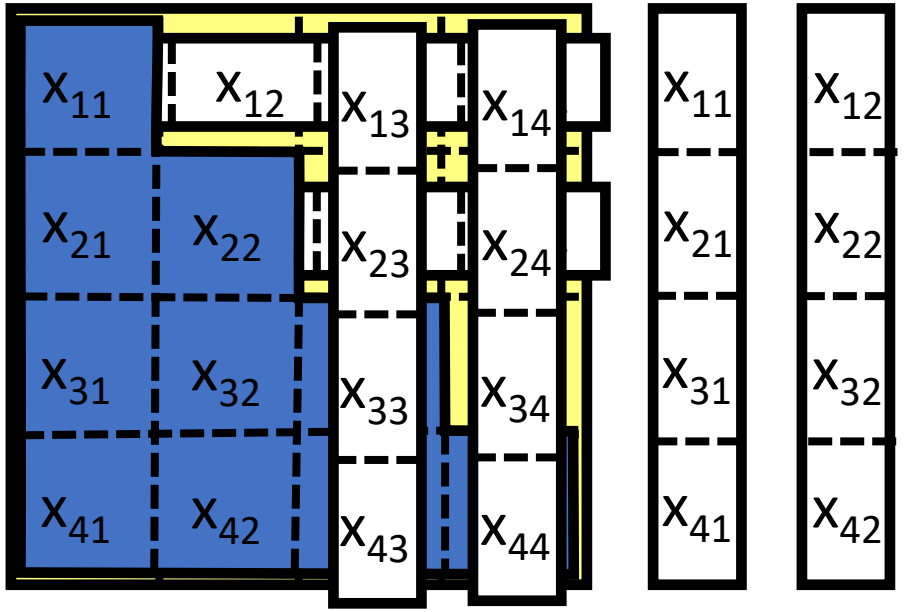

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
+
x11
x21 x22
x31 x32 x33
x41 x42 x43 x44
x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
+
x11
x21 x22
x31 x32 x33
x41 x42 x43 x44

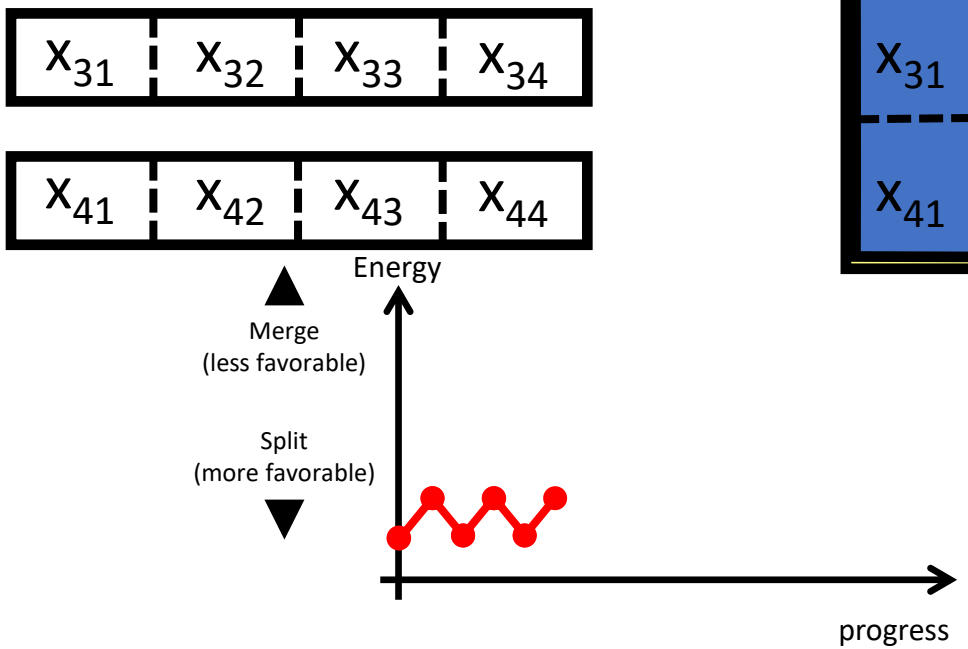

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11 x12 x13
x21 x22 x23
x31 x32 x33
x41 x42 x43
x11 x12 x13
x21 x22 x23
x31 x32 x33
x41 x42 x43


x31 x32 x33 x34
x41 x42 x43 x44
Energy
Merge
(less favorable)
Split
(more favorable)
progress

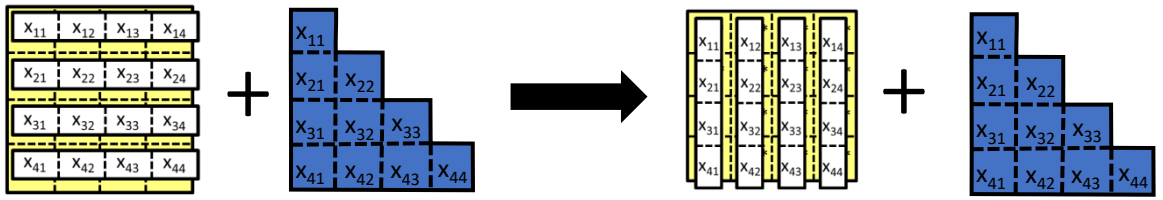
Catalysis

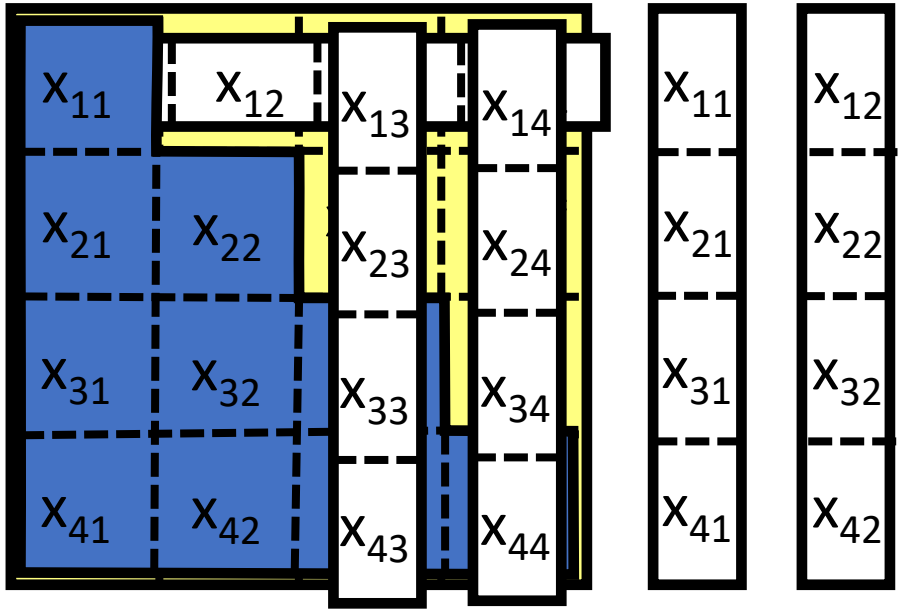

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

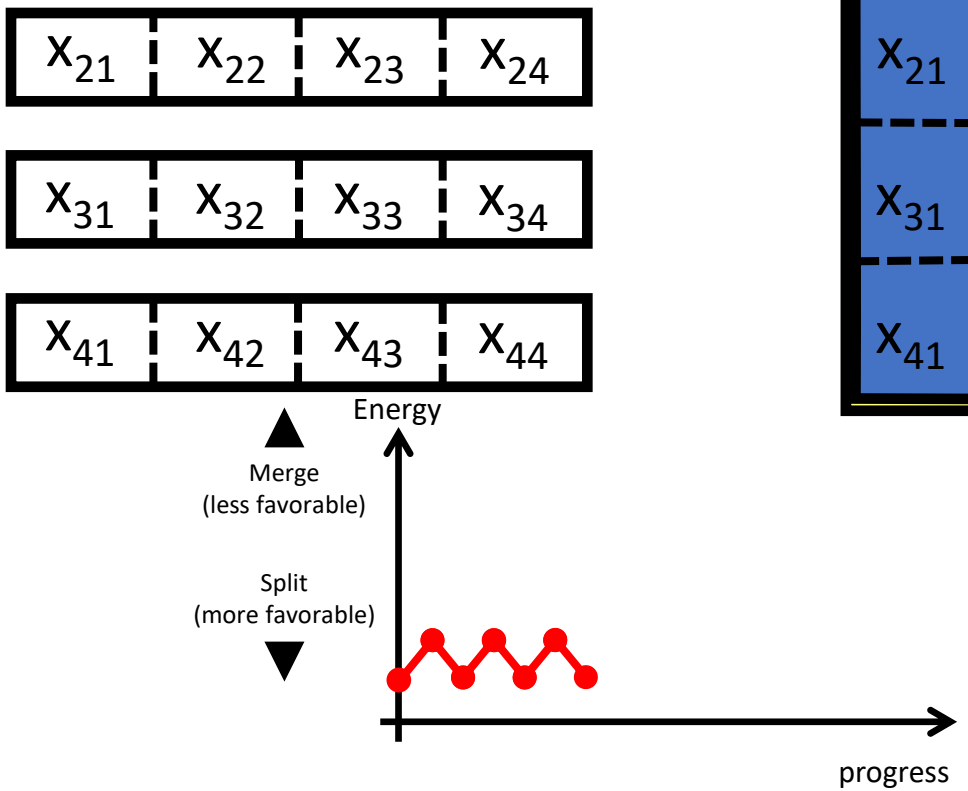

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11 x12
x21 x22
x31 x32
x41 x42


x31 x32 x33 x34
x41 x42 x43 x44
Energy
Merge
(less favorable)
Split
(more favorable)
progress

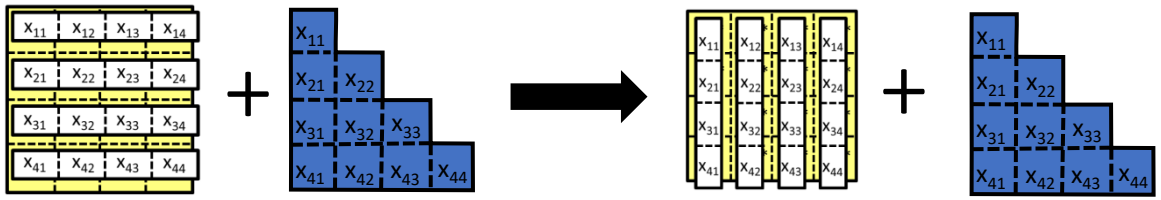
Catalysis

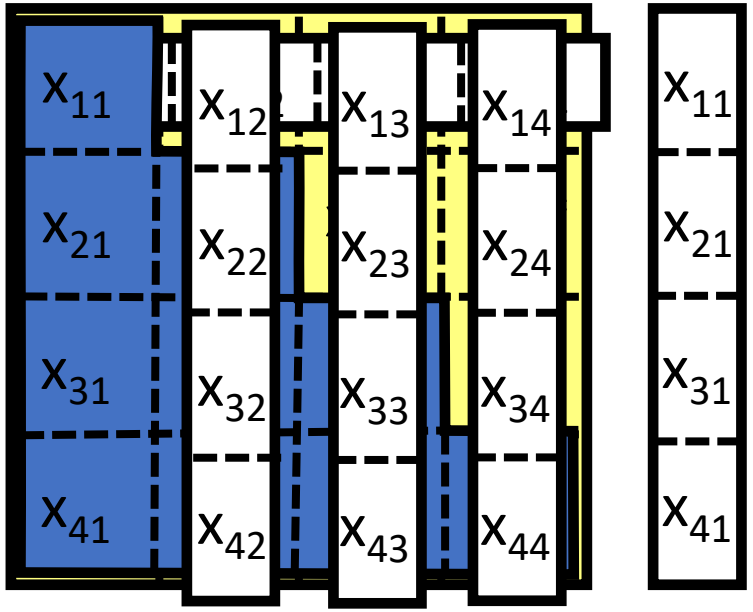

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

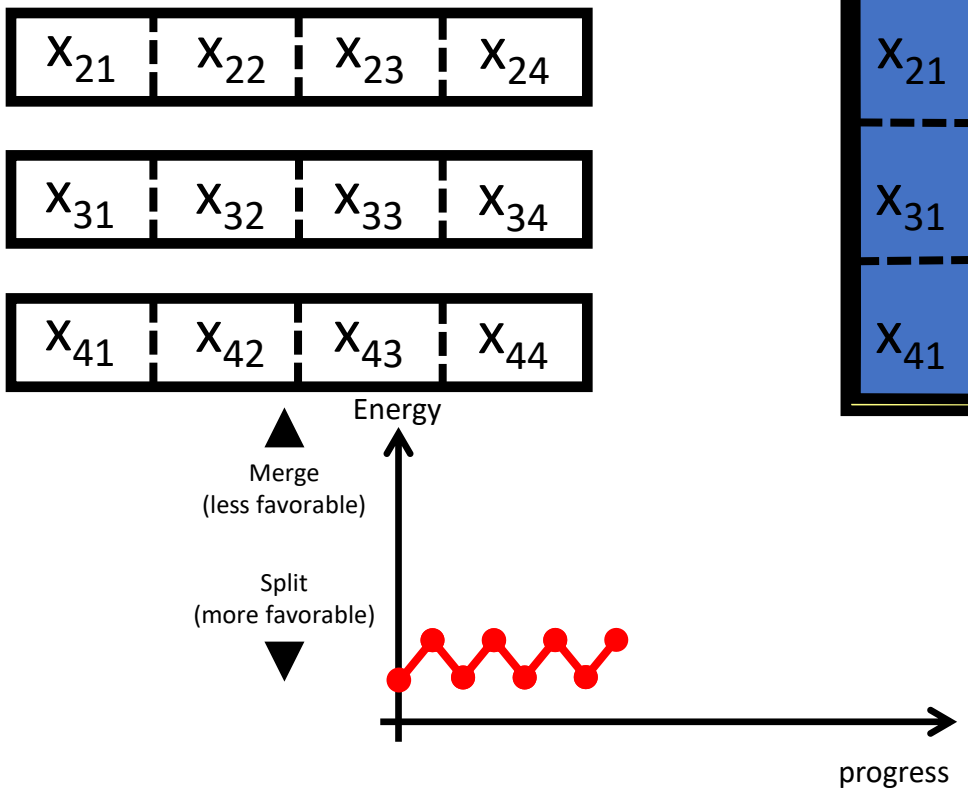

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11 x12
x21 x22
x31 x32
x41 x42


x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
Energy
Merge
(less favorable)
Split
(more favorable)
progress

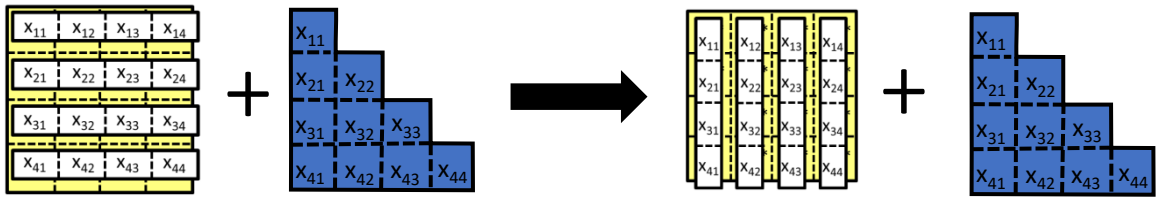
Catalysis

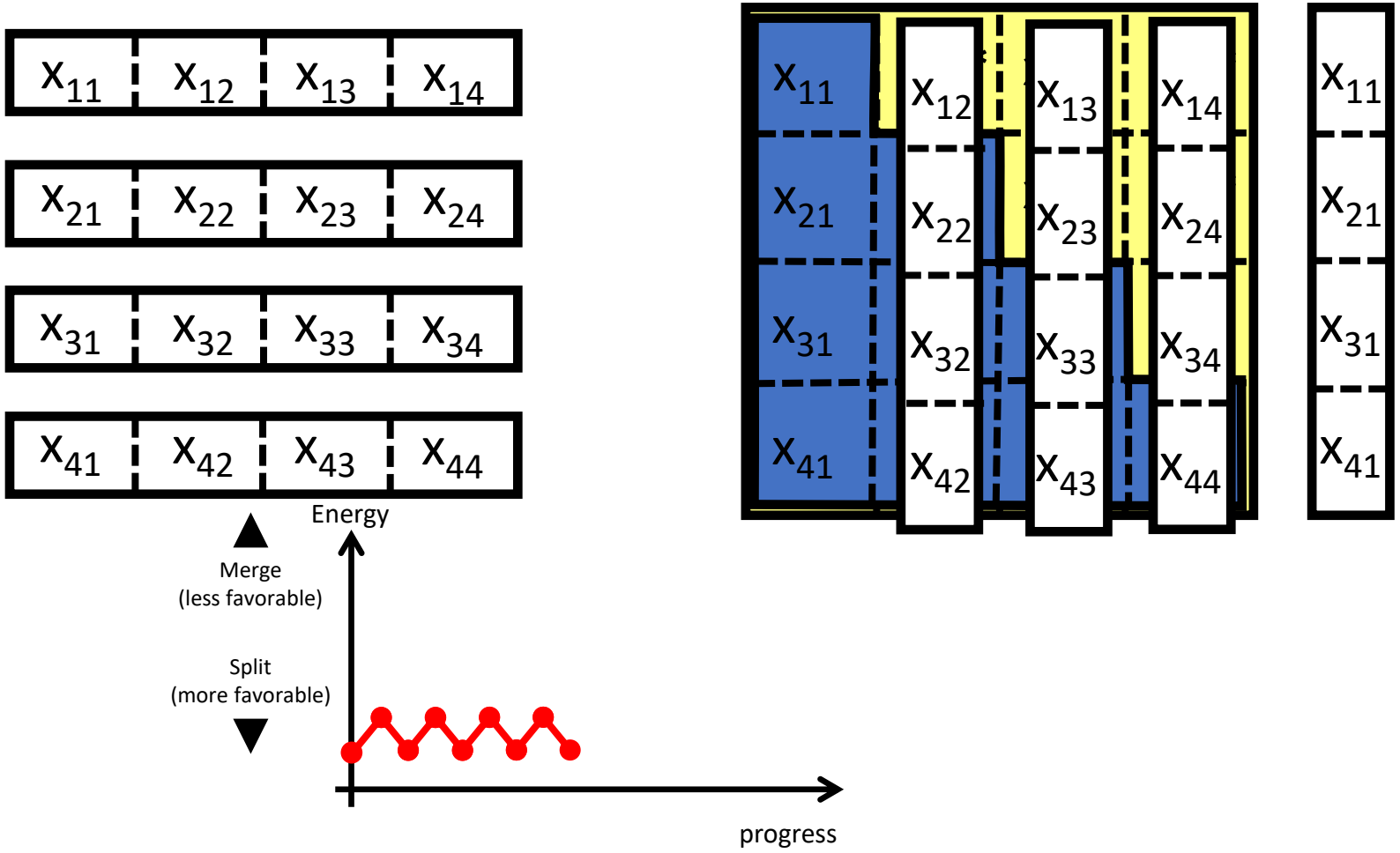

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
+
x11
x21 x22
x31 x32 x33
x41 x42 x43 x44
x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
+
x11
x21 x22
x31 x32 x33
x41 x42 x43 x44


x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
x11
x21
x31
x41


x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
Energy
Merge
(less favorable)
Split
(more favorable)
progress

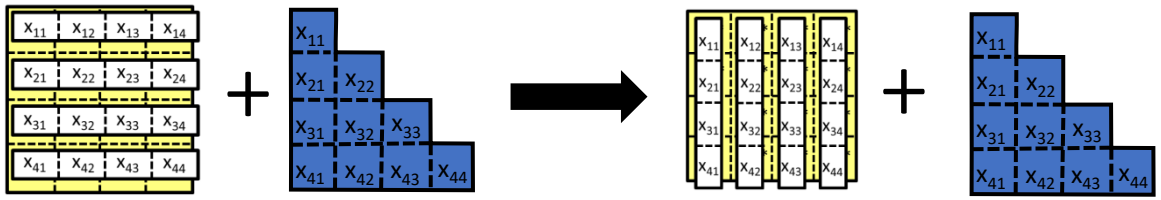
Catalysis

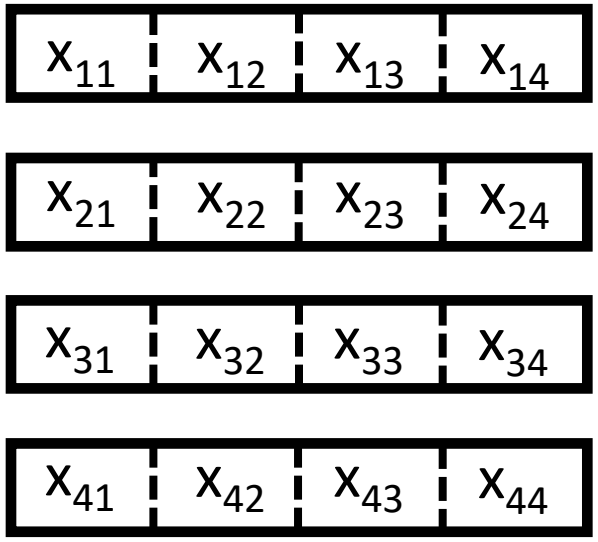

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

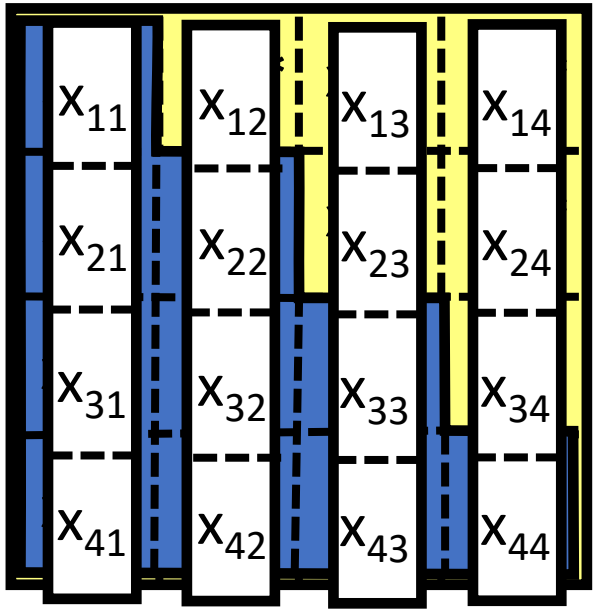

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44
Energy
Merge
(less favorable)
Split
(more favorable)
progress

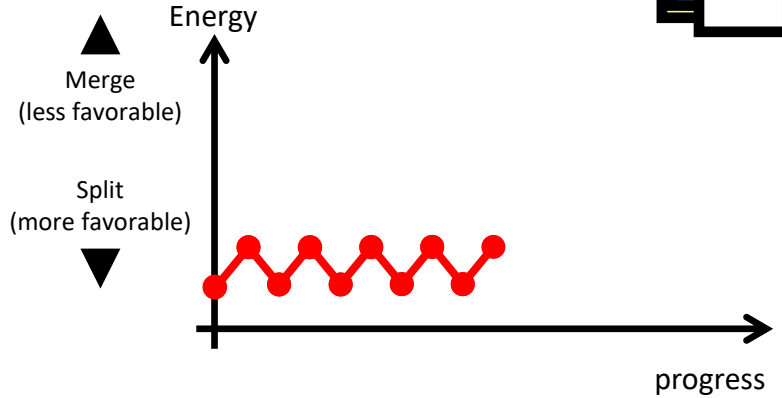
x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

Catalysis

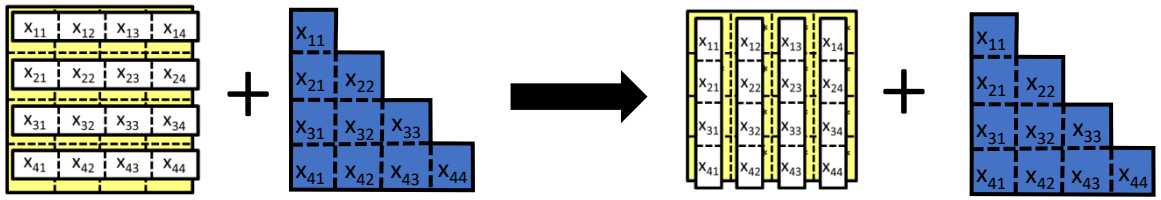

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

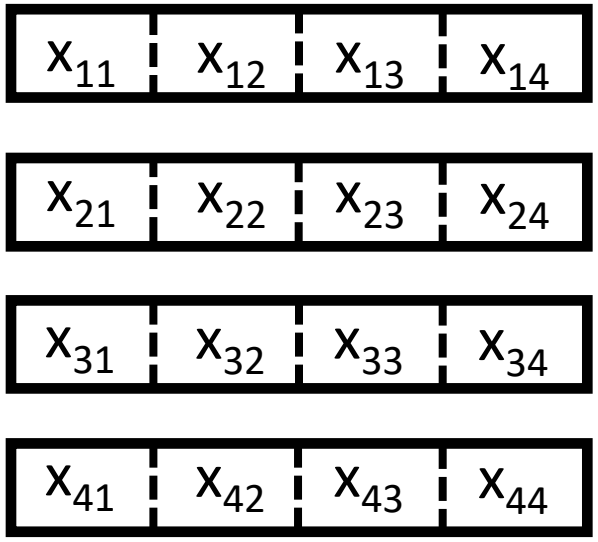

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

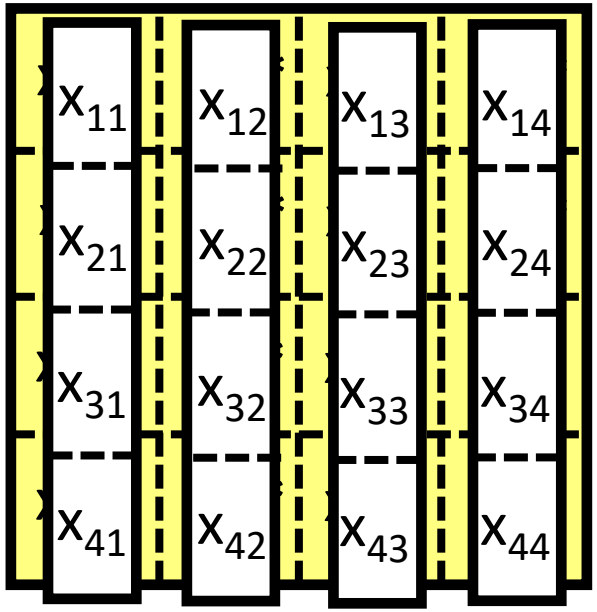

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

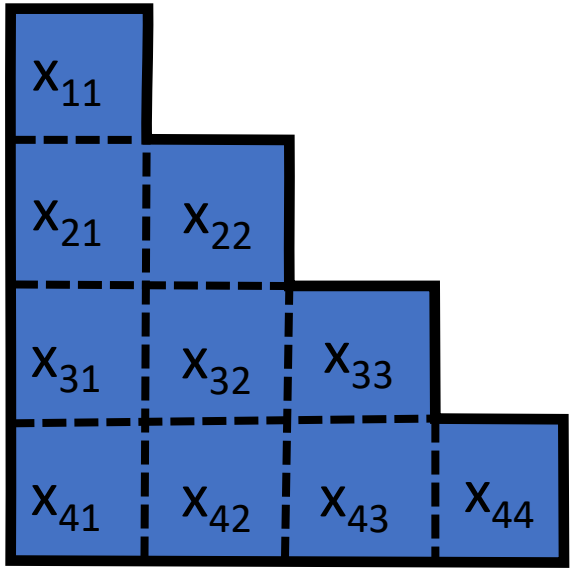

Energy
Merge
(less favorable)
Split
(more favorable)
progress

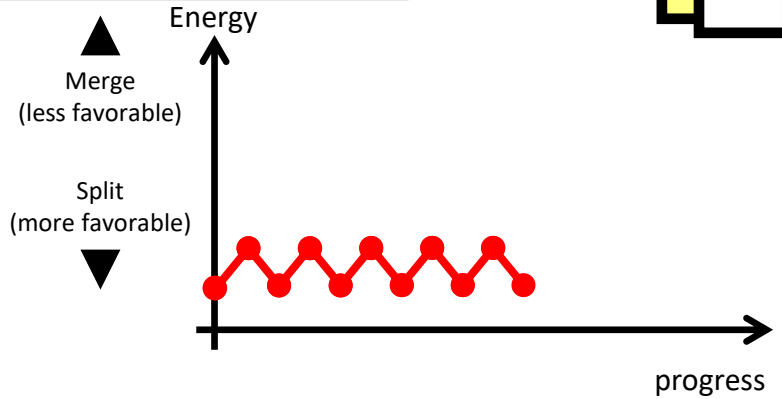
Catalysis


x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

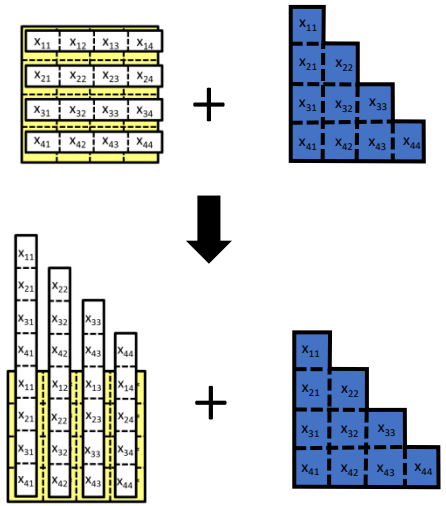

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

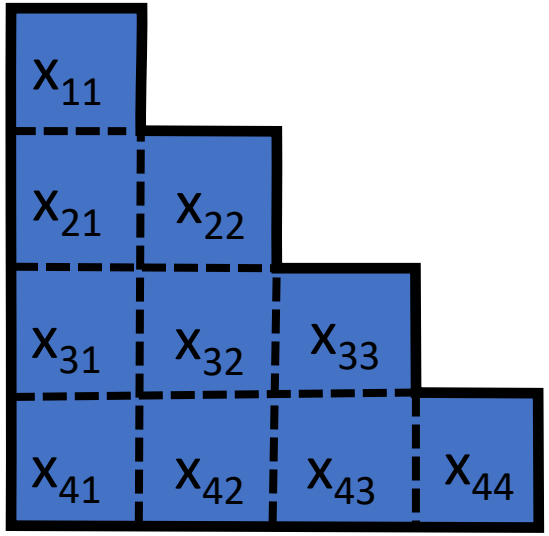

x11 x12 x13 x14
x21 x22 x23 x24
x31 x32 x33 x34
x41 x42 x43 x44

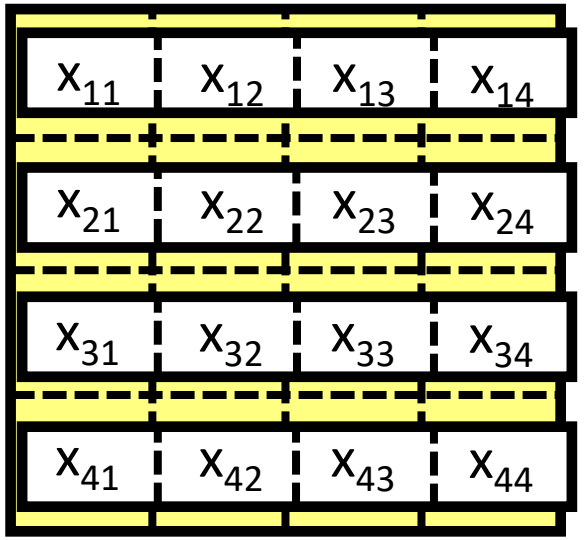

x11
x21 x22
x31 x32 x33
x41 x42 x43 x44

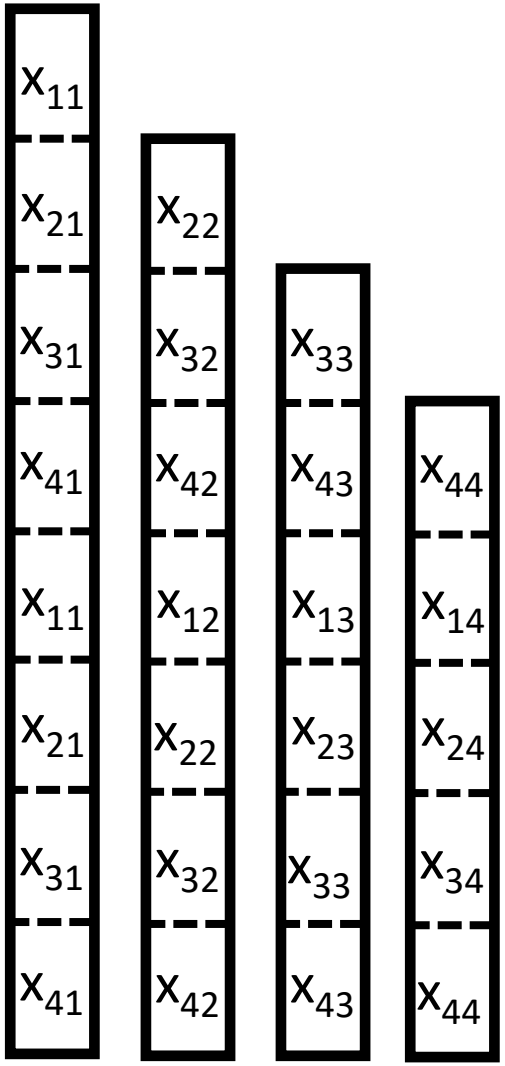

Energy
Merge
(less favorable)
Split
(more favorable)
progress

Autocatalysis

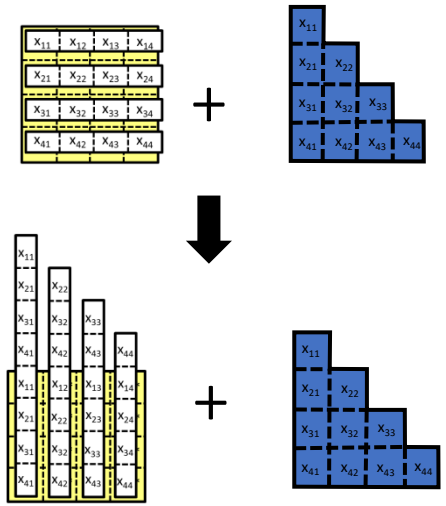


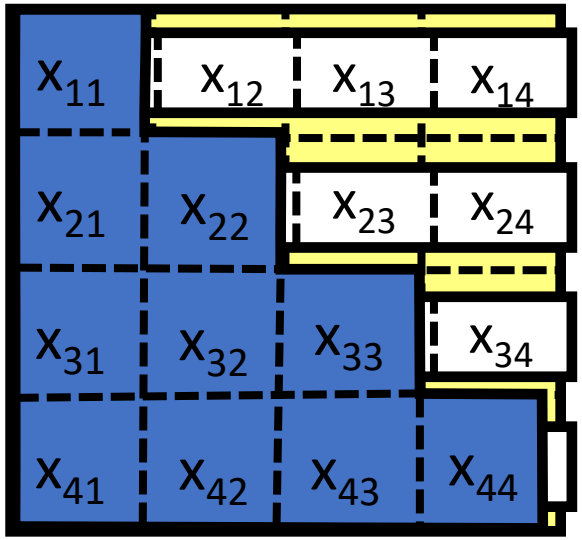


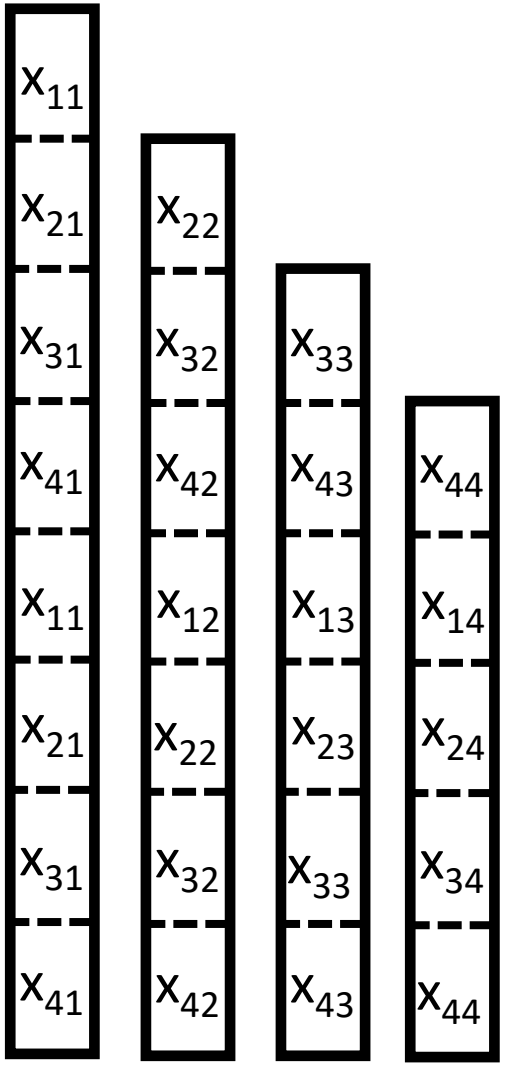




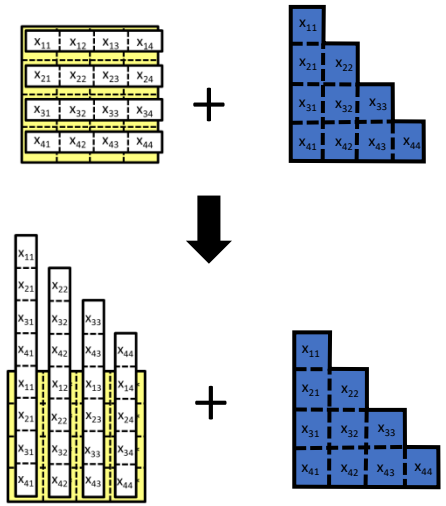
Autocatalysis

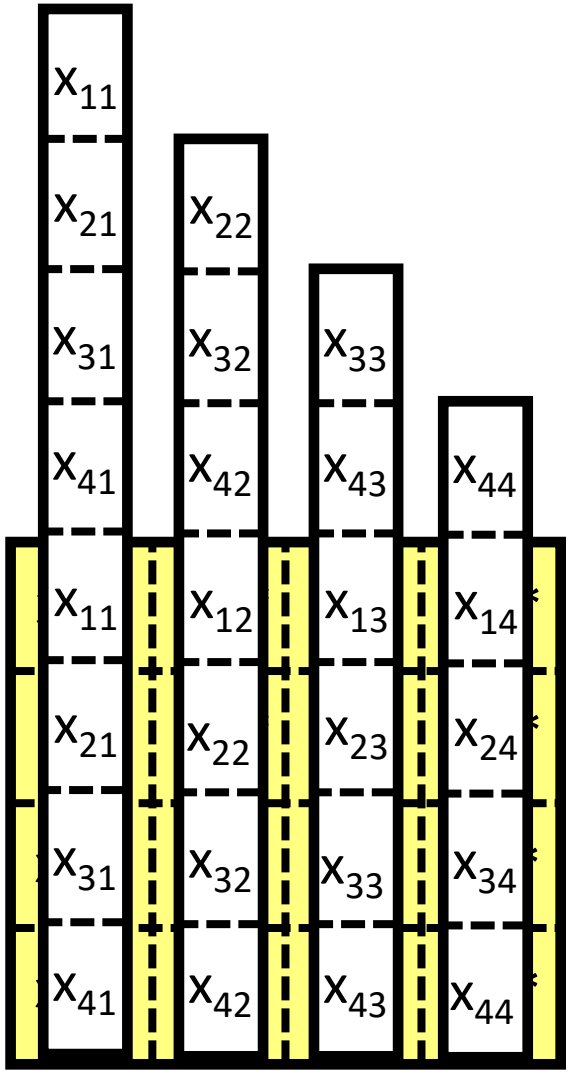


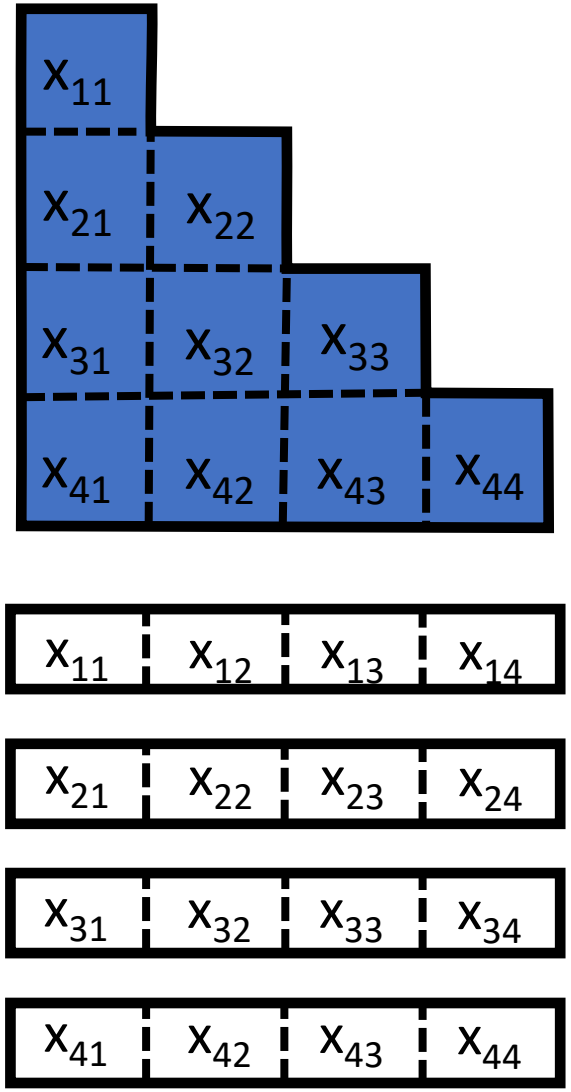




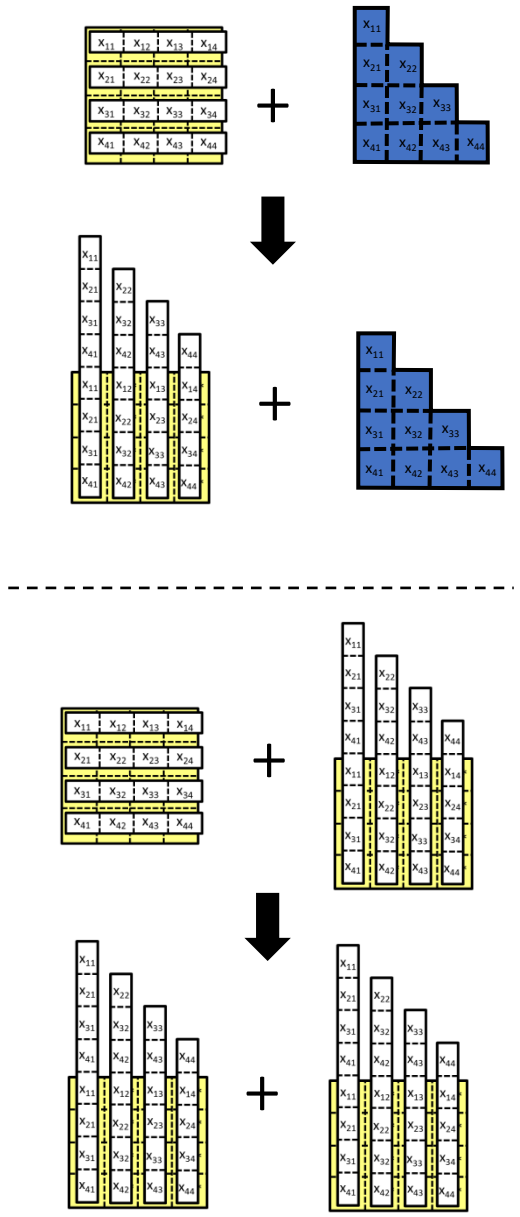
Autocatalysis

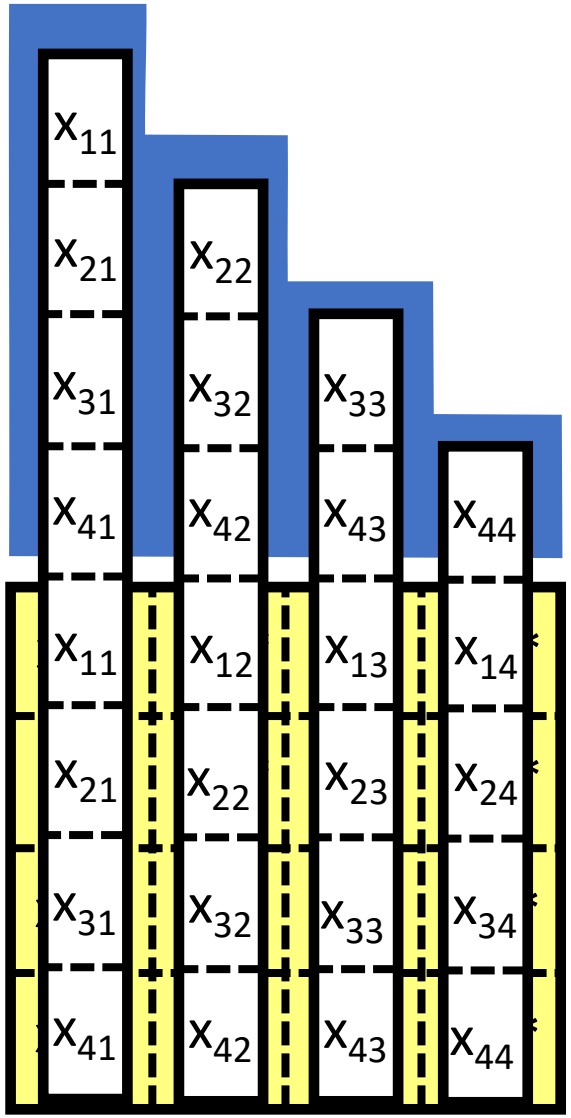


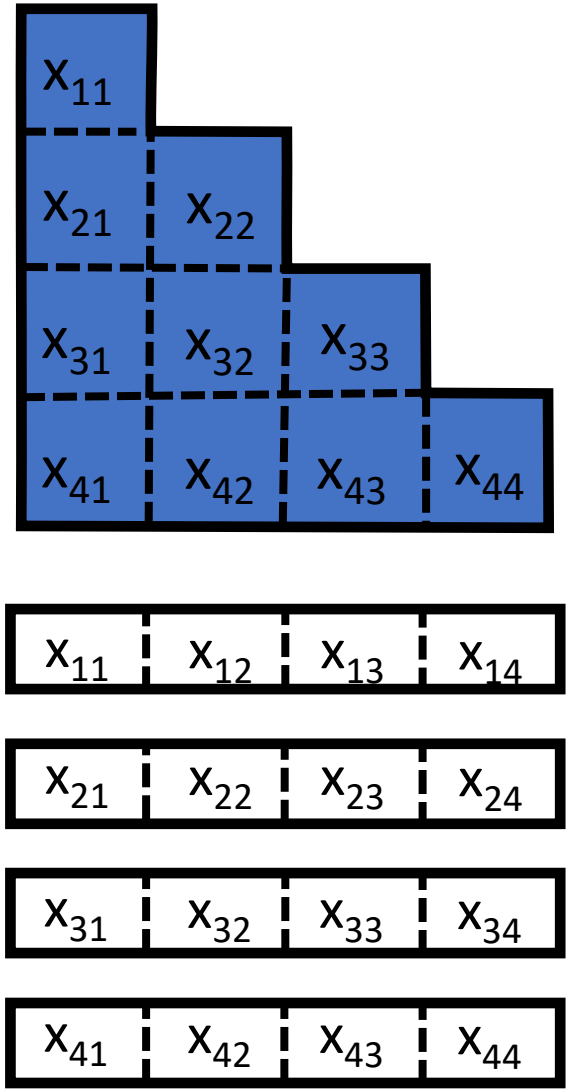




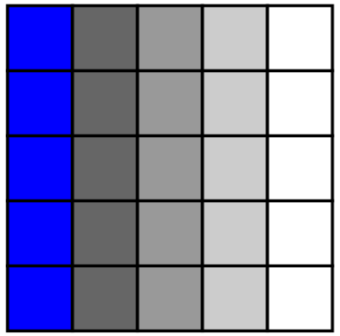
Autocatalysis

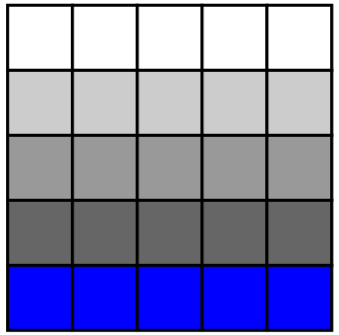




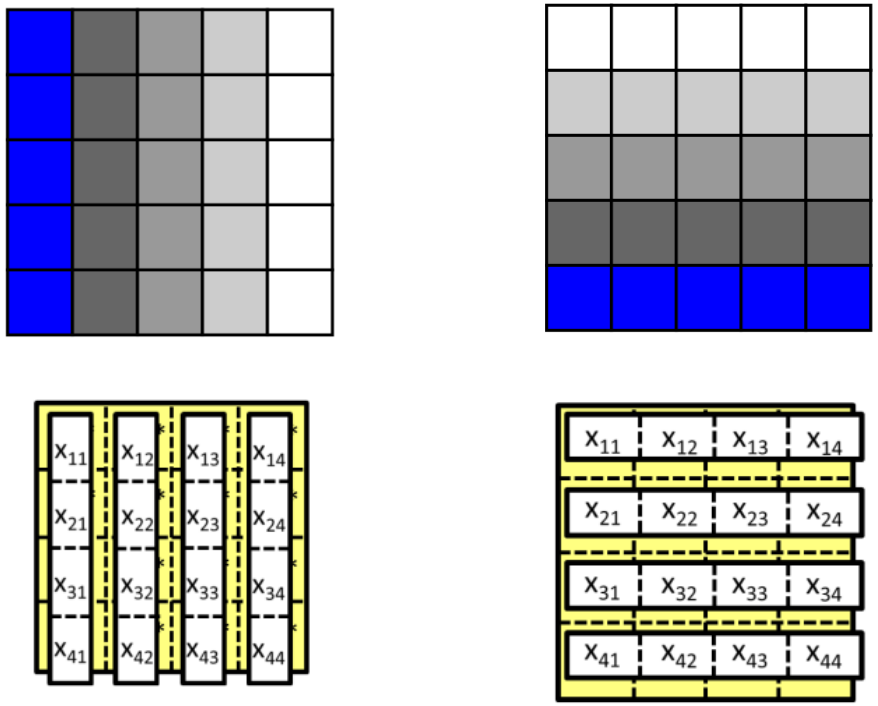


Multiple Stable Configurations

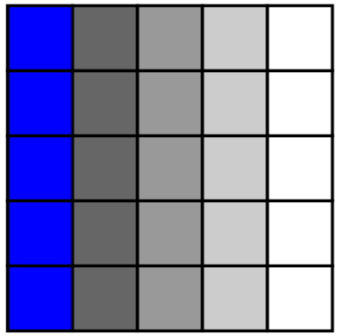


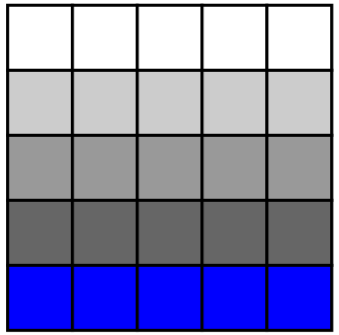


Multiple Stable Configurations

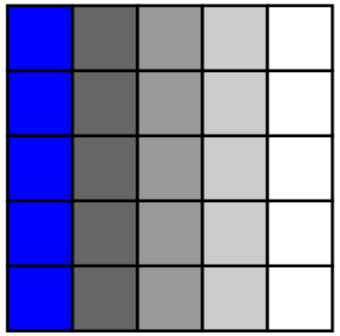


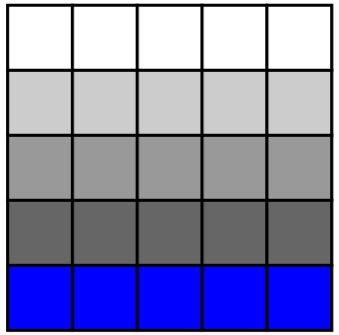
Multiple Stable Configurations

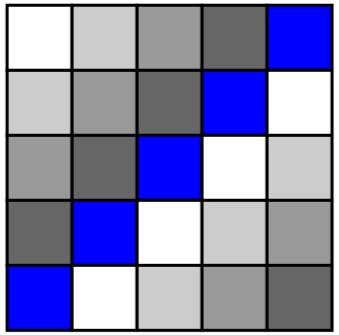




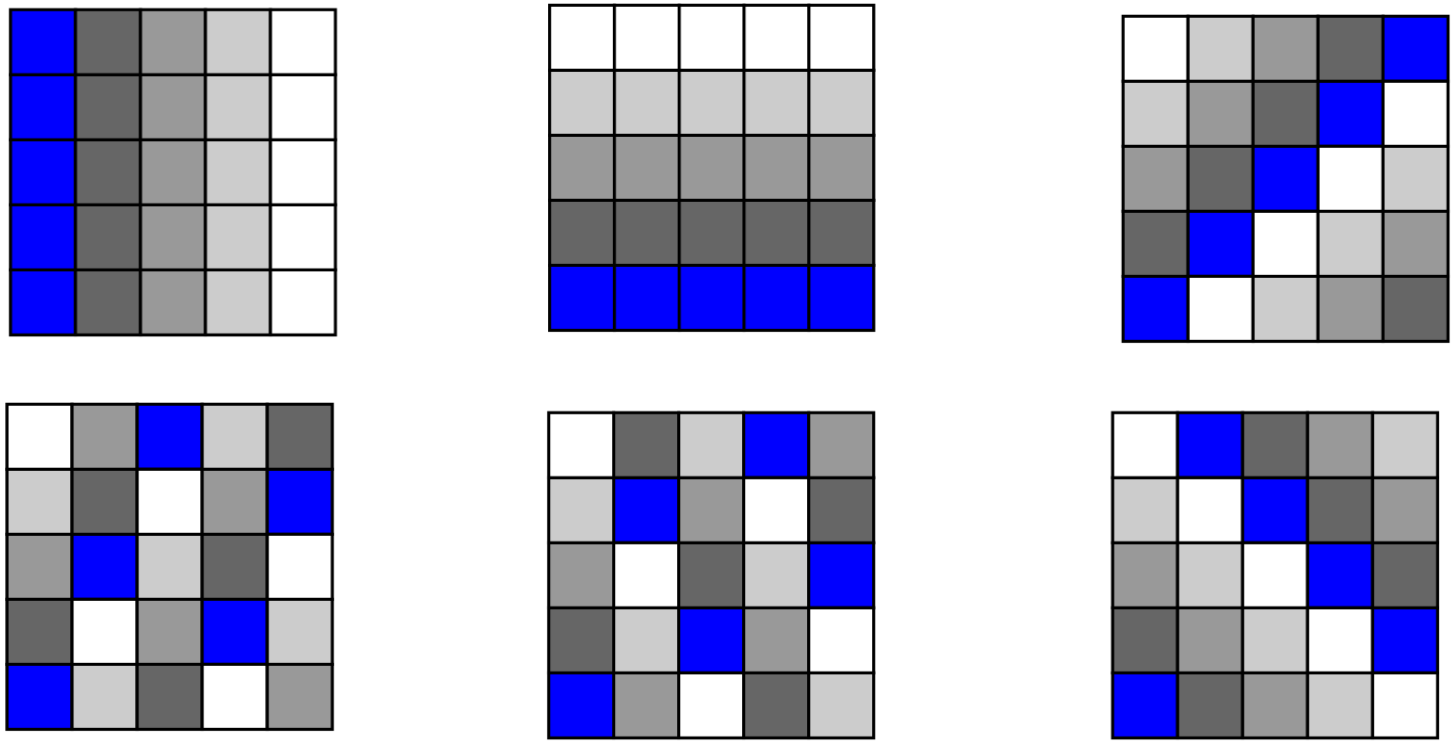
Multiple Stable Configurations





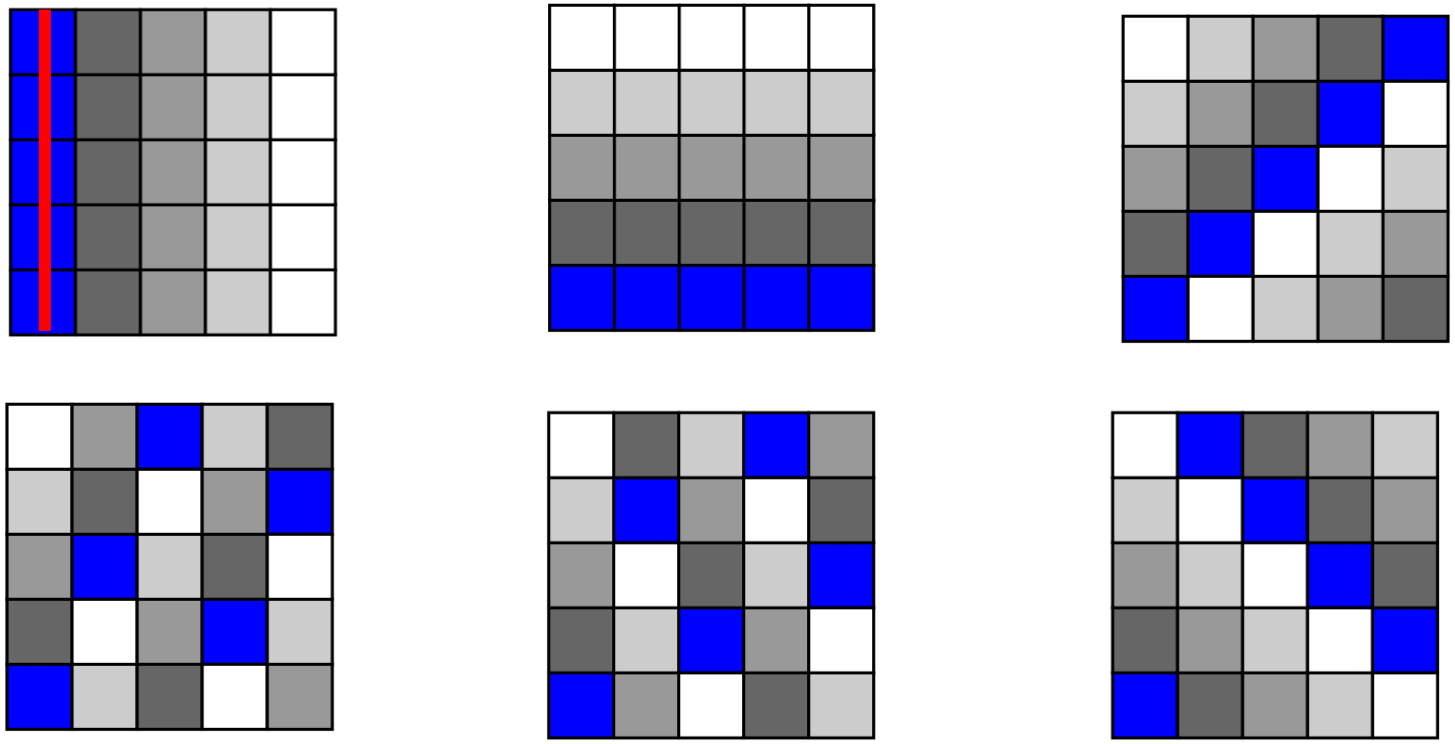


Multiple Stable Configurations



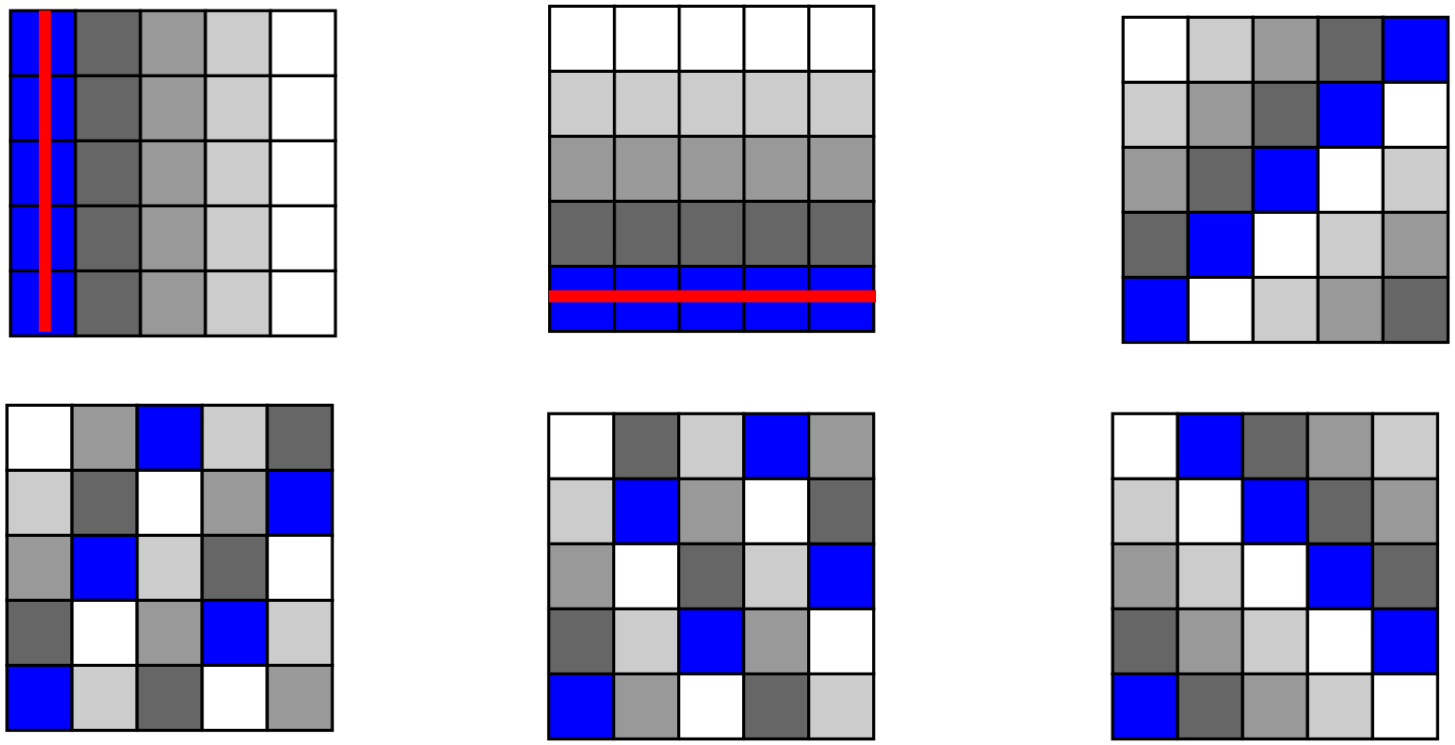
For a grid of prime size *n* x *n*, there can be at most *n*+1 different stable configurations with barrier *n* to pass between any of them

Multiple Stable Configurations



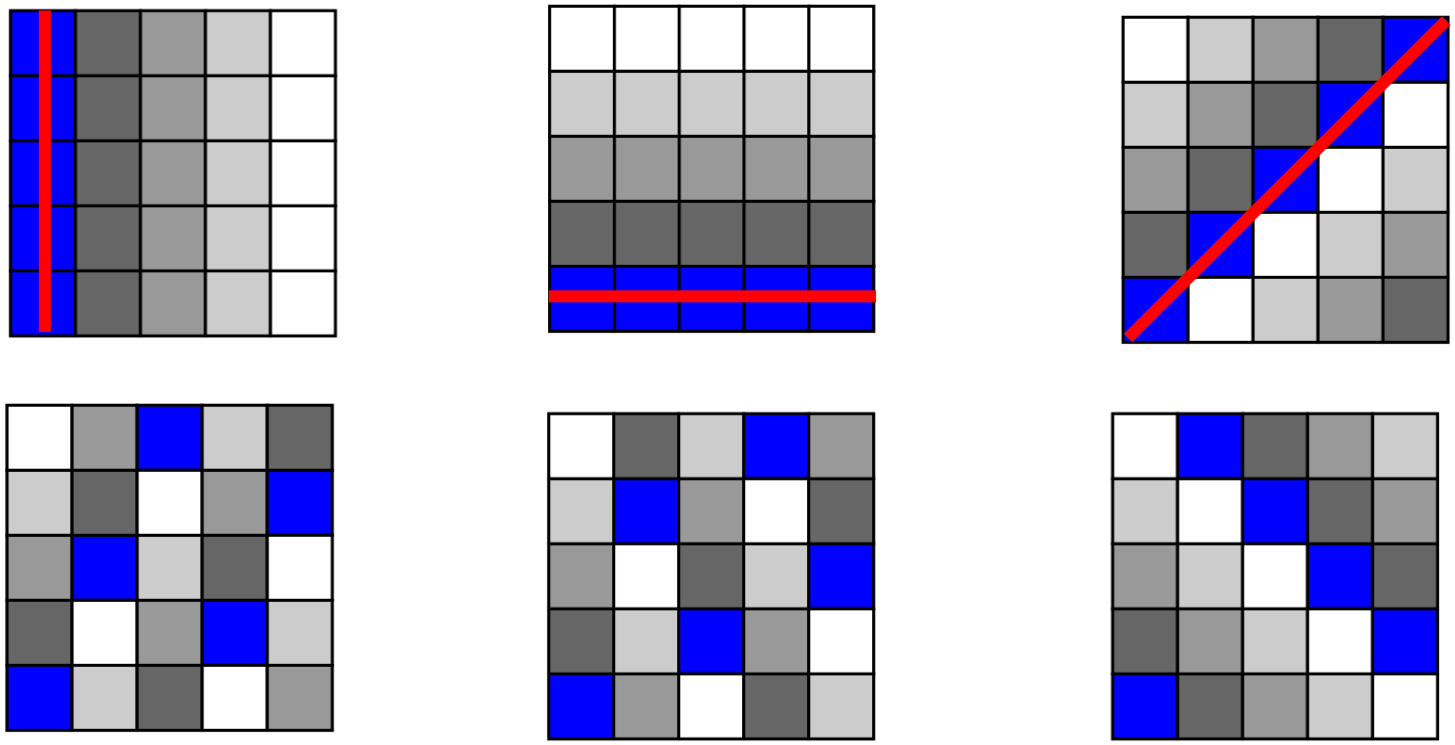
For a grid of prime size *n* x *n*, there can be at most *n*+1 different stable configurations with barrier *n* to pass between any of them

Multiple Stable Configurations



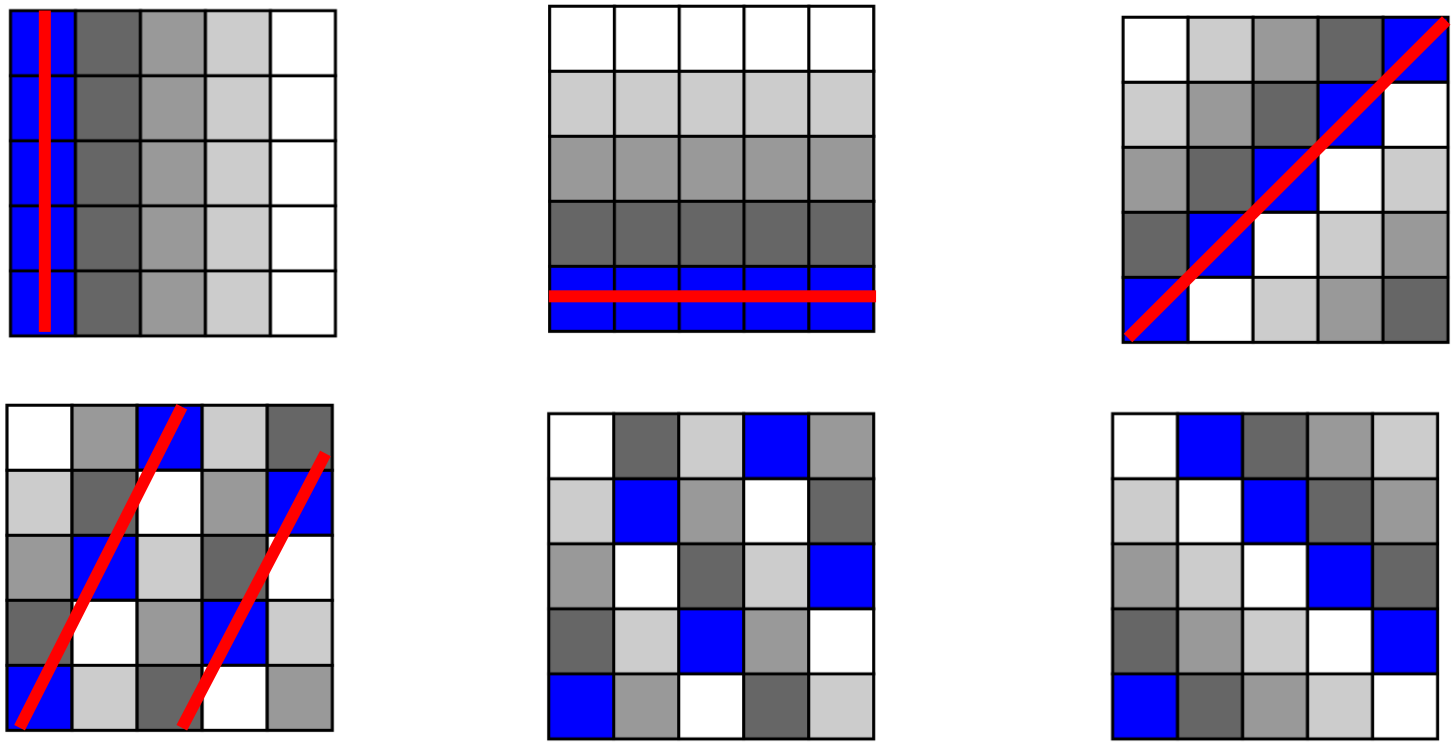
For a grid of prime size *n* x *n*, there can be at most *n*+1 different stable configurations with barrier *n* to pass between any of them

Multiple Stable Configurations



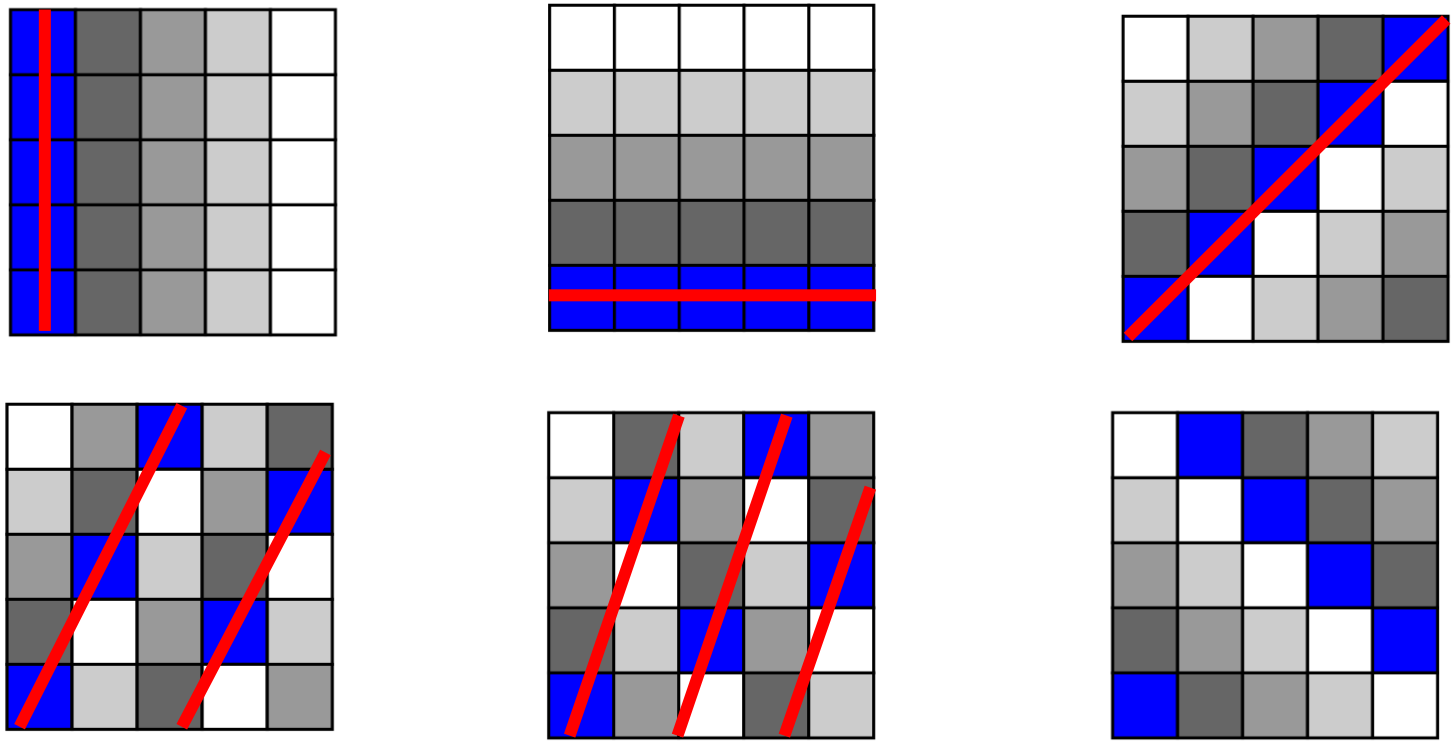
For a grid of prime size *n* x *n*, there can be at most *n*+1 different stable configurations with barrier *n* to pass between any of them

Multiple Stable Configurations



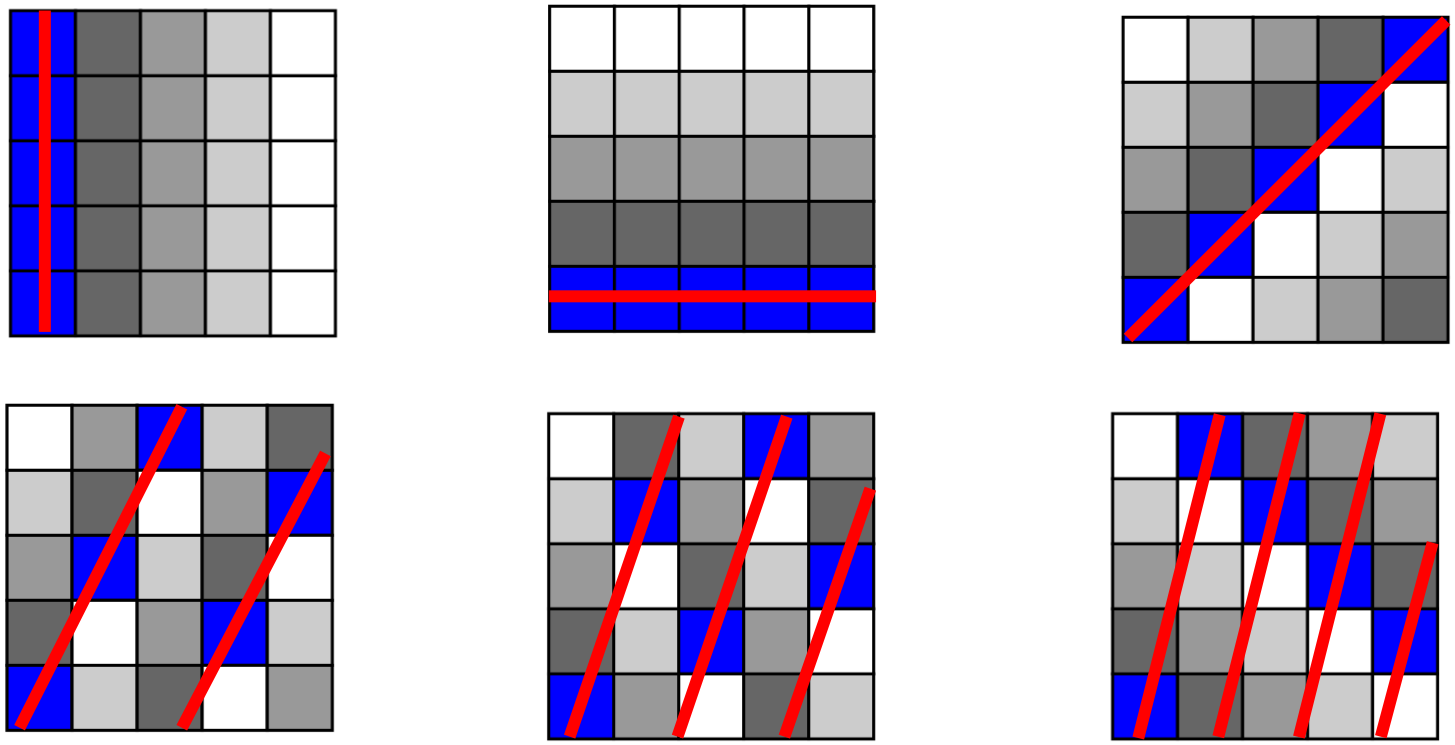
For a grid of prime size *n* x *n*, there can be at most *n*+1 different stable configurations with barrier *n* to pass between any of them

Multiple Stable Configurations



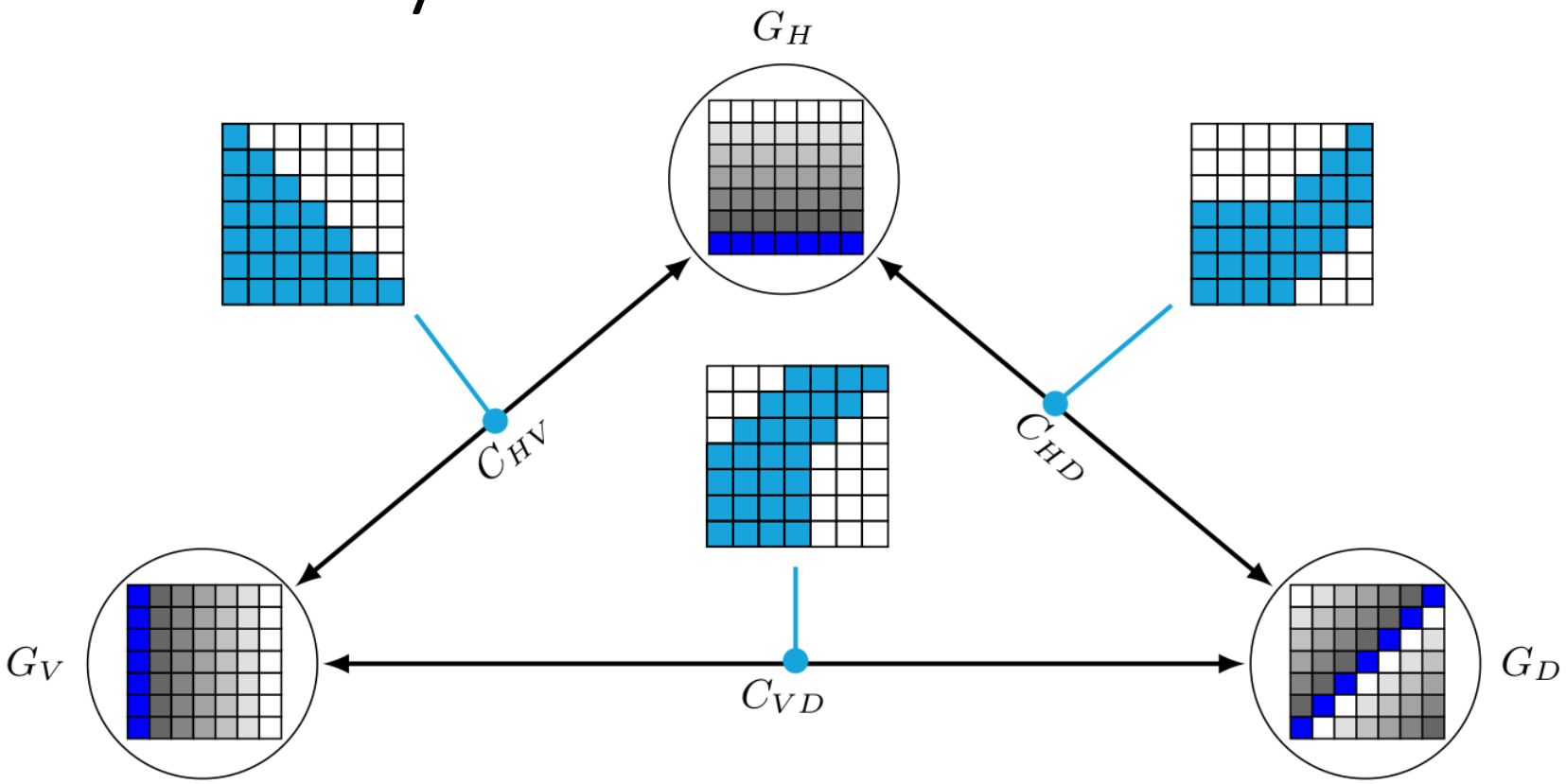
For a grid of prime size *n* x *n*, there can be at most *n*+1 different stable configurations with barrier *n* to pass between any of them

Multiple Stable Configurations

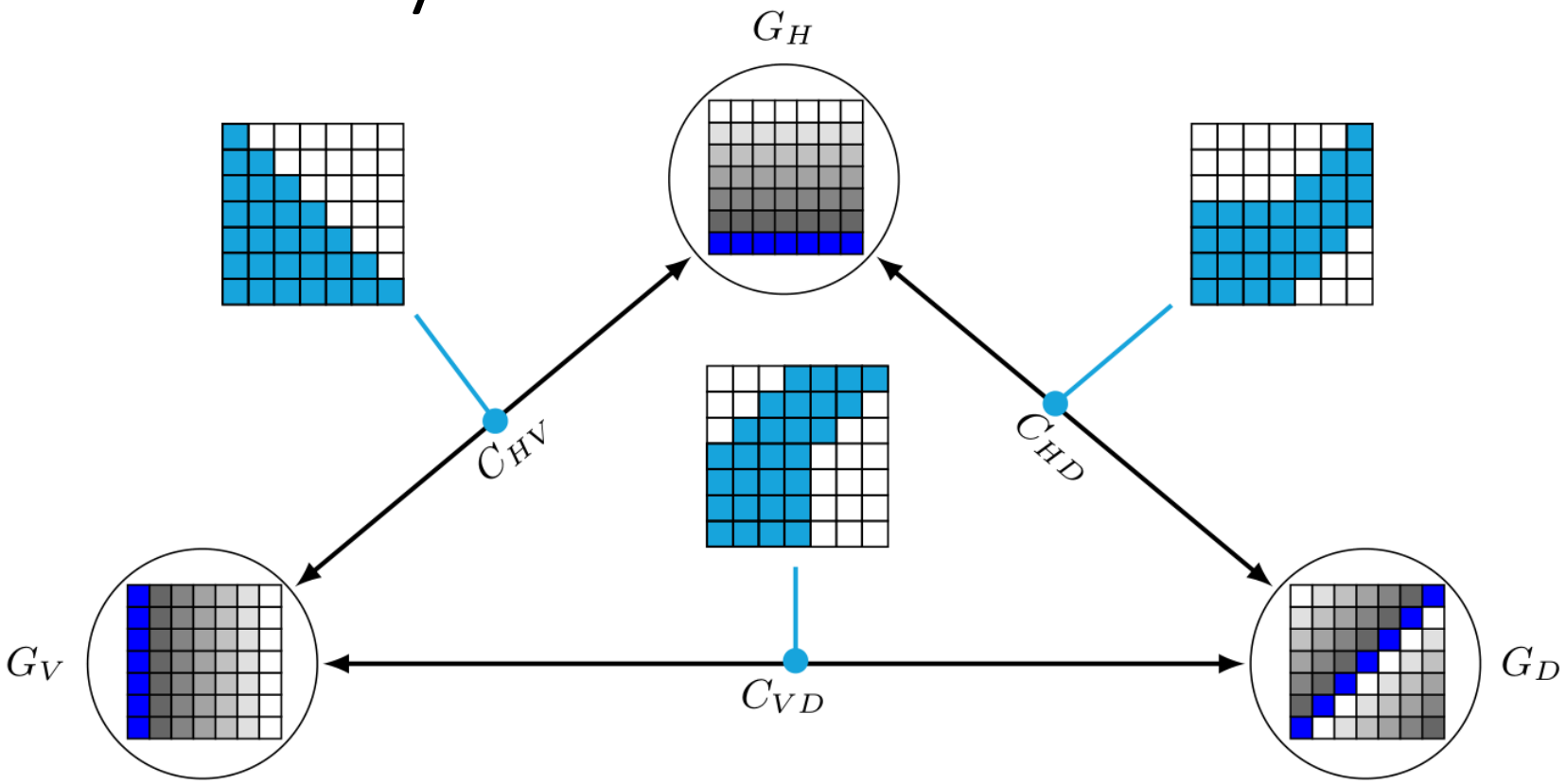


For a grid of prime size *n* x *n*, there can be at most *n*+1 different stable configurations with barrier *n* to pass between any of them

Directed Catalysis

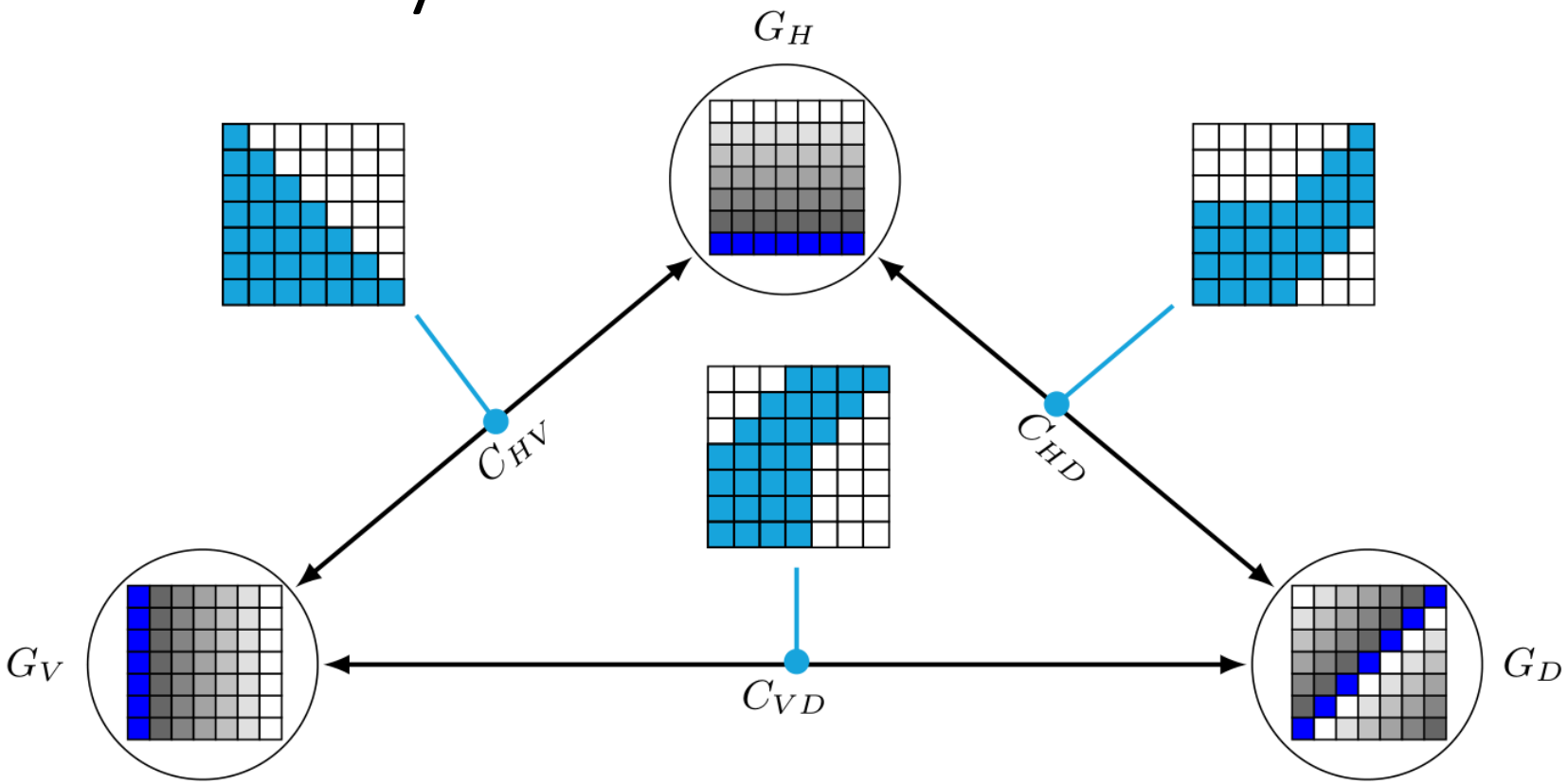


Directed Catalysis



Along a catalyzed pathway, the barrier is 1

Directed Catalysis



Along a catalyzed pathway, the barrier is 1

Otherwise the barrier is n/2

Social Golfer Problem

- Can 25 (n^2) golfers play in 5-somes (n-somes) for 6 (n+1) days, so that no two golfers play together more than once?

Social Golfer Problem

- Can 25 (n^2) golfers play in 5-somes (n-somes) for 6 (n+1) days, so that no two golfers play together more than once?
- First studied by Euler.

Social Golfer Problem

- Can 25 (n^2) golfers play in 5-somes (n-somes) for 6 (n+1) days, so that no two golfers play together more than once?
- First studied by Euler.
- True if n is a prime power (2,3,4,5,7,8,9,11,13,…)

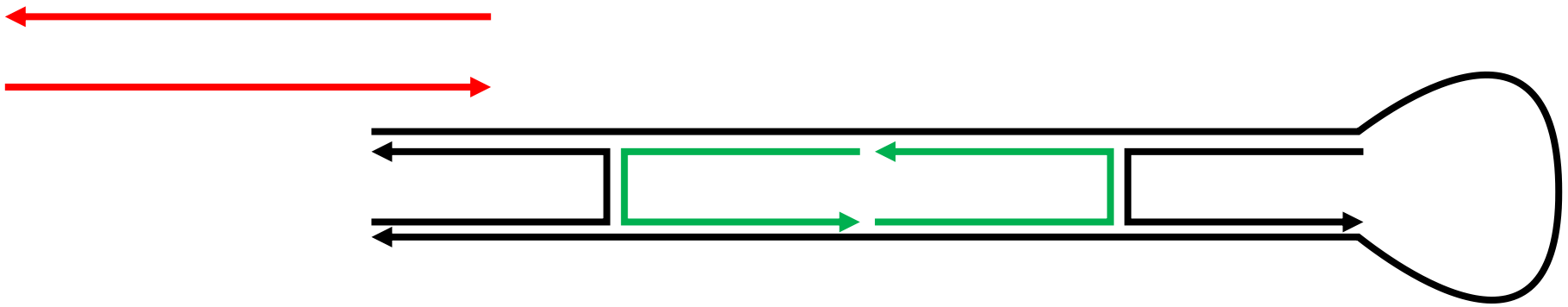
Social Golfer Problem

- Can 25 (n^2) golfers play in 5-somes (n-somes) for 6 (n+1) days, so that no two golfers play together more than once?
- First studied by Euler.
- True if n is a prime power (2,3,4,5,7,8,9,11,13,…)
- False for smallest non-prime power n=6: can only play for 3 days!
  [Gaston Tarry (1901). "Le Probléme des 36 Officiers". *Compte Rendu de l'Association Française pour l'Avancement des Sciences*. Secrétariat de l'Association. 2: 170–203.]

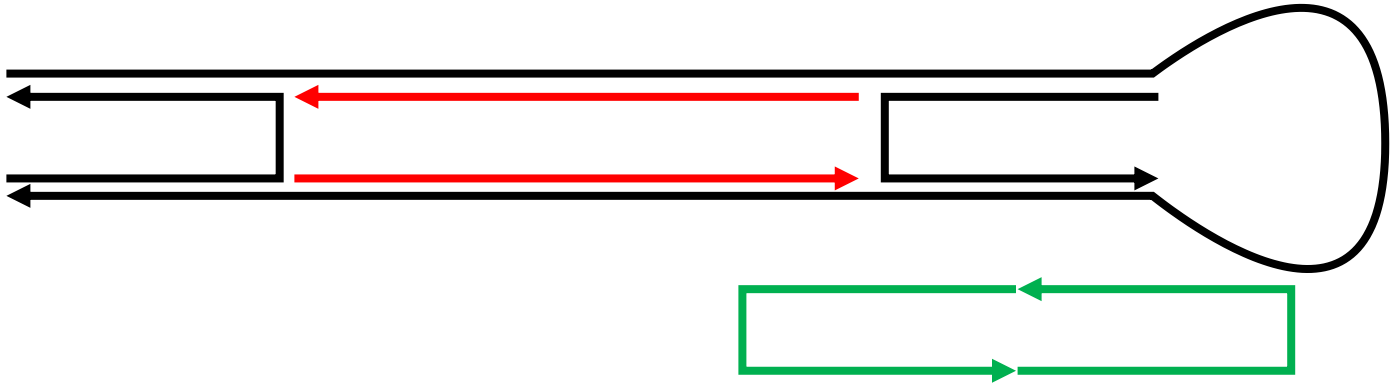
Social Golfer Problem

- Can 25 (n^2) golfers play in 5-somes (n-somes) for 6 (n+1) days, so that no two golfers play together more than once?
- First studied by Euler.
- True if n is a prime power (2,3,4,5,7,8,9,11,13,…)
- False for smallest non-prime power n=6: can only play for 3 days!
  [Gaston Tarry (1901). "Le Probléme des 36 Officiers". *Compte Rendu de l'Association Française pour l'Avancement des Sciences*. Secrétariat de l'Association. 2: 170–203.]
- Unknown for next prime power n=10:
  - trivial upper bound is 11 days
  - best known lower bound is 3

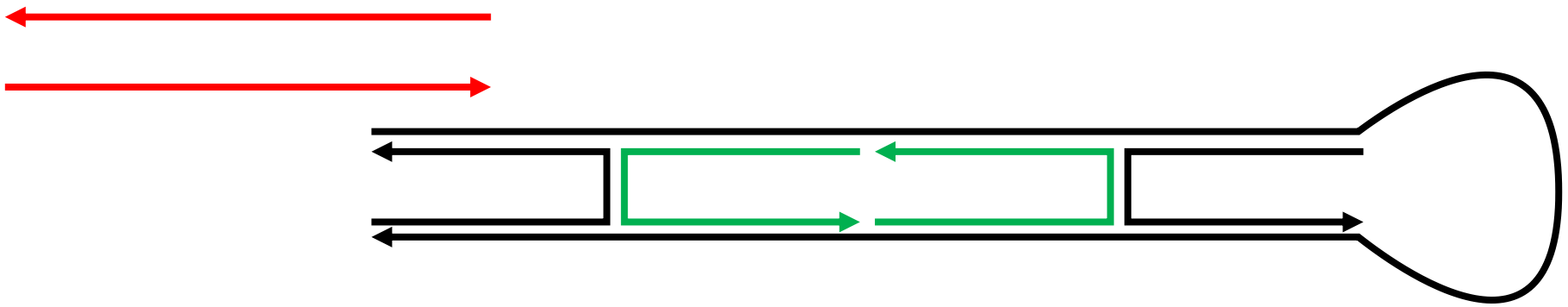
(Feasible?) DNA implementation



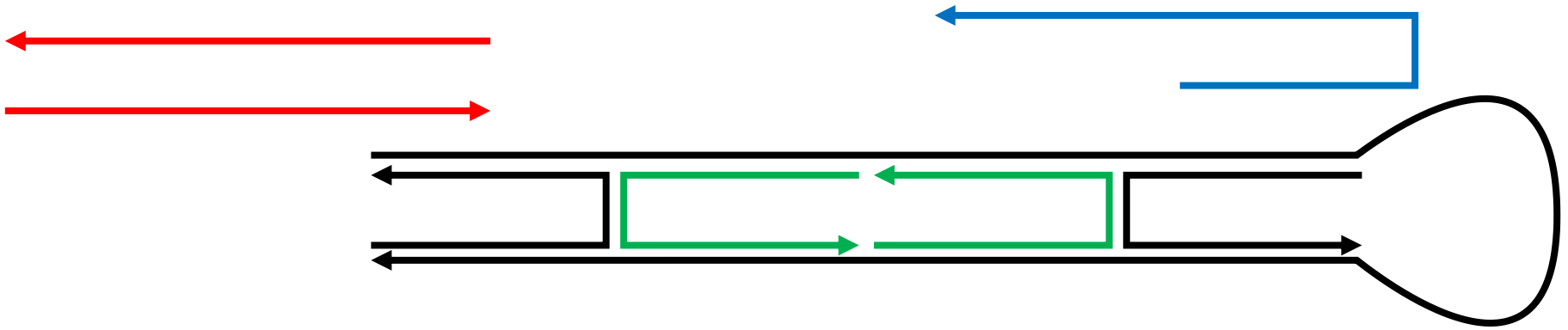
(Feasible?) DNA implementation



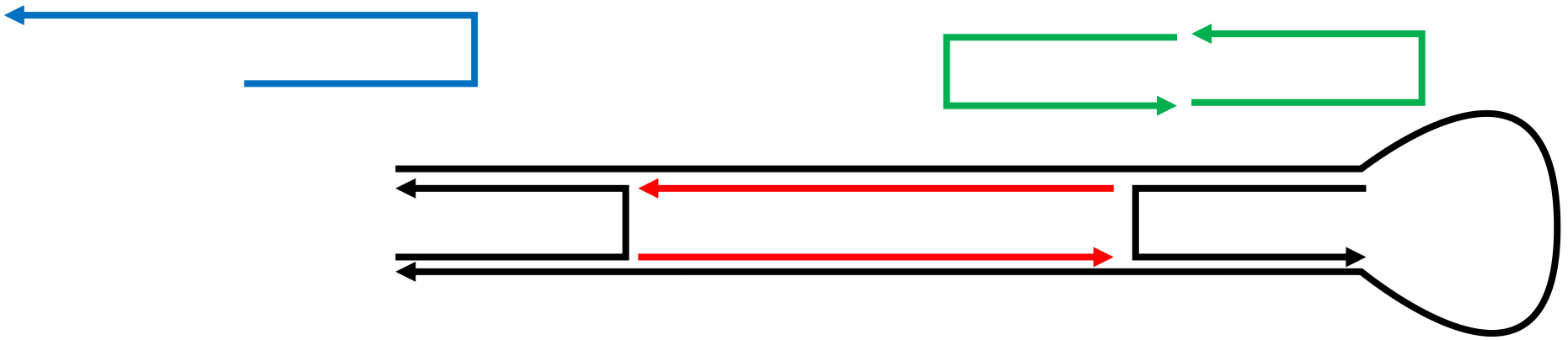
(Feasible?) DNA implementation



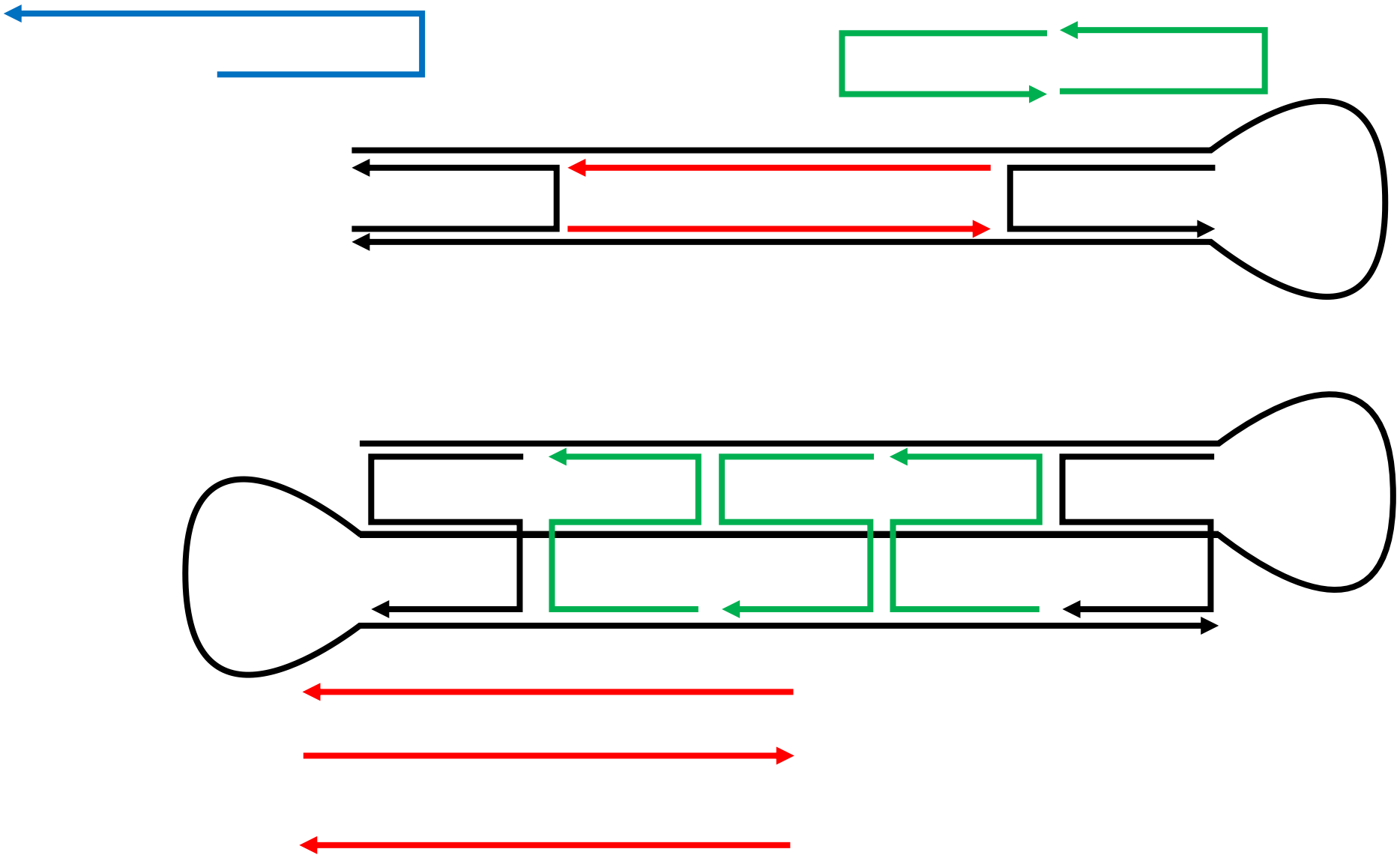
(Feasible?) DNA implementation



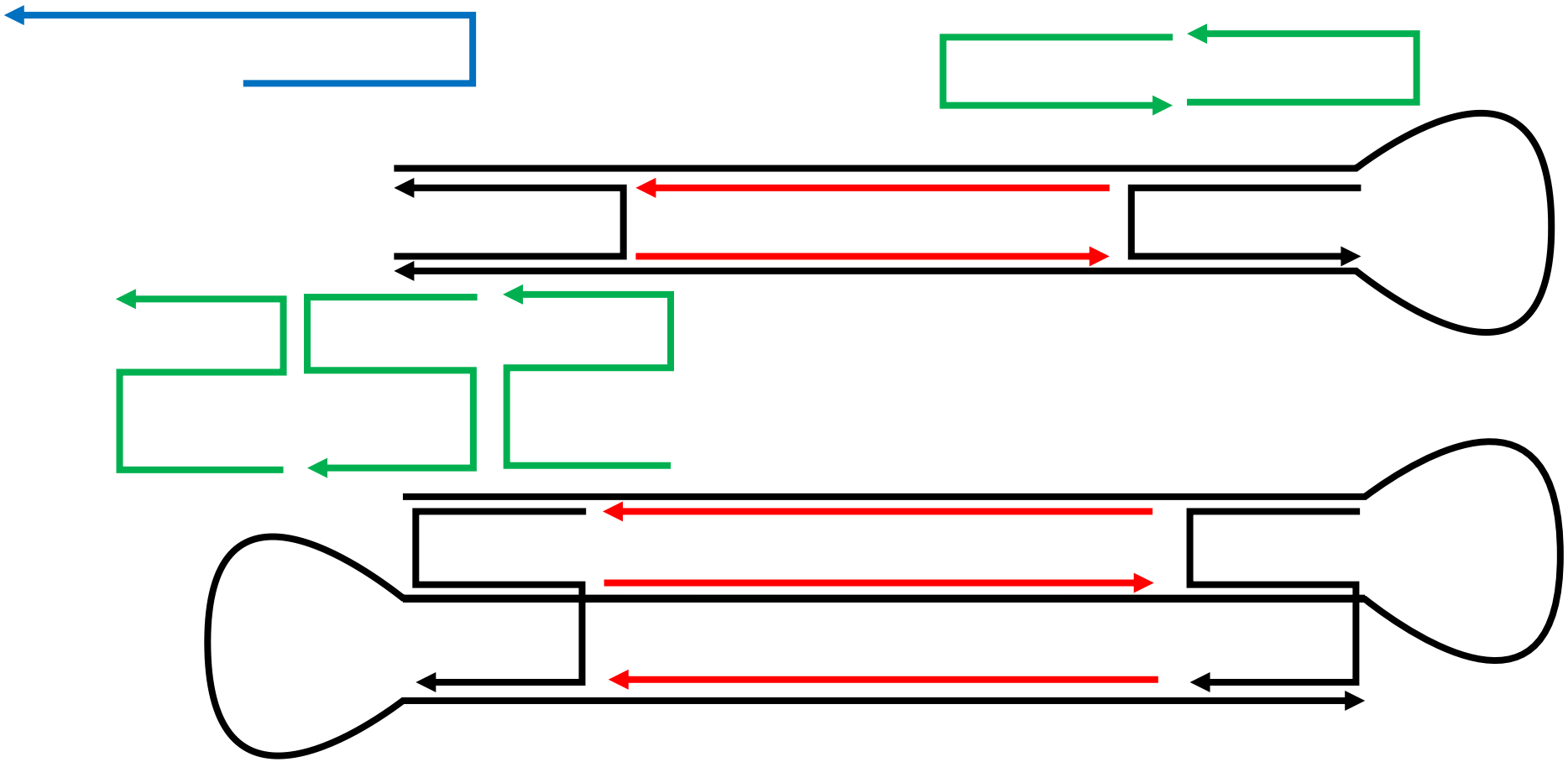
(Feasible?) DNA implementation



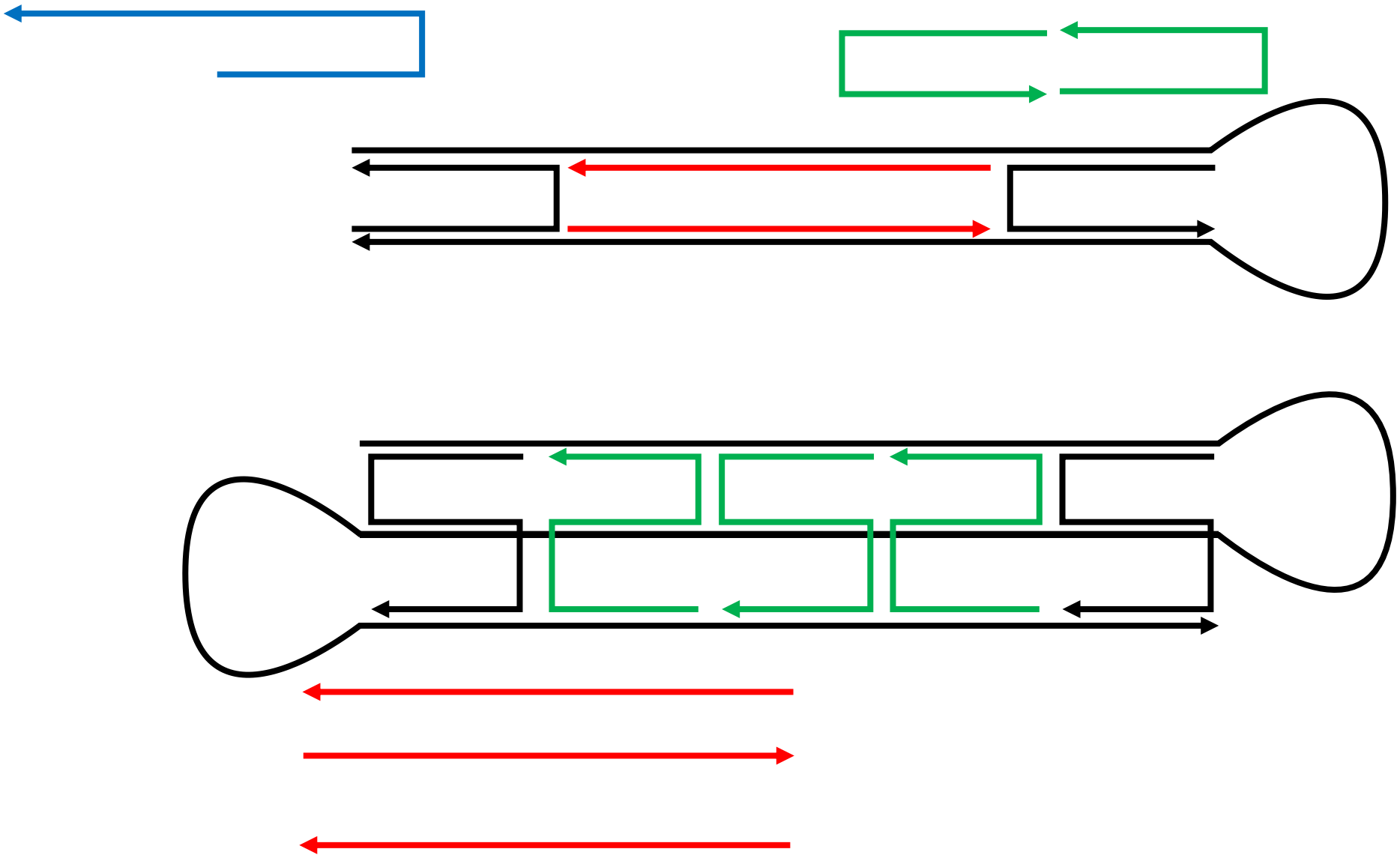
(Feasible?) DNA implementation



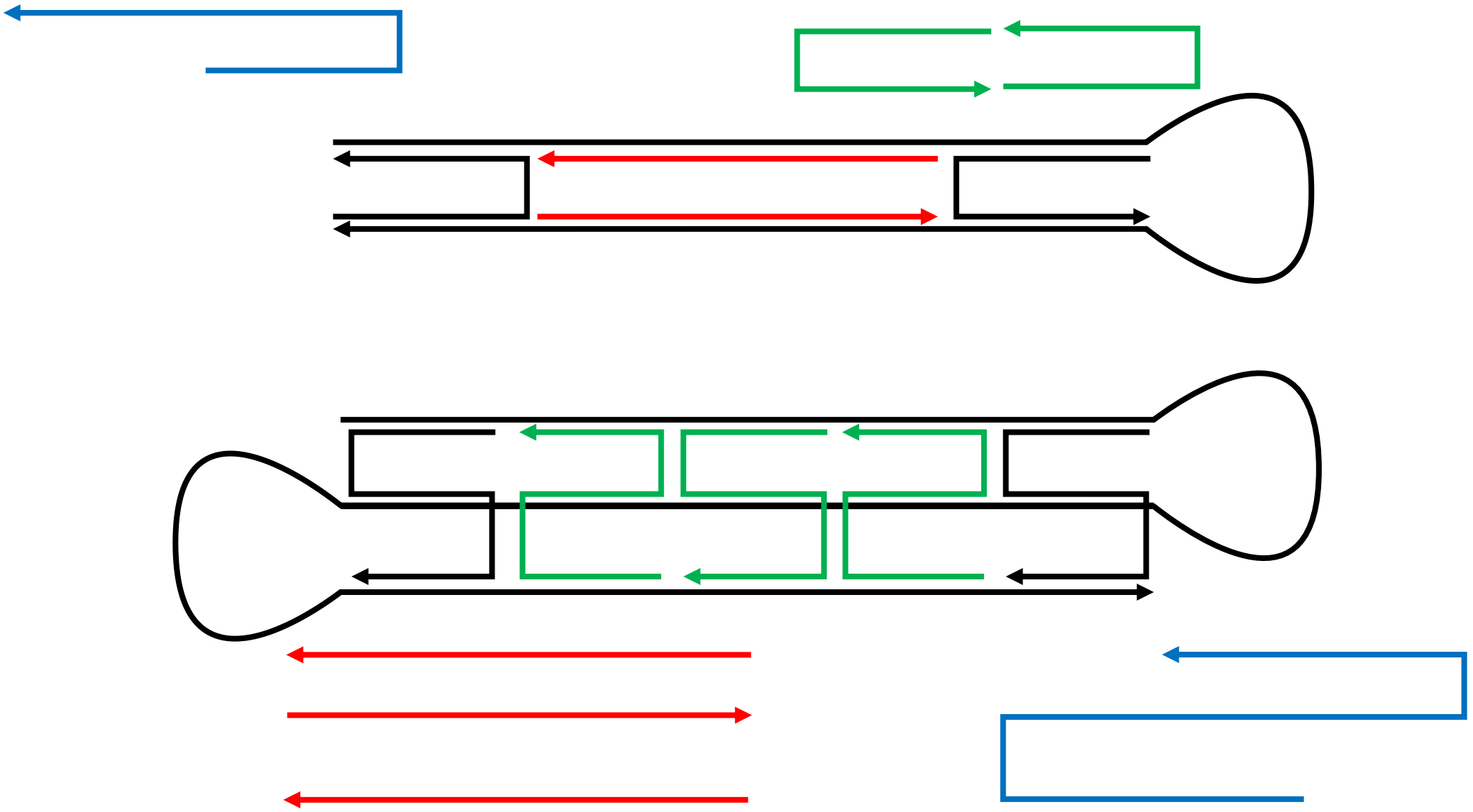
(Feasible?) DNA implementation



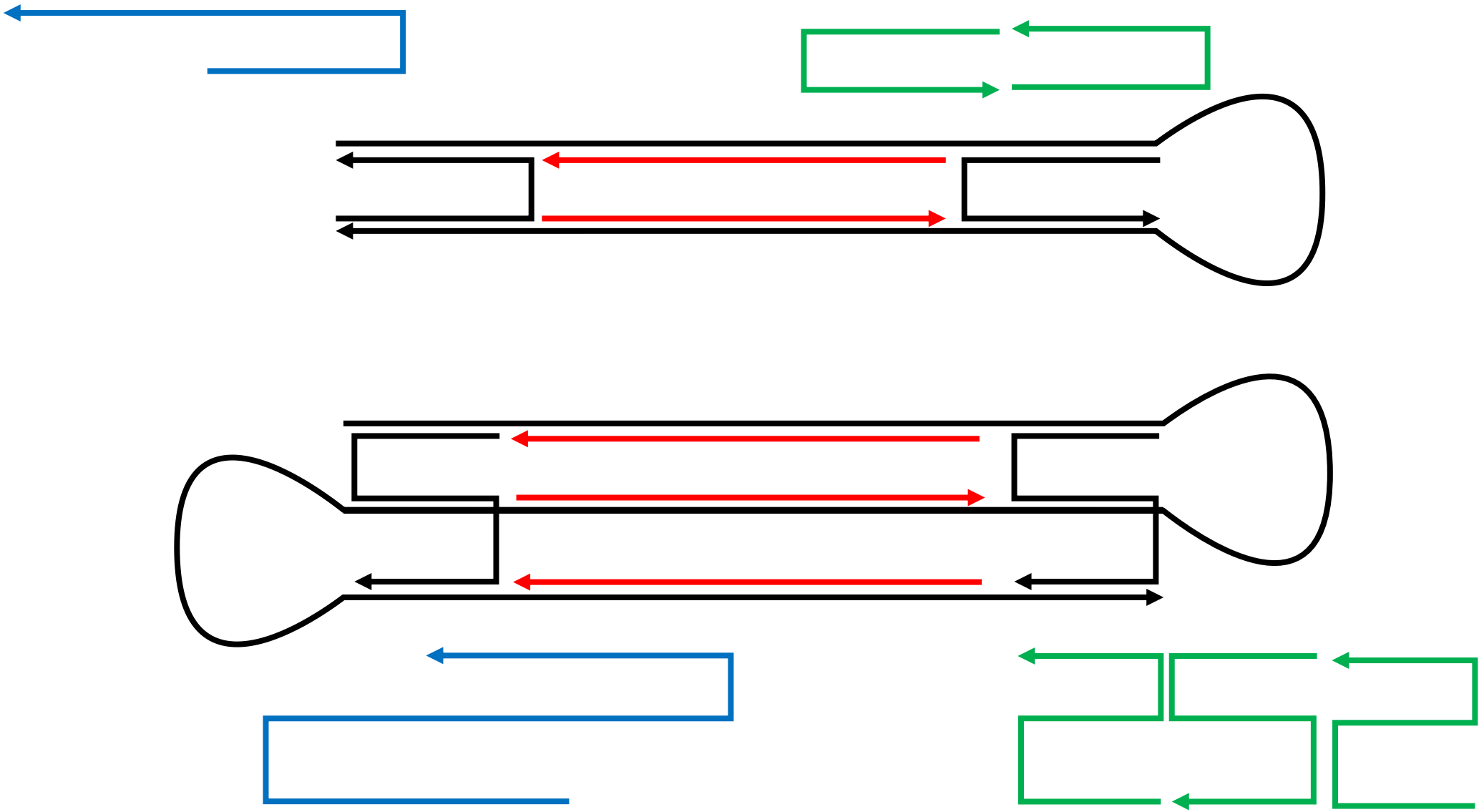
(Feasible?) DNA implementation



(Feasible?) DNA implementation



(Feasible?) DNA implementation



Thermodynamic self-assembly

Grafting the TBN model onto self-assembly

A modest goal

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

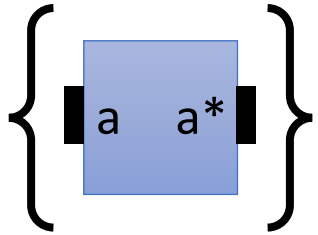
A modest goal

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

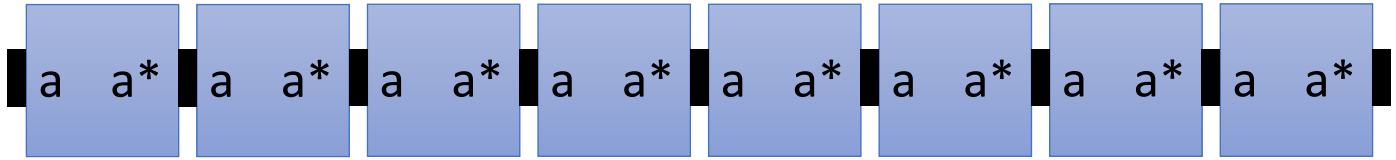
A modest goal

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

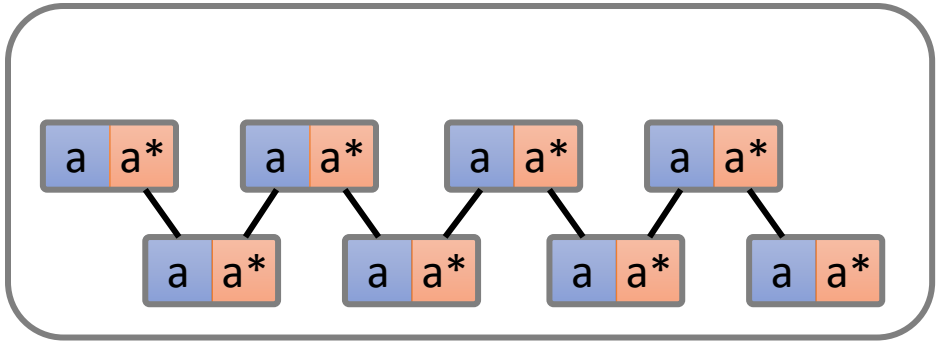
set of monomer types:



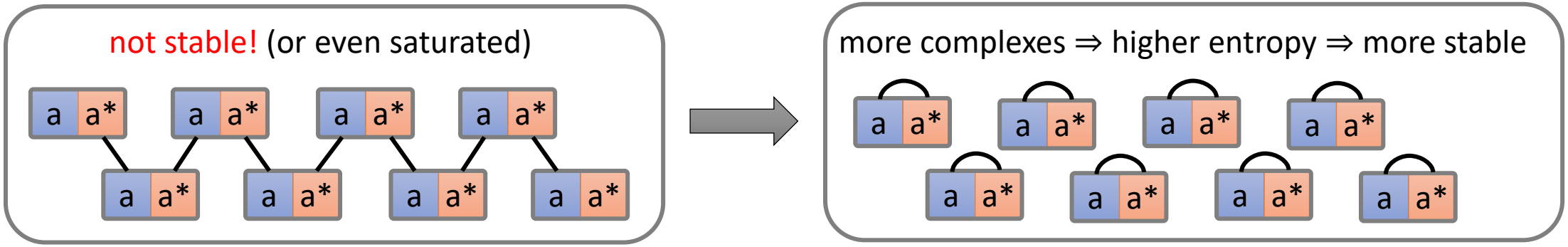
size-8 complex (assembly) formed with 8 copies of monomer



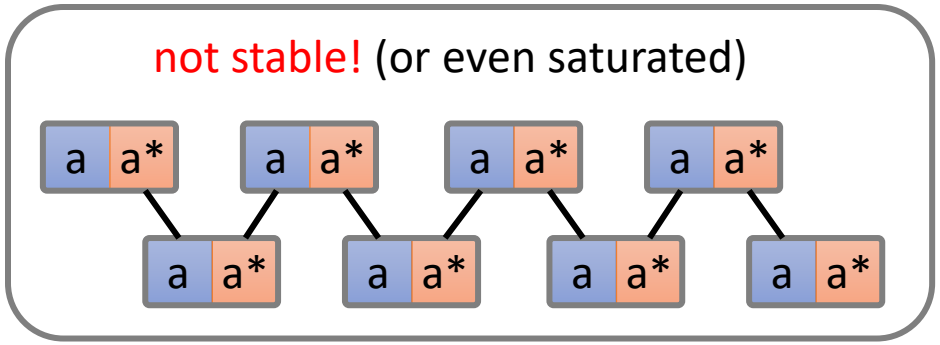
Difficulty of self-assembling large complexes

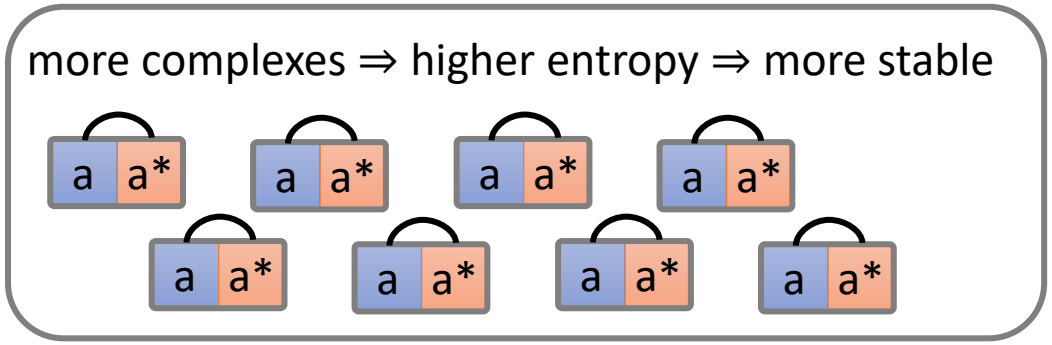


Difficulty of self-assembling large complexes

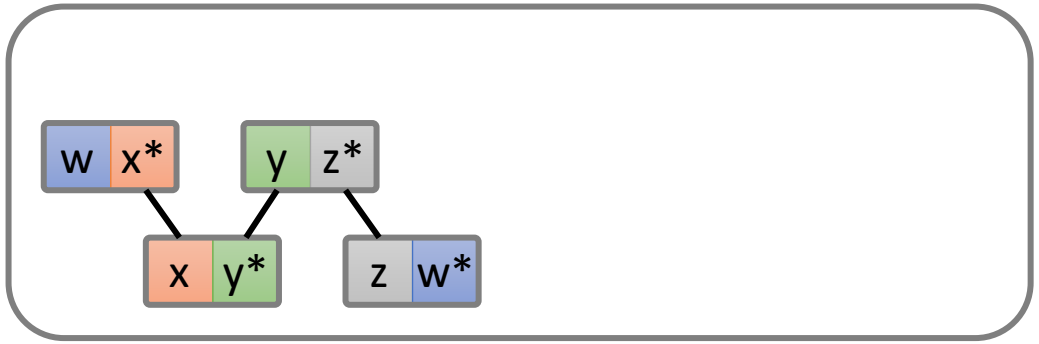


Difficulty of self-assembling large complexes

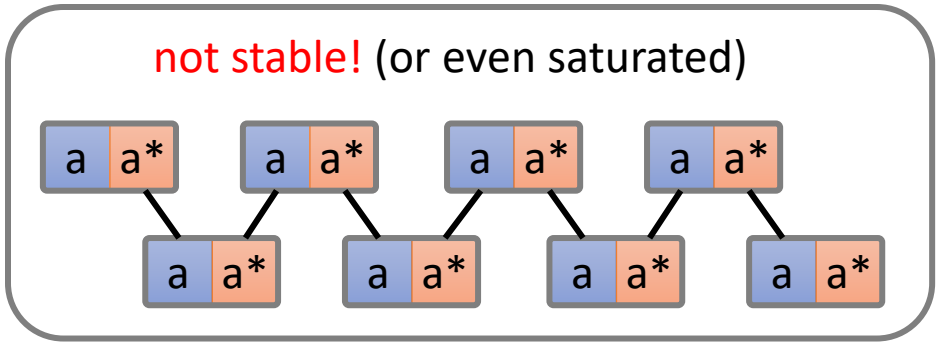


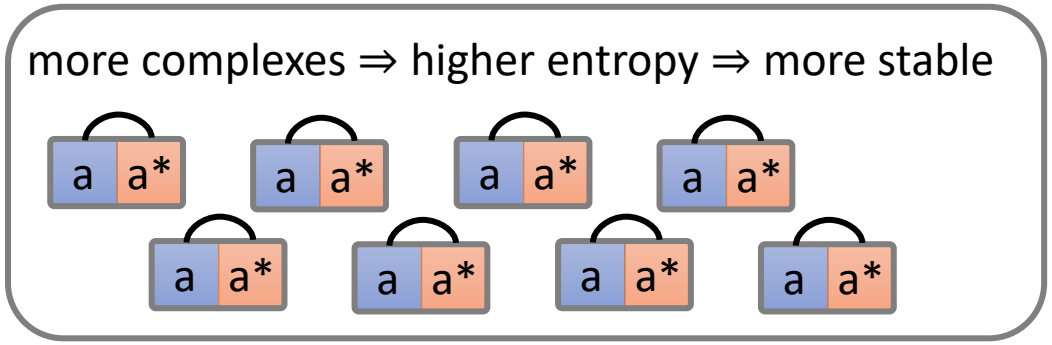


attempt 2:

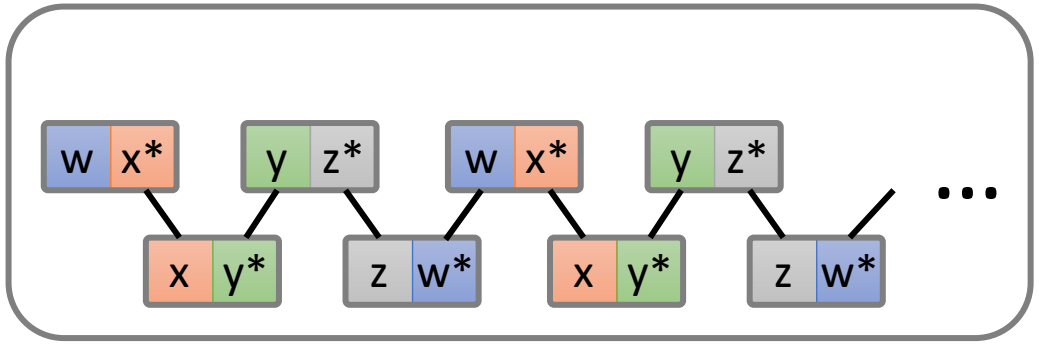


Difficulty of self-assembling large complexes

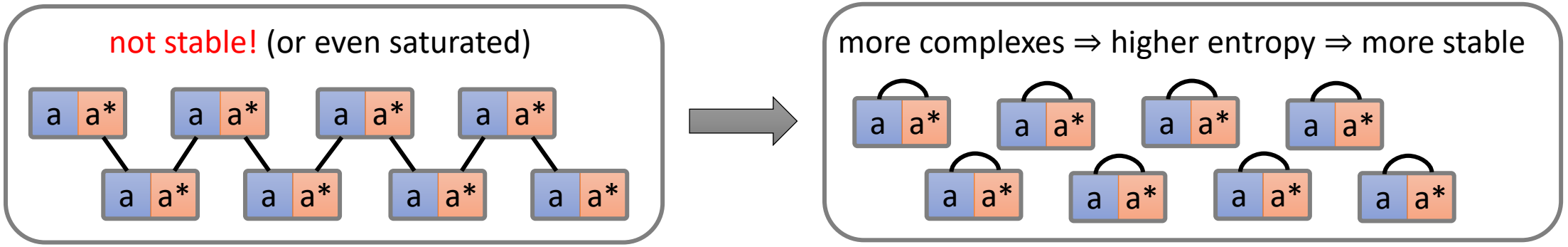




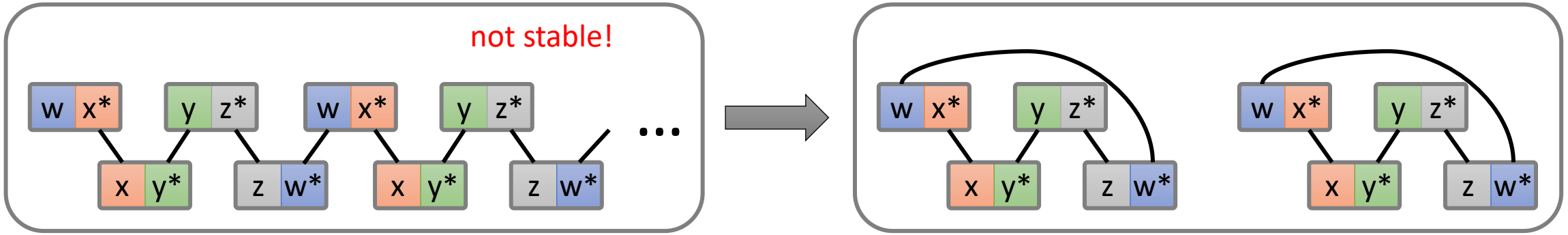
attempt 2:



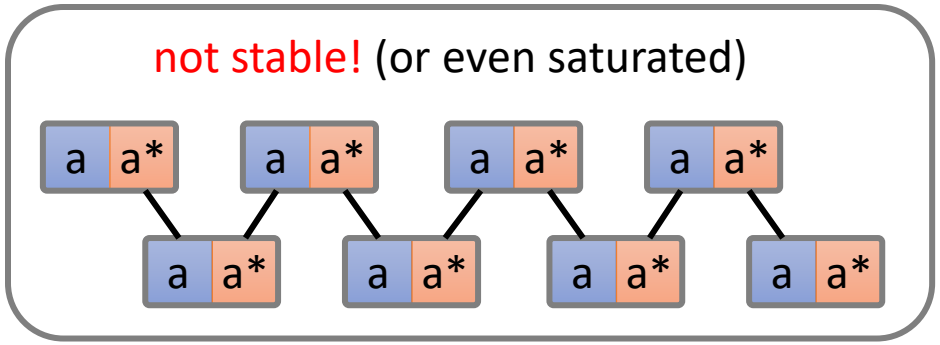
Difficulty of self-assembling large complexes

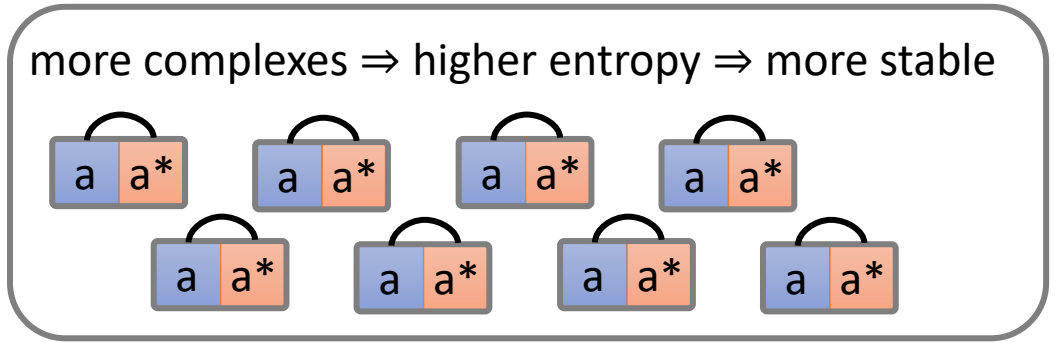


attempt 2:



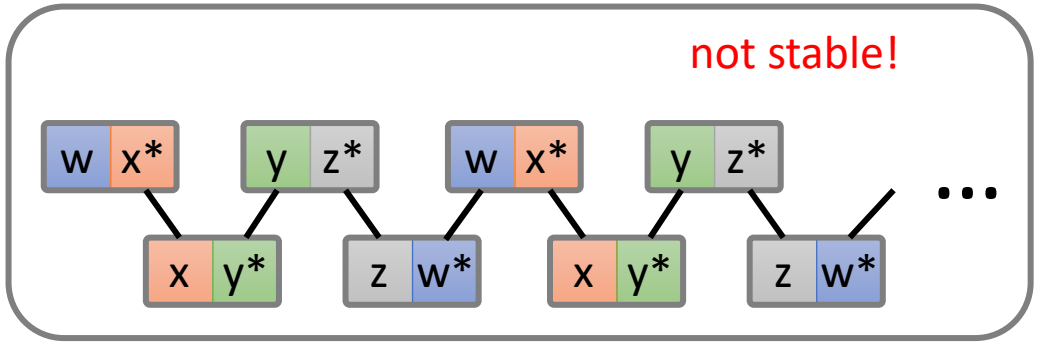
Difficulty of self-assembling large complexes

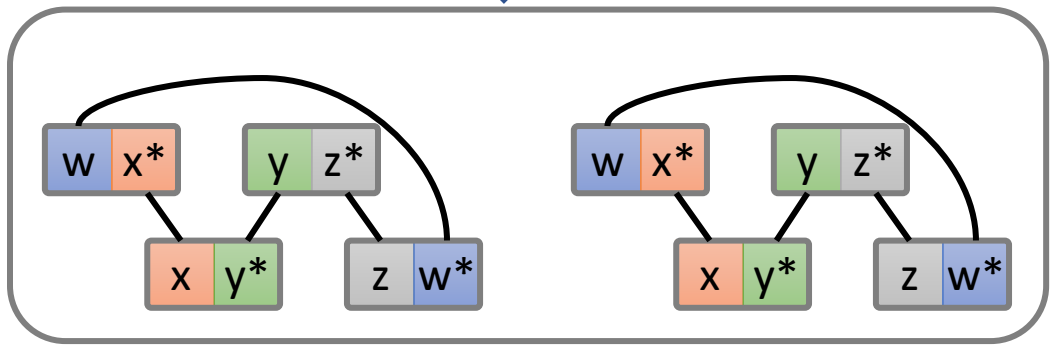




These have more complexes, and each is self-saturating
(every domain can be bound *within* the complex)

attempt 2:





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 complex 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 complex 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 complex 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 complex 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 complex of size at least S.

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

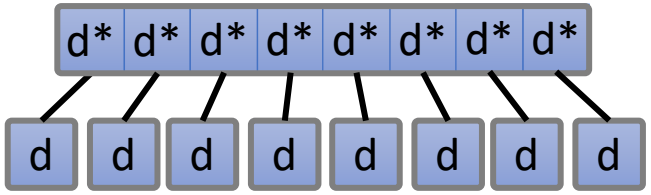
An even more modest goal

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

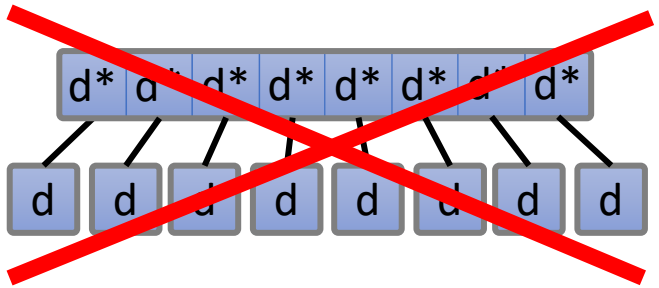
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable complex of size at least S.~~

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 complex 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

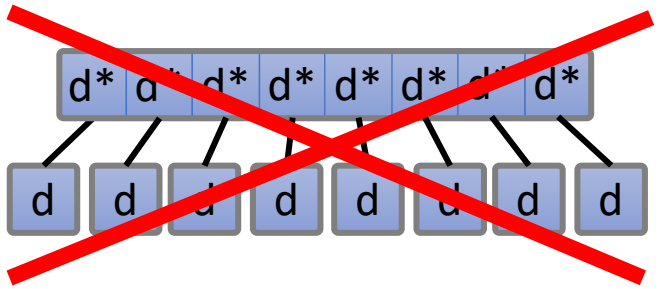
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable complex of size at least S.~~

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 complex 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

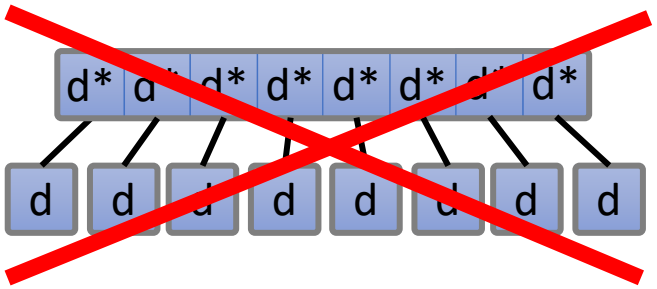
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable complex of size at least S.~~

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 complex 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 complex of size at least S.~~

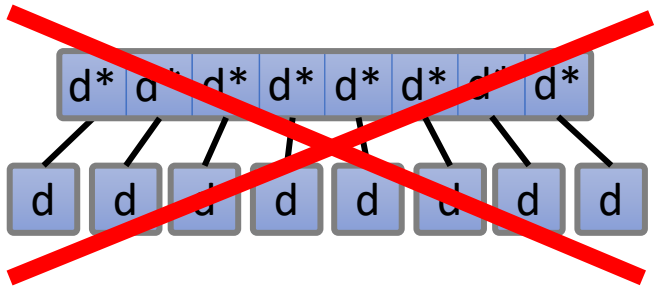
and $O(1)$ domains per monomer

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

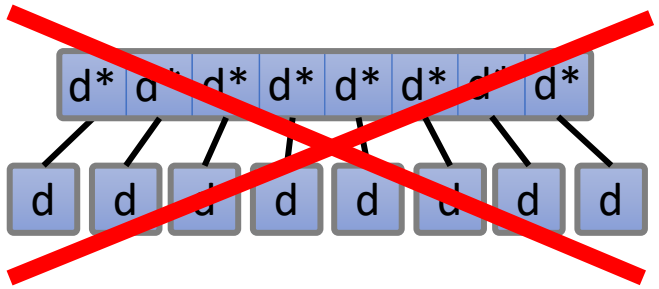
and $O(1)$ domains per monomer

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

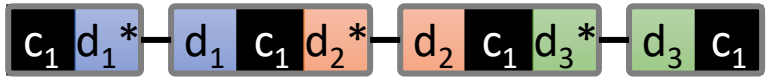
How large can we make S relative to D and M?

$S \approx D^2$

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



$S \approx D$



An even more modest goal

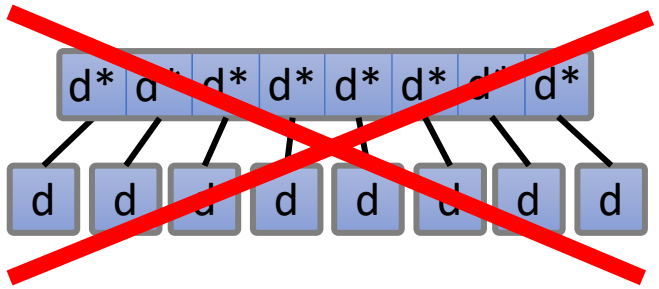
~~**Original goal**: Design a set of monomer types so that, for all $S \in \mathbb{N}$, there is a stable complex of size at least S.~~

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 complex of size at least S.

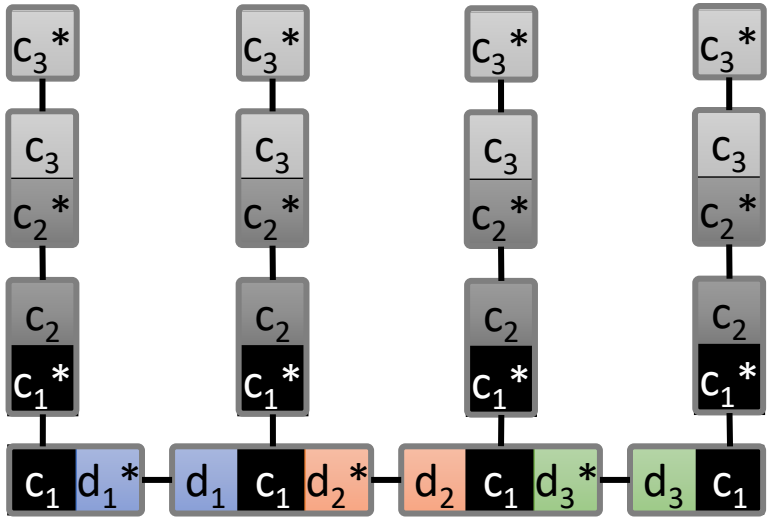
How large can we make S relative to D and M?

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



$S \approx D$

$S \approx D^2$

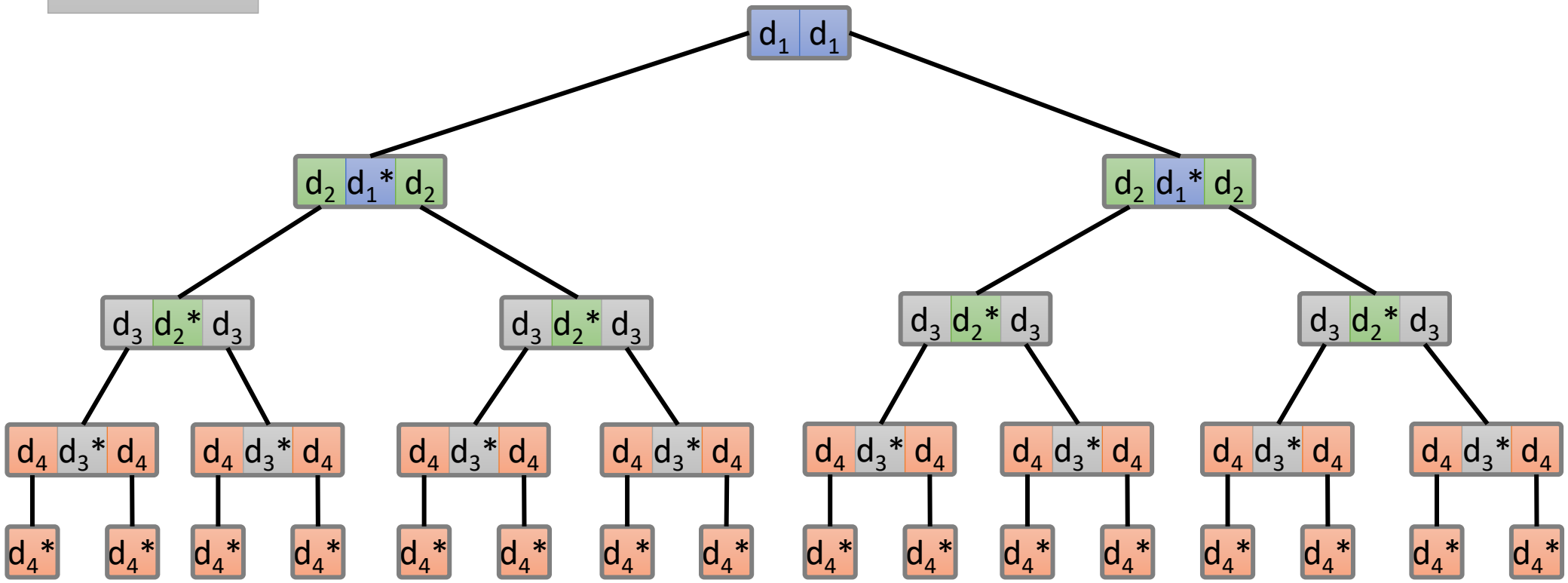


How large can we make *S* relative to *D* and *M*?

$S \approx 2^D$?

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

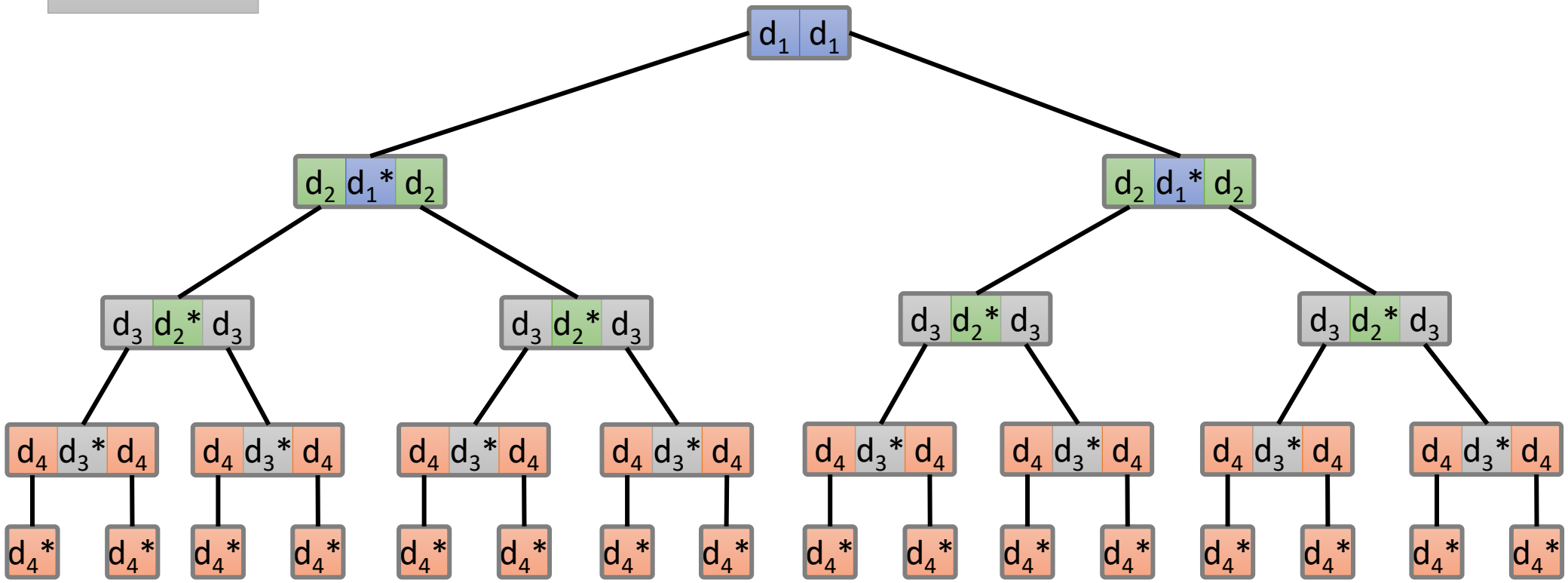
$S \approx 2^D$?



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

$S \approx 2^D$?

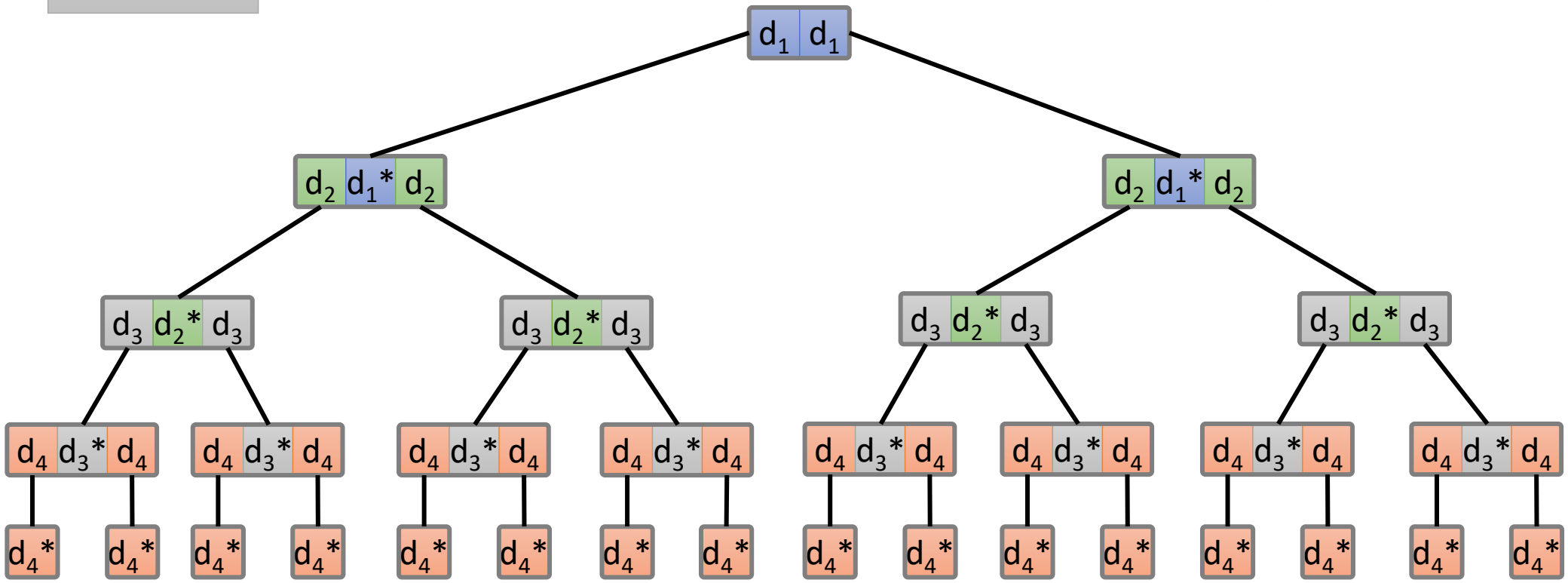
$S \approx 2^{2^D}$??



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

$S \approx 2^D$?

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



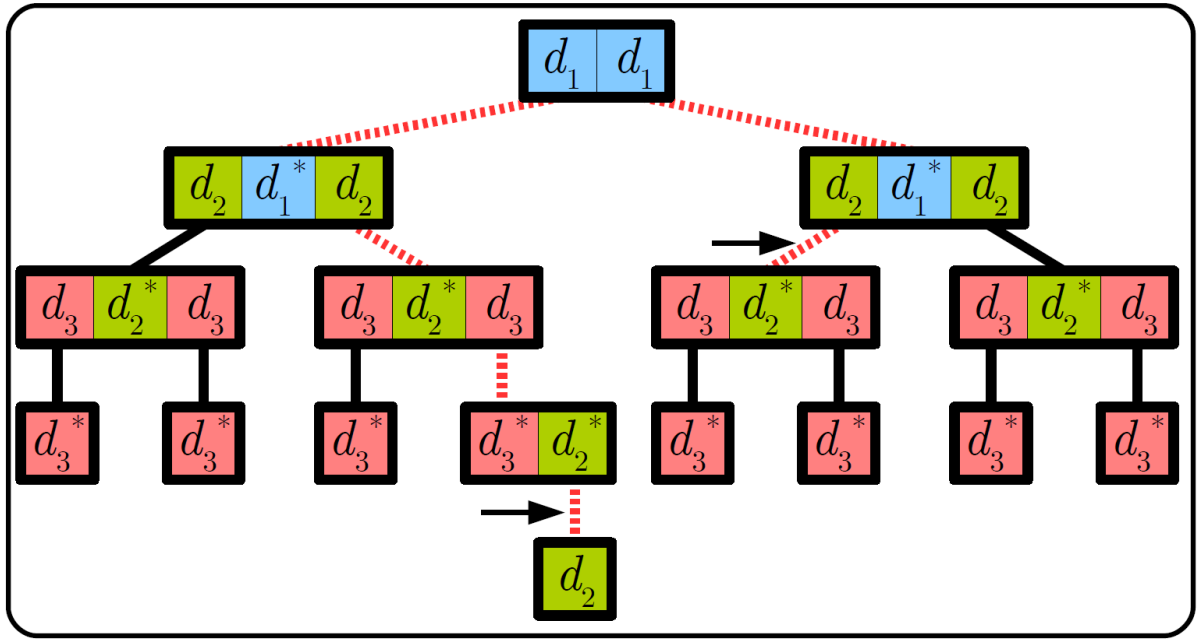
Stable complexes have at most exponential size

Theorem: Any thermodynamic binding network with

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

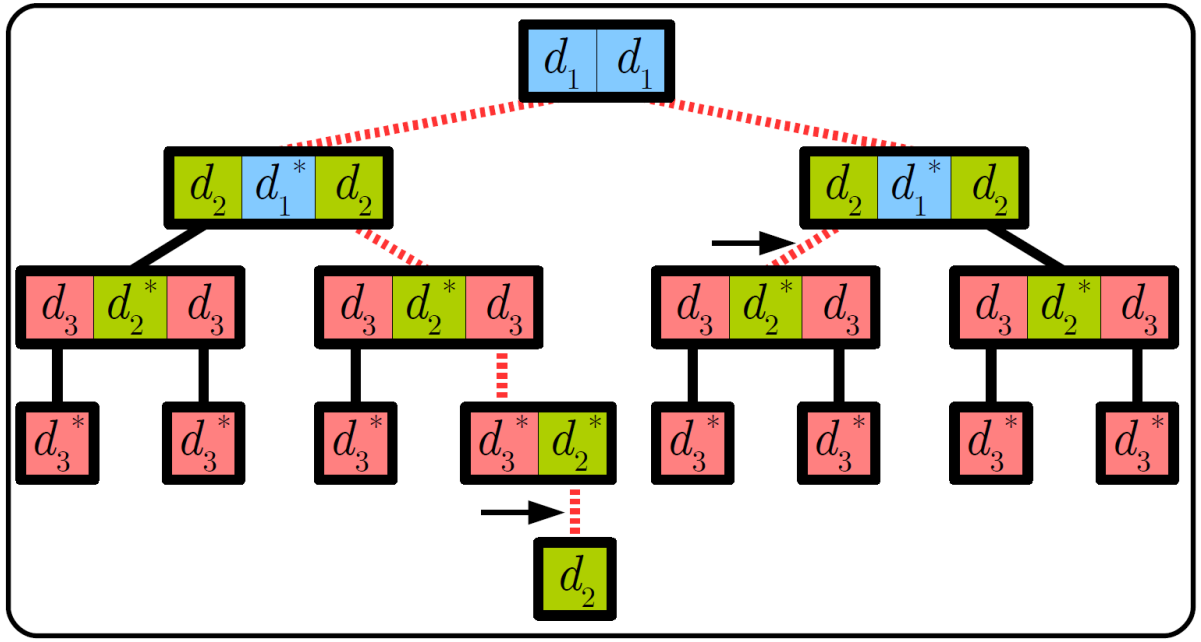
Has stable complexes of size $\leq 2(M+D)(AD)^{2D+3} = \text{poly}(D^D)$ if $A = O(1)$

Easy proof if binding graph is acyclic (tree)



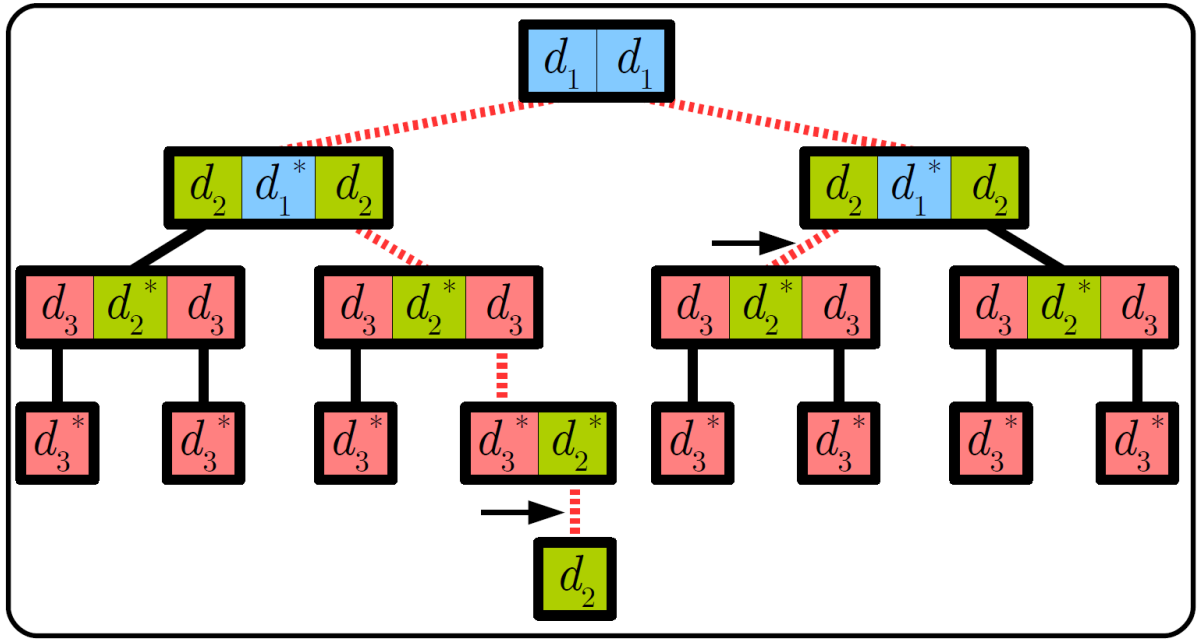
Easy proof if binding graph is acyclic (tree)

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



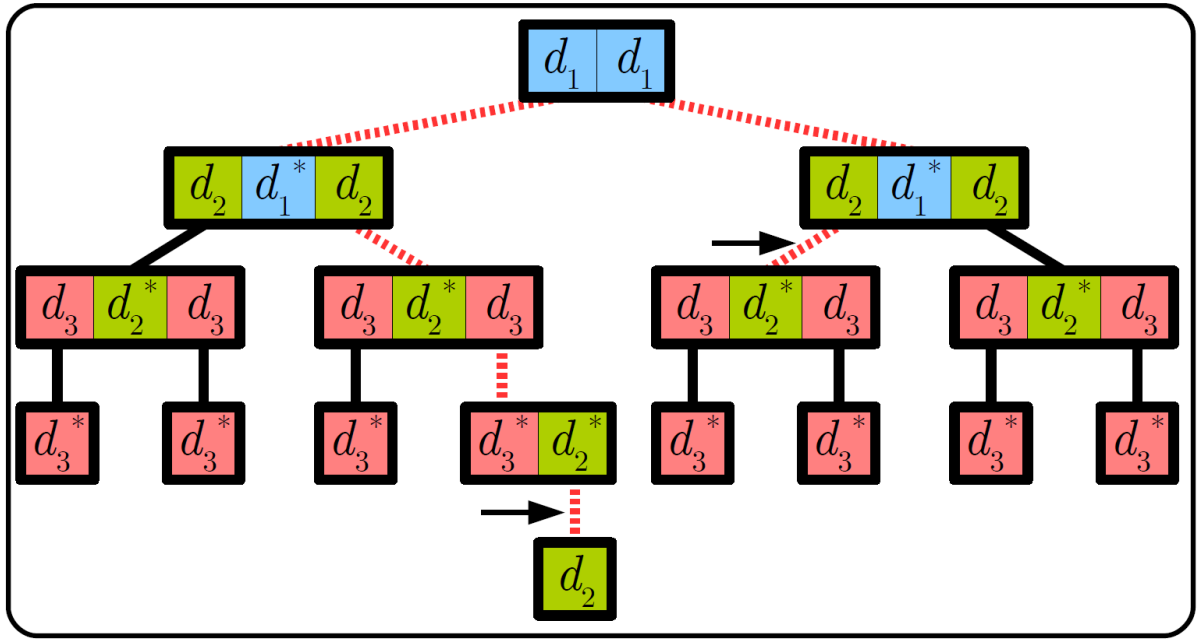
Easy proof if binding graph is acyclic (tree)

- 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)



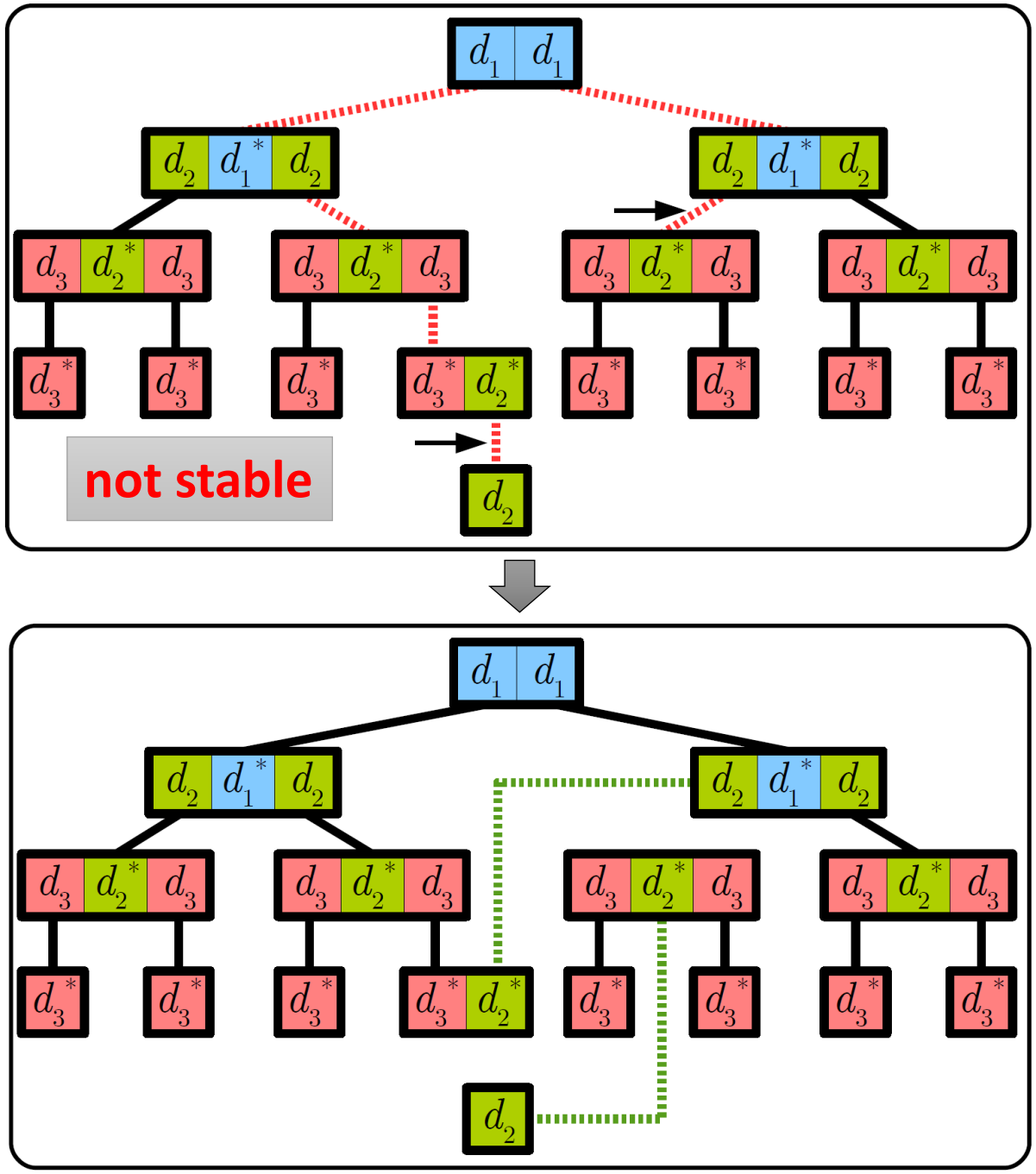
Easy proof if binding graph is acyclic (tree)

- 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 complexes as shown.



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) complex 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

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.
   2. Let $S_i = m_{1i} + m_{2i} + \ldots + m_{ki}$. Note that there is a m x d matrix M such that $S_i = M \cdot P_i$,

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.
   2. Let $S_i = m_{1i} + m_{2i} + \ldots + m_{ki}$. Note that there is a m x d matrix M such that $S_i = M \cdot P_i$,
3. Take several infinite subsequences:

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.
   2. Let $S_i = m_{1i} + m_{2i} + \ldots + m_{ki}$. Note that there is a m x d matrix M such that $S_i = M \cdot P_i$,
3. Take several infinite subsequences:
   1. Since there are a finite number of domain types, some infinite subsequence of P_i's agrees on which set of domain types are unbound.

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.
   2. Let $S_i = m_{1i} + m_{2i} + \ldots + m_{ki}$. Note that there is a m x d matrix M such that $S_i = M \cdot P_i$,
3. Take several infinite subsequences:
   1. Since there are a finite number of domain types, some infinite subsequence of P_i's agrees on which set of domain types are unbound.
   2. By Dickson's Lemma we may assume $P_1 < P_2 < \ldots$ and $S_1 < S_2 < \ldots$ i.e., each has all the monomers of the previous, plus some more, and each has all the unbound domains of the previous, plus some more.

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.
   2. Let $S_i = m_{1i} + m_{2i} + \ldots + m_{ki}$. Note that there is a m x d matrix M such that $S_i = M \cdot P_i$,
3. Take several infinite subsequences:
   1. Since there are a finite number of domain types, some infinite subsequence of P_i's agrees on which set of domain types are unbound.
   2. By Dickson's Lemma we may assume $P_1 < P_2 < \ldots$ and $S_1 < S_2 < \ldots$ i.e., each has all the monomers of the previous, plus some more, and each has all the unbound domains of the previous, plus some more.
4. Let $d = P_2 - P_1$. Then $M \cdot d = M \cdot P_2 - M \cdot P_1 = S_2 - S_1 \geq 0$.

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.
   2. Let $S_i = m_{1i} + m_{2i} + \ldots + m_{ki}$. Note that there is a m x d matrix M such that $S_i = M \cdot P_i$,
3. Take several infinite subsequences:
   1. Since there are a finite number of domain types, some infinite subsequence of P_i's agrees on which set of domain types are unbound.
   2. By Dickson's Lemma we may assume $P_1 < P_2 < \ldots$ and $S_1 < S_2 < \ldots$ i.e., each has all the monomers of the previous, plus some more, and each has all the unbound domains of the previous, plus some more.
4. Let $d = P_2 - P_1$. Then $M \cdot d = M \cdot P_2 - M \cdot P_1 = S_2 - S_1 \geq 0$.
5. i.e., $S_2 = S_1 + M \cdot d$ and all three are nonnegative,

Somewhat easy proof that unbounded size complexes cannot be assembled

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

Theorem: Original goal is impossible.

Proof:

1. Suppose otherwise, let P_1, P_2, … in $\mathbb{N}^m$ be an infinite sequence of stable complexes increasing in size.
   m is number of monomer types, $P_i(j)$ = # monomers of type j in complex P_i.
2. Represent each monomer type as a vector in $\mathbb{Z}^d$ as on previous slide.
   1. P_i is composed of monomers m_{1i}, m_{2i}, …, m_{ki}.
   2. Let $S_i = m_{1i} + m_{2i} + \ldots + m_{ki}$. Note that there is a m x d matrix M such that $S_i = M \cdot P_i$,
3. Take several infinite subsequences:
   1. Since there are a finite number of domain types, some infinite subsequence of P_i's agrees on which set of domain types are unbound.
   2. By Dickson's Lemma we may assume $P_1 < P_2 < \ldots$ and $S_1 < S_2 < \ldots$ i.e., each has all the monomers of the previous, plus some more, and each has all the unbound domains of the previous, plus some more.
4. Let $d = P_2 - P_1$. Then $M \cdot d = M \cdot P_2 - M \cdot P_1 = S_2 - S_1 \geq 0$.
5. i.e., $S_2 = S_1 + M \cdot d$ and all three are nonnegative,
6. i.e., we can split S_2 into 2 disjoint nonempty nonnegative subsets, S_1 and $M \cdot d$. **QED**

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**?

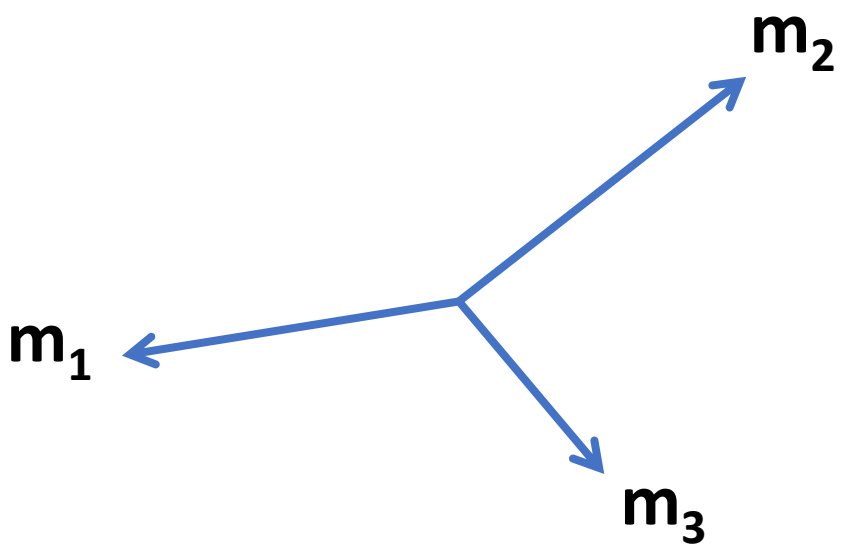
- 0/1-INTEGER-PROGRAMMING is **NP**-complete (Karp 1972).
- Non-obvious fact: INTEGER-PROGRAMMING is in **NP**. *(independently due to [Borosh and Treybig 1976], [Gathen and Sieveking 1978], [Kannan and Monma 1978])*

  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.

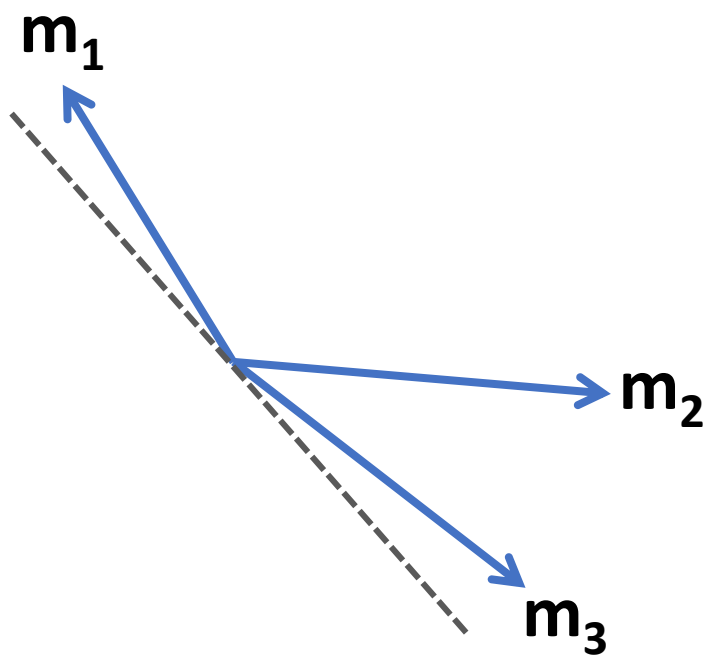
Farkas' Lemma

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



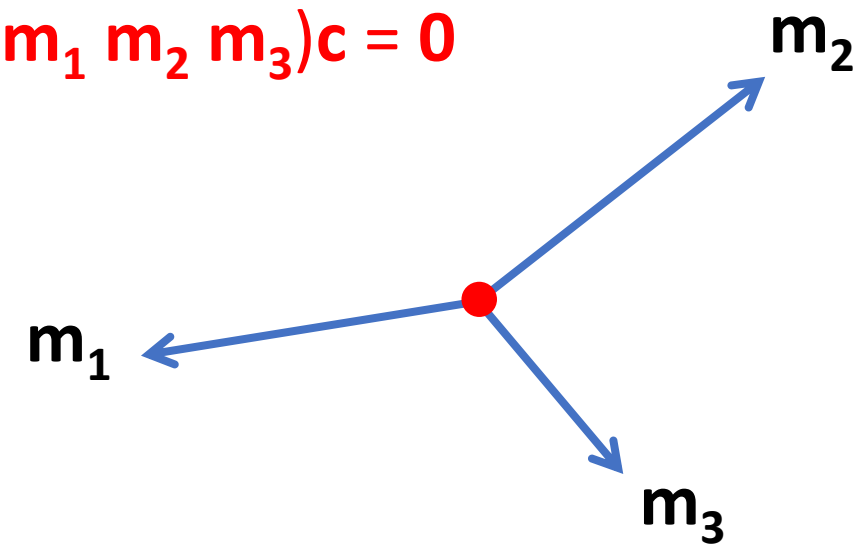
Farkas' Lemma

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

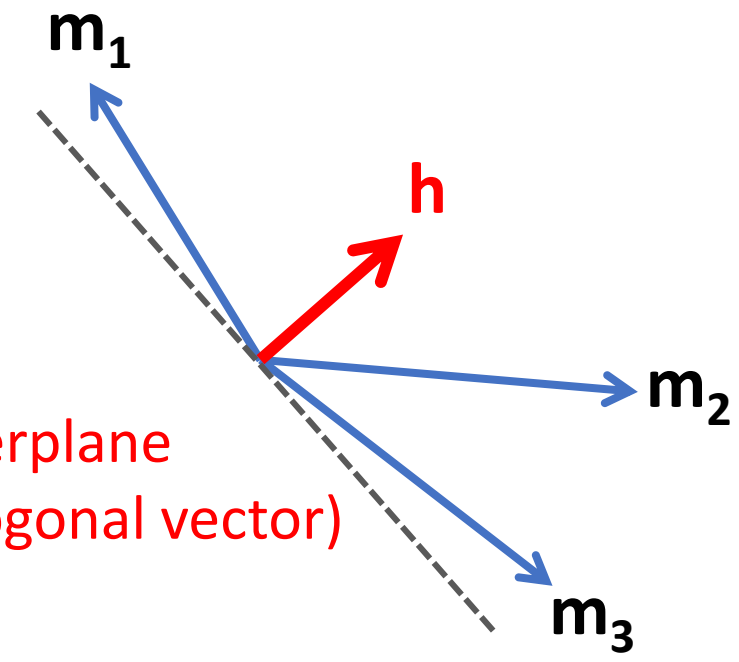
a) are directions of balanced forces

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

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



b) lie on one side of some hyperplane

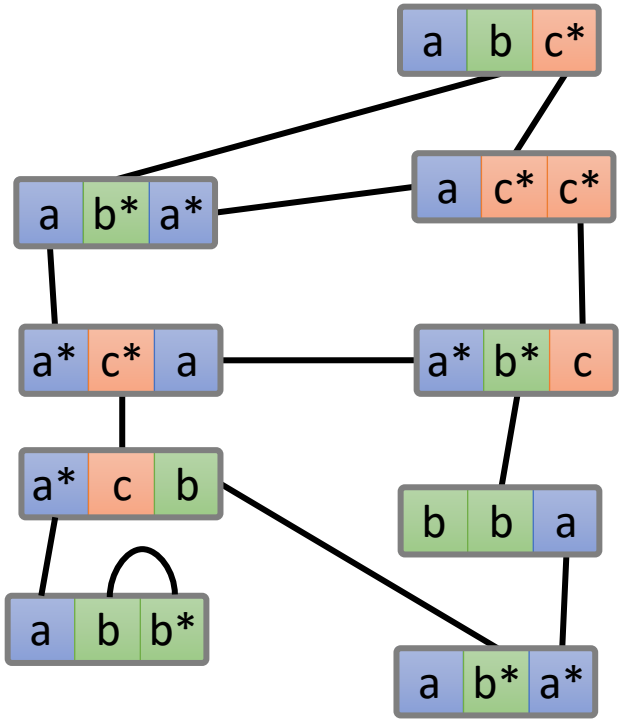


$\exists \mathbf{h}$ (hyperplane orthogonal vector)

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

How to prove exponential complex size bound for complexes with cycles in binding graph?

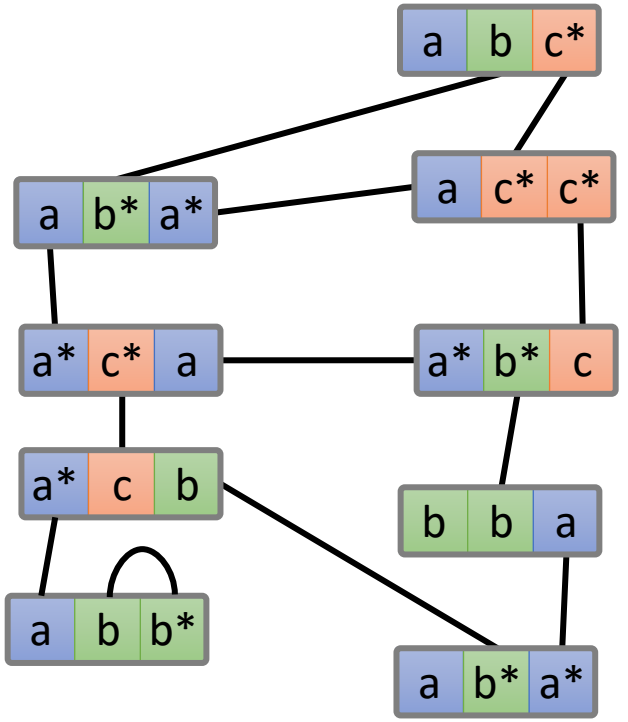
monomer collection $\mathbf{c} \in \mathbb{N}^M$



How to prove exponential complex size bound for complexes 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*

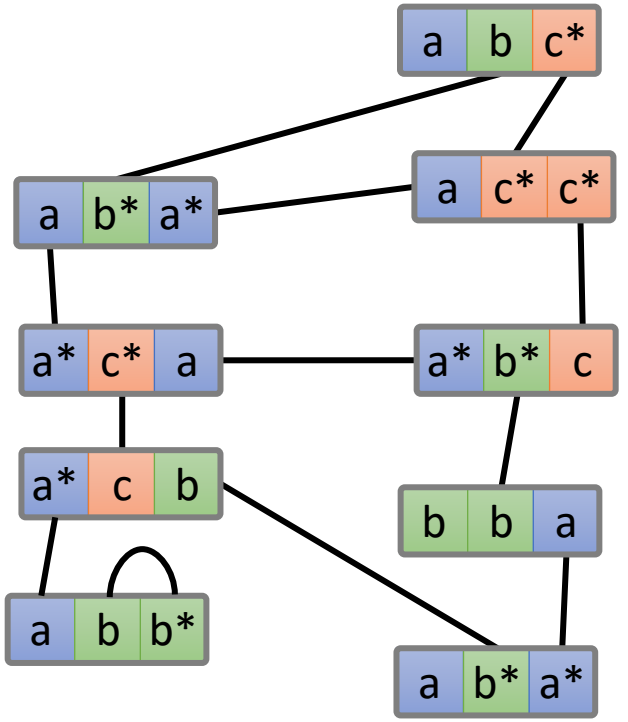
monomer collection $\mathbf{c} \in \mathbb{N}^M$



How to prove exponential complex size bound for complexes 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 **c**

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



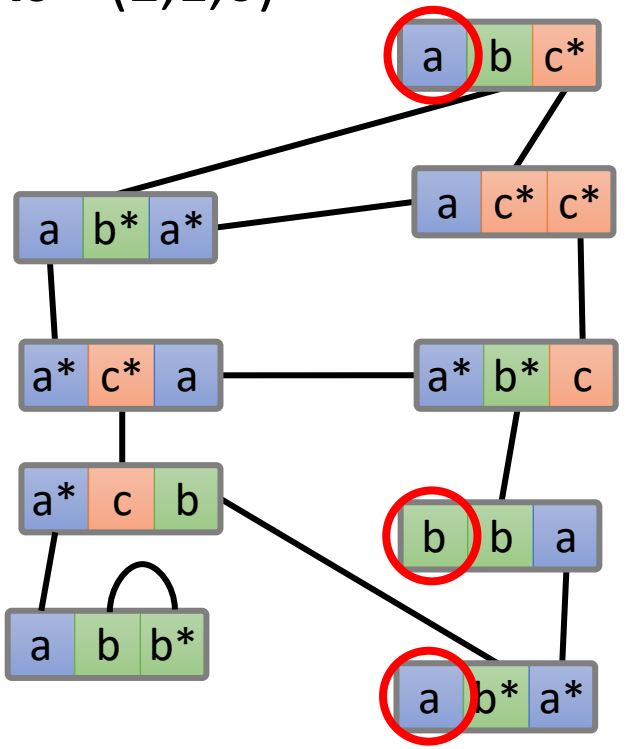
How to prove exponential complex size bound for complexes with cycles in binding graph?

- **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 **c**

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

a b c

Ac = (2,1,0)



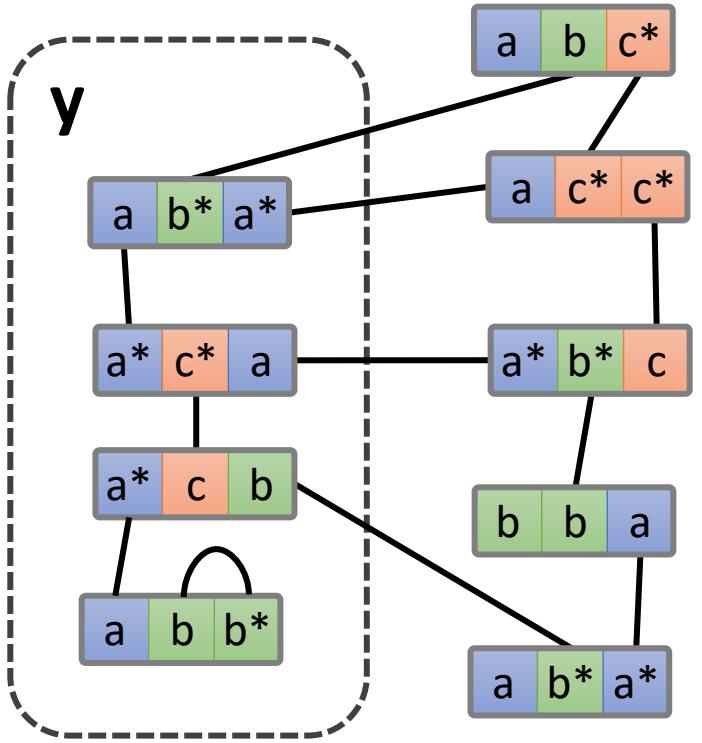
How to prove exponential complex size bound for complexes with cycles in binding graph?

- **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 **c**
- 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)



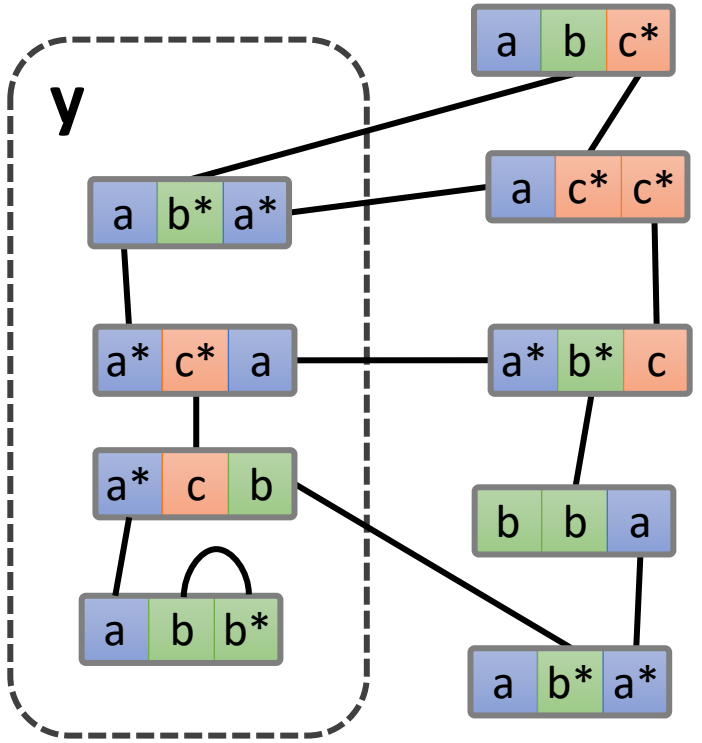
How to prove exponential complex size bound for complexes with cycles in binding graph?

- **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 **c**
- 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)



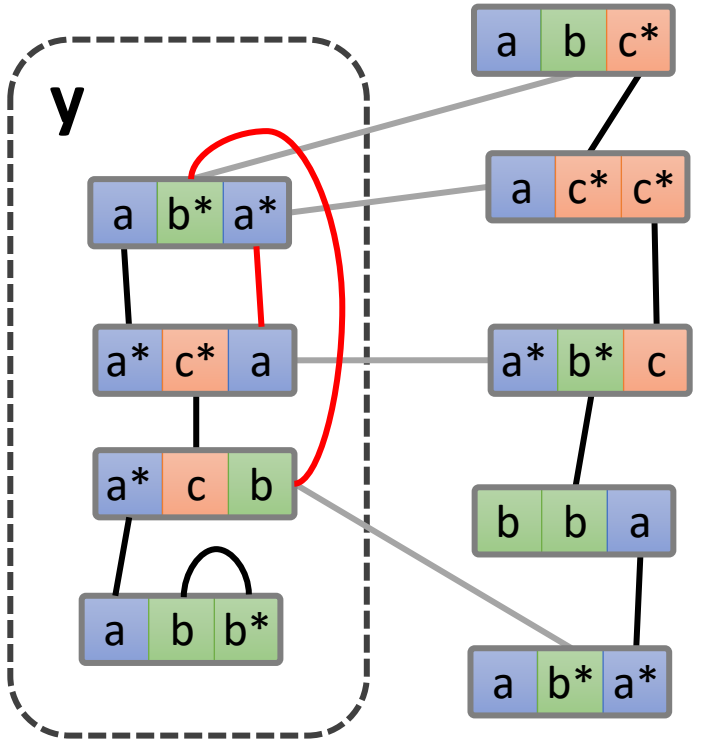
How to prove exponential complex size bound for complexes with cycles in binding graph?

- **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 **c**
- 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** $\in \mathbb{N}^M$

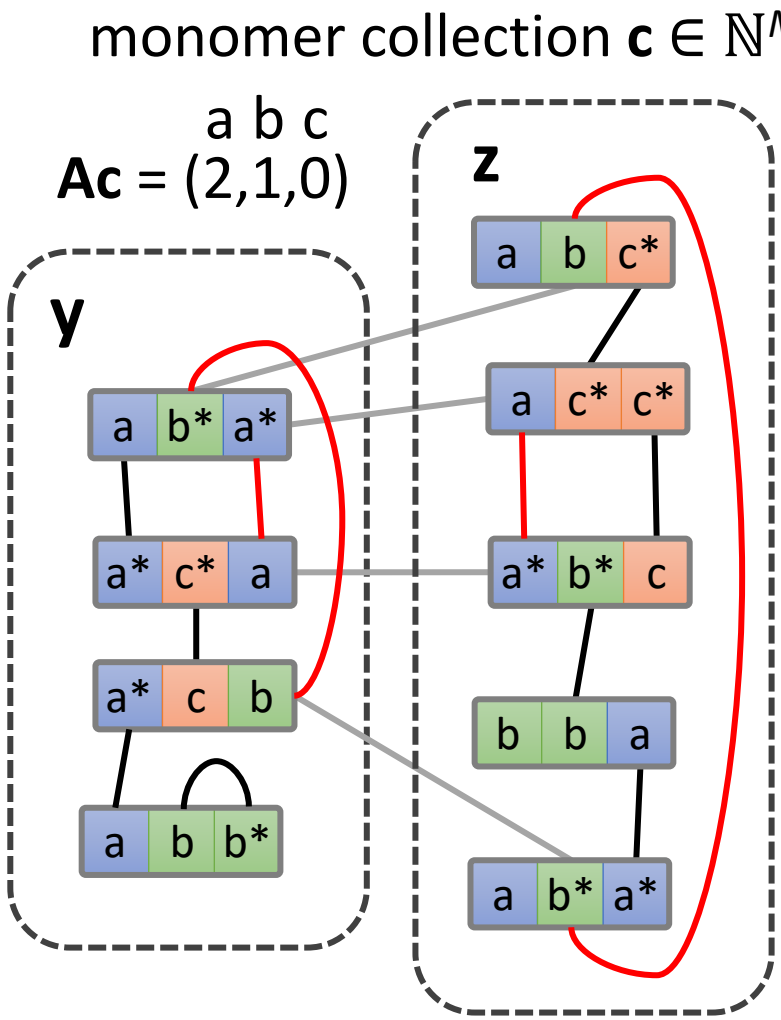
a b c

Ac = (2,1,0)



How to prove exponential complex size bound for complexes 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 **c**
- 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: **h**·**Ac** = **h**·**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 $\mathbf{h}\cdot\mathbf{m_i} = O(1)$

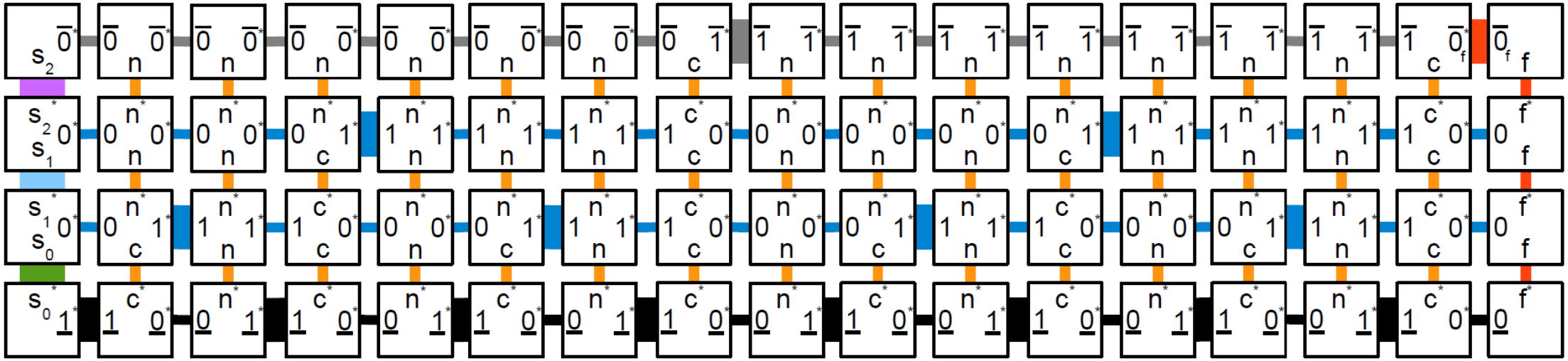
above

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

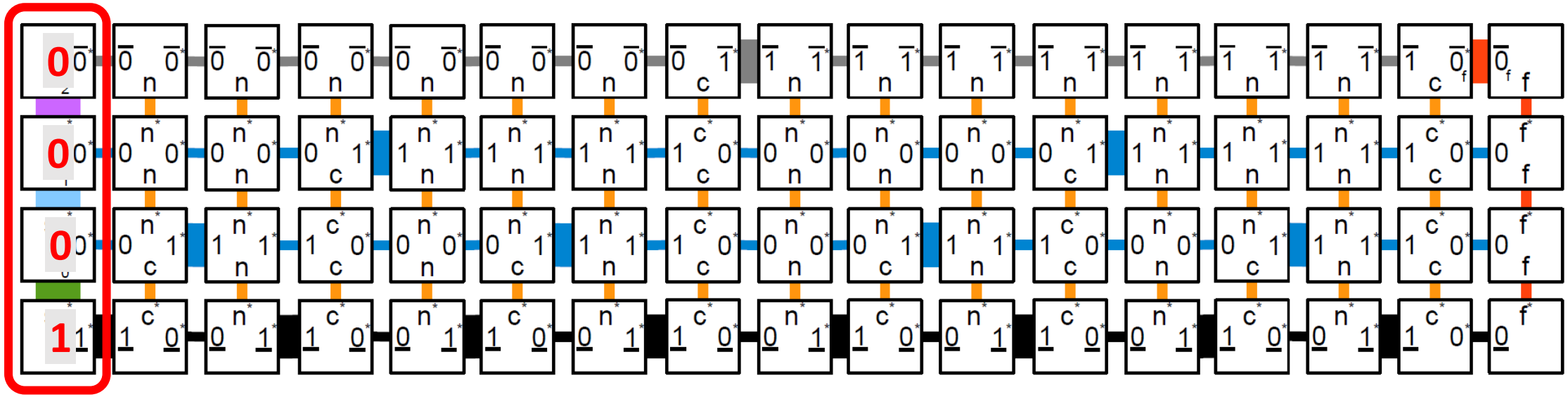
Applying thermodynamic model to tile assembly

- 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?

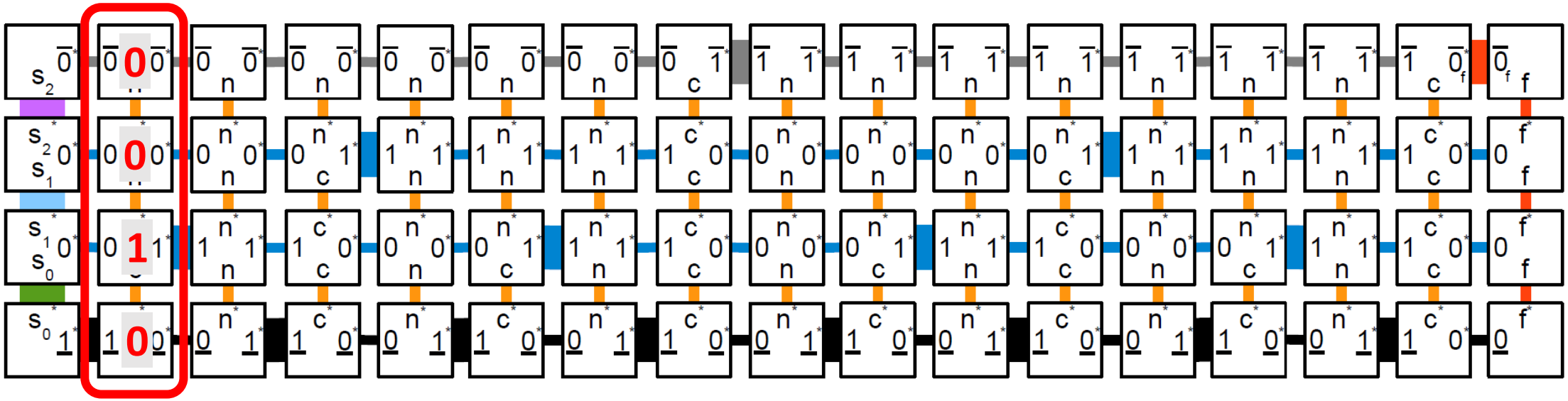
A thermodynamically unstable tile assembly counter



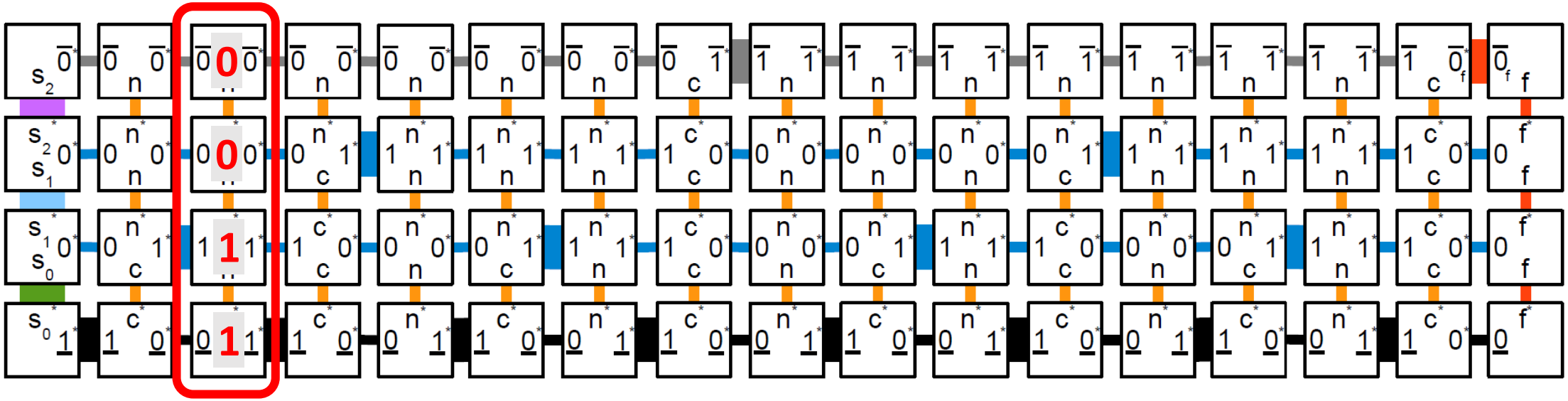
A thermodynamically unstable tile assembly counter



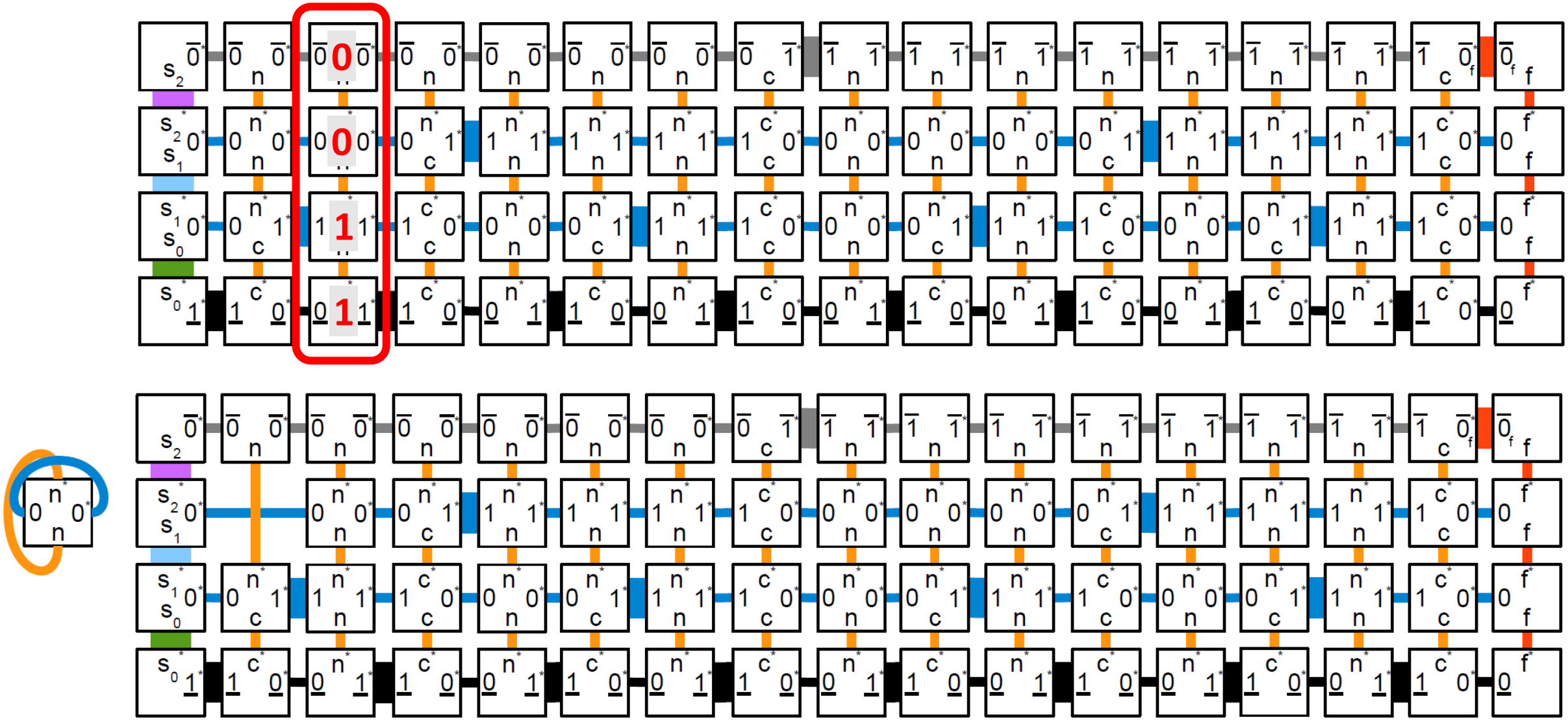
A thermodynamically unstable tile assembly counter



A thermodynamically unstable tile assembly counter

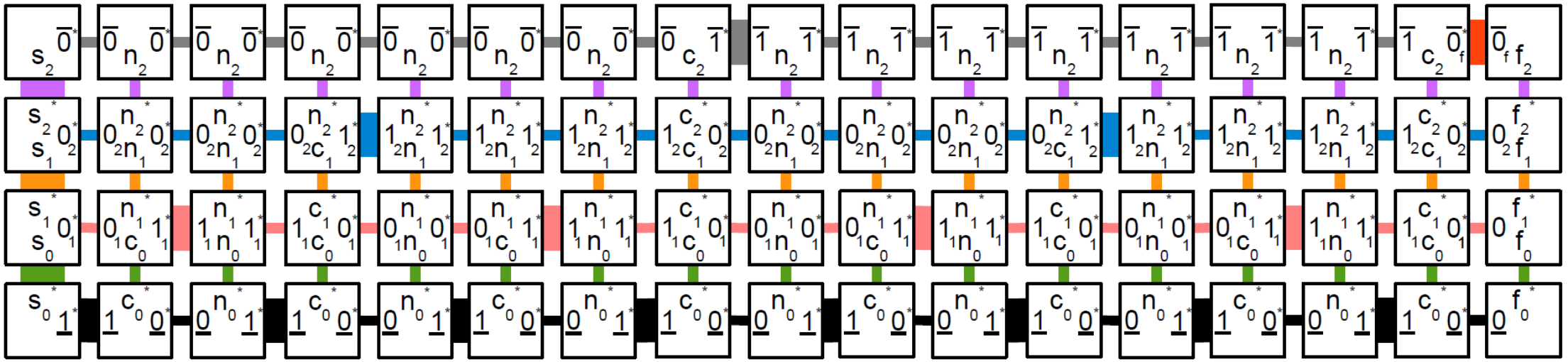


A thermodynamically unstable tile assembly counter



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



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*, Theoretical Computer Science 2019]