Introduction

slides © 2021, David Doty

ECS 232: Theory of Molecular Computation

UC Davis

Theory of molecular computation = Theory of computation ∩ nanotechnology

In principle, any nanotechnology…
In practice, DNA nanotechnology. Why?

*   limitations of my own expertise

*   DNA is naturally information bearing… because of recent technological breakthroughs, also information <u>processing</u>

*   More experimentalists in DNA nanotech interested in the idea of "making molecules compute" than in other nanotech fields.
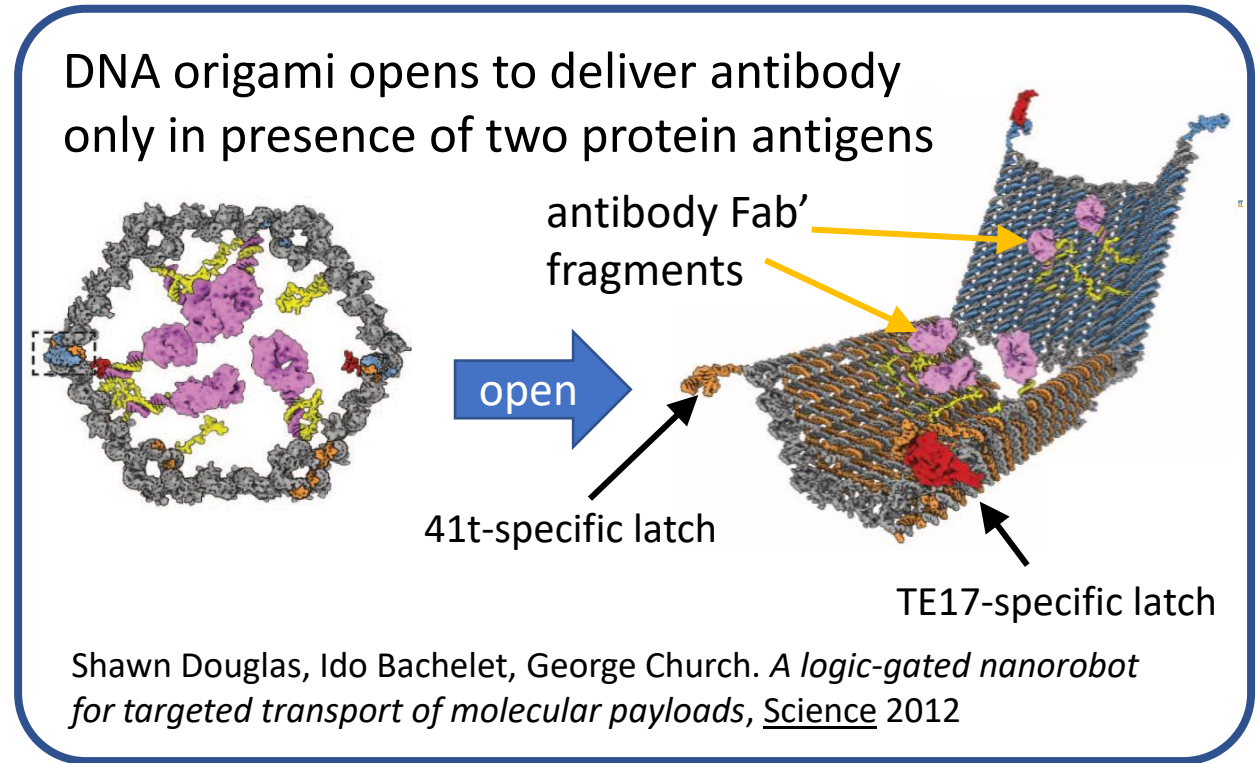
Potential DNA nanotechnology applications

nonbiological:

- nanoscale resolution surface placement
- X-ray crystallization scaffolding
- molecular motors
- super-resolution imaging
- molecular circuits

biological:
- smart drugs
- mRNA detection
- cell surface marker detection
- genetically encoded structures

Potential DNA nanotechnology applications

nonbiological:
- nanoscale resolution surface placement
- X-ray crystallization scaffolding
- molecular motors
- super-resolution imaging
- molecular circuits

biological:
- smart drugs
- mRNA detection
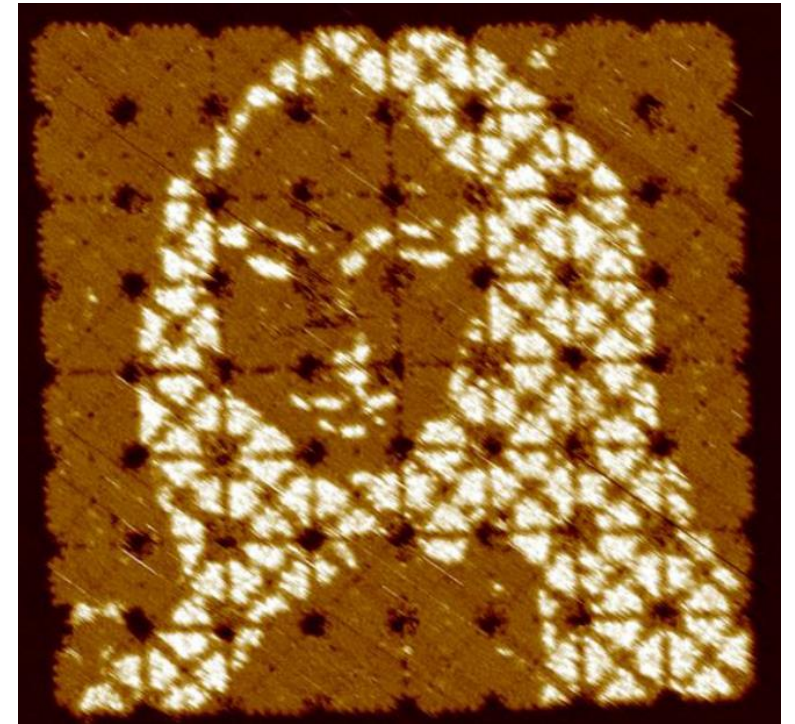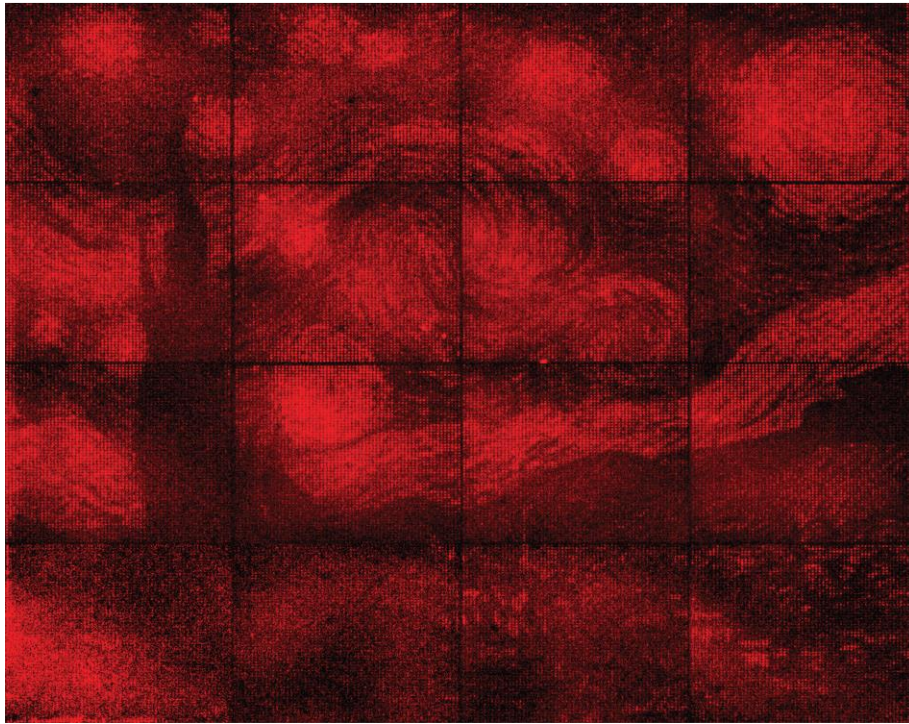- cell surface marker detection
- genetically encoded structures

example

DNA origami opens to deliver antibody only in presence of two protein antigens

antibody Fab' fragments

open

41t-specific latch

TE17-specific latch

Shawn Douglas, Ido Bachelet, George Church. *A logic-gated nanorobot for targeted transport of molecular payloads*, Science 2012

Potential DNA nanotechnology applications

nonbiological:
- ## art

Ashwin Gopinath, Evan Miyazono, Andrei Faraon, Paul Rothemund. *Engineering and mapping nanocavity emission via precision placement of DNA origami*, <u>Nature</u> 2016

Grigory Tikhomirov, Philip Petersen, and Lulu Qian. *Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns*. <u>Nature</u> 2017.

Theory of computation: Bird's eye view

Theory of computation: Bird's eye view

- <u>Church-Turing thesis</u>: *For any physical device that can be built in our universe that can 'reasonably' said to compute a function f*: {0,1}*→{0,1}*, *f is also computable by a Turing machine.*

Theory of computation: Bird's eye view

- Church-Turing thesis: *For any physical device that can be built in our universe that can 'reasonably' said to compute a function f*: {0,1}*→{0,1}*, *f is also computable by a Turing machine.*

- Not a mathematical theorem; it is a physical hypothesis.

Theory of computation: Bird's eye view

- Church-Turing thesis: *For any physical device that can be built in our universe that can 'reasonably' said to compute a function f*: $\{0,1\}^* \rightarrow \{0,1\}^*$, *f is also computable by a Turing machine.*

- Not a mathematical theorem; it is a physical hypothesis.
  - Physicists call such hypotheses "laws of nature" (always the optimists).
  - It's the connection between the physical world of *voltages, transistors,* and *time in seconds,* and the abstract ideas of *digital data, Boolean logic,* and *number of steps in an algorithm.*

Theory of computation: Bird's eye view

- <u>Church-Turing thesis</u>: *For any physical device that can be built in our universe that can 'reasonably' said to compute a function f*: $\{0,1\}^* \rightarrow \{0,1\}^*$*, f is also computable by a Turing machine.*

- Not a mathematical theorem; it is a physical hypothesis.
  - Physicists call such hypotheses "laws of nature" (always the optimists).
  - It's the connection between the <u>physical world</u> of *voltages*, *transistors*, and *time in seconds*, and the <u>abstract ideas</u> of *digital data*, *Boolean logic*, and *number of steps in an algorithm*.

- If no Turing machine can solve a computational problem, then no physical device can solve it either.

Theory of computation: Bird's eye view

- Church-Turing thesis: *For any physical device that can be built in our universe that can 'reasonably' said to compute a function f*: $\{0,1\}^* \rightarrow \{0,1\}^*$, *f is also computable by a Turing machine.*

- Not a mathematical theorem; it is a physical hypothesis.
  - Physicists call such hypotheses "laws of nature" (always the optimists).
  - It's the connection between the physical world of *voltages*, *transistors*, and *time in seconds*, and the abstract ideas of *digital data*, *Boolean logic*, and *number of steps in an algorithm*.

- If no Turing machine can solve a computational problem, then no physical device can solve it either.

- Some problems *are* inherently difficult (Boolean satisfiability) or impossible (Halting problem) for computers to solve.
  - Fact about the problem itself, not about a lack of human ingenuity.

Theory of computation: Bird's eye view

- Many models of computation:
    - Turing machine
    - finite state machine
    - polynomial-time Turing machine
    - Boolean circuit
    - distributed network with limited communication between nodes but unlimited computational ability at each node

Theory of computation: Bird's eye view

- Many models of computation:
  - Turing machine
  - finite state machine
  - polynomial-time Turing machine
  - Boolean circuit
  - distributed network with limited communication between nodes but unlimited computational ability at each node
- None is more or less "correct" as a model of <u>all computing devices</u>.
  - Each is an abstraction useful in some contexts, like Newtonian spherical, frictionless cows.

Theory of computation: Bird's eye view

- Many models of computation:
  - Turing machine
  - finite state machine
  - polynomial-time Turing machine
  - Boolean circuit
  - distributed network with limited communication between nodes but unlimited computational ability at each node
- None is more or less "correct" as a model of <u>all computing devices</u>.
  - Each is an abstraction useful in some contexts, like Newtonian spherical, frictionless cows.
- Consider: no finite state machine can decide if a binary string has equal 0s and 1s
  - <u>Overly literal interpretation</u>: Your laptop has finite memory, so cannot solve this problem.

Theory of computation: Bird's eye view

- Many models of computation:
  - Turing machine
  - finite state machine
  - polynomial-time Turing machine
  - Boolean circuit
  - distributed network with limited communication between nodes but unlimited computational ability at each node
- None is more or less "correct" as a model of <u>all computing devices</u>.
  - Each is an abstraction useful in some contexts, like Newtonian spherical, frictionless cows.
- Consider: no finite state machine can decide if a binary string has equal 0s and 1s
  - <u>Overly literal interpretation</u>: Your laptop has finite memory, so cannot solve this problem.
  - <u>Better interpretation</u>: If you write an algorithm to solve this problem, then it must, somewhere, use unbounded memory (list, stack, recursion, etc.)

Theory of molecular computation: Bird's eye view

Goal of course: Apply the Theory of Computing "lens" to molecular engineering.

Topics we'll cover:

Theory of molecular computation: Bird's eye view

Goal of course: Apply the Theory of Computing "lens" to molecular engineering.

Topics we'll cover:

- Basic experimental background
  - structural DNA nanotechnology (building things from molecules)
    - DNA tile self-assembly
    - DNA origami

Theory of molecular computation: Bird's eye view

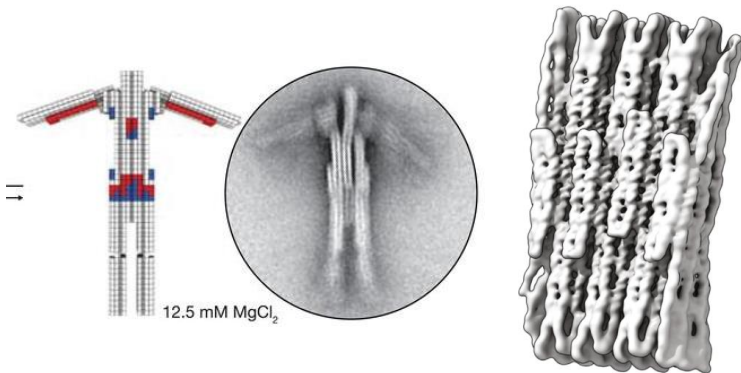Goal of course: Apply the Theory of Computing "lens" to molecular engineering.

Topics we'll cover:

- Basic experimental background
  - structural DNA nanotechnology (building things from molecules)
    - DNA tile self-assembly
    - DNA origami
  - dynamic DNA nanotechnology (reconfiguring molecules)
    - DNA strand displacement

Theory of molecular computation: Bird's eye view

Goal of course: Apply the Theory of Computing "lens" to molecular engineering.

Topics we'll cover:

- Basic experimental background
  - structural DNA nanotechnology (building things from molecules)
    - DNA tile self-assembly
    - DNA origami
  - dynamic DNA nanotechnology (reconfiguring molecules)
    - DNA strand displacement
- Algorithmic tile self-assembly

Theory of molecular computation: Bird's eye view

Goal of course: Apply the Theory of Computing "lens" to molecular engineering.

Topics we'll cover:

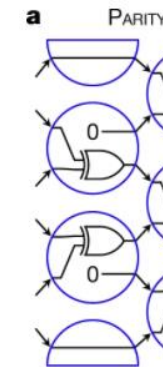- Basic experimental background
  - structural DNA nanotechnology (building things from molecules)
    - DNA tile self-assembly
    - DNA origami
  - dynamic DNA nanotechnology (reconfiguring molecules)
    - DNA strand displacement
- Algorithmic tile self-assembly
- Chemical reaction networks (and relation to distributed computing)

Theory of molecular computation: Bird's eye view

Goal of course: Apply the Theory of Computing "lens" to molecular engineering.

Topics we'll cover:

- Basic experimental background
  - structural DNA nanotechnology (building things from molecules)
    - DNA tile self-assembly
    - DNA origami
  - dynamic DNA nanotechnology (reconfiguring molecules)
    - DNA strand displacement

- Algorithmic tile self-assembly

- Chemical reaction networks (and relation to distributed computing)

- Thermodynamic binding networks

Theory of molecular computation: Bird's eye view

Goal of course: Apply the Theory of Computing "lens" to molecular engineering.

Topics we'll cover:

- Basic experimental background
  - structural DNA nanotechnology (building things from molecules)
    - DNA tile self-assembly
    - DNA origami
  - dynamic DNA nanotechnology (reconfiguring molecules)
    - DNA strand displacement

- Algorithmic tile self-assembly

- Chemical reaction networks (and relation to distributed computing)

- Thermodynamic binding networks

- DNA sequence design (classical algorithmic problem helpful in experiments)

Basic experimental background: Structural DNA nanotechnology
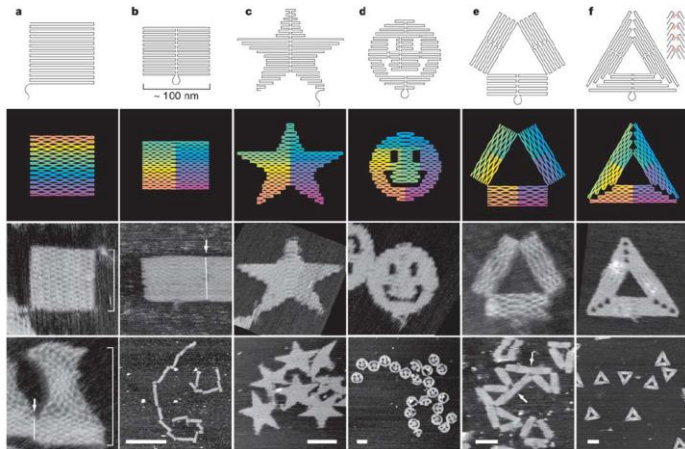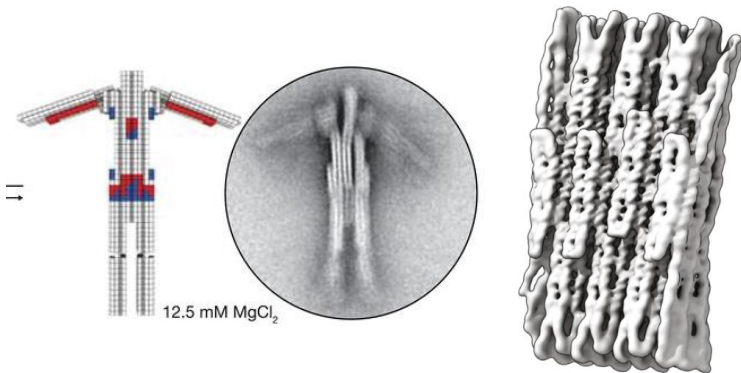
DNA origami

Basic experimental background: Structural DNA nanotechnology
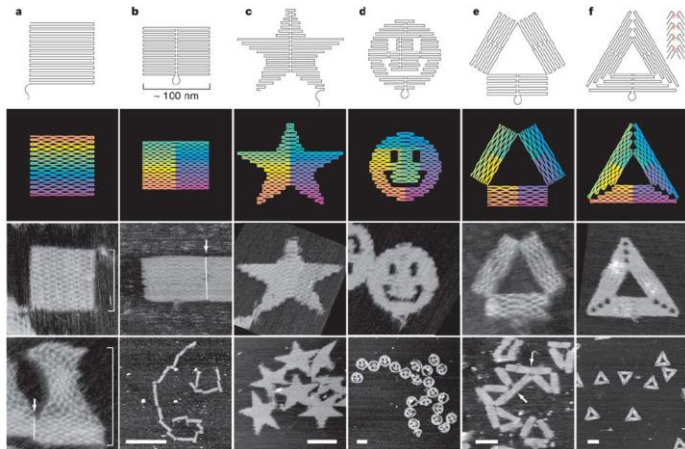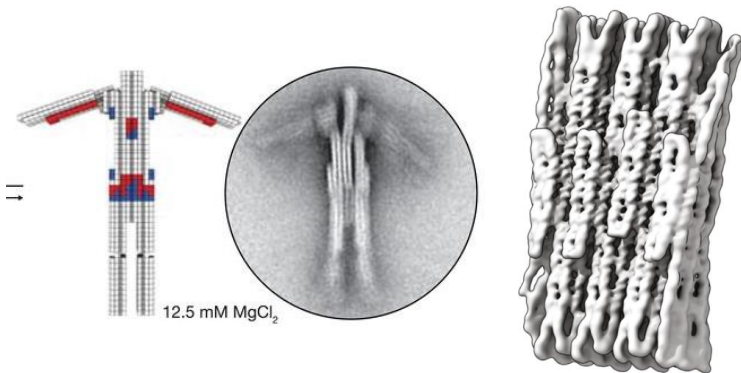
DNA origami

DNA tile self-assembly

Basic experimental background: Structural DNA nanotechnology

DNA origami

https://www.nature.com/articles/nature04586

12.5 mM MgCl₂

https://science.sciencemag.org/content/347/6229/1446.full
https://www.nature.com/articles/s41467-020-20020-7

DNA tile self-assembly

Molecular canvas

Eagle head

Triangle

https://www.nature.com/articles/nature11075
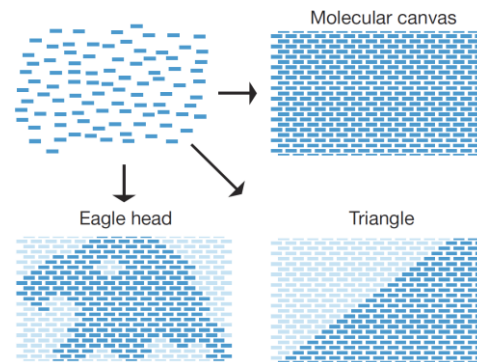
algorithmic DNA tile self-assembly

g   SORTING circuit layer

h   Input: 000001, Output: 100000

a   PARITY

Is the number of 1s odd?

Yes

No

Yes

https://www.dna.caltech.edu/Papers/cge-thesis2014.pdf
https://www.nature.com/articles/s41586-019-1014-9

Basic experimental background: Dynamic DNA nanotechnology

DNA strand displacement: DNA reconfiguring itself without enzymes

Basic experimental background: Dynamic DNA nanotechnology

DNA strand displacement: DNA reconfiguring itself without enzymes

Basic experimental background: Dynamic DNA nanotechnology

DNA strand displacement: DNA reconfiguring itself without enzymes

Basic experimental background: Dynamic DNA nanotechnology

DNA strand displacement: DNA reconfiguring itself without enzymes

8

Basic experimental background:
Dynamic DNA nanotechnology

DNA strand displacement: DNA reconfiguring itself without enzymes

DNA strand displacement
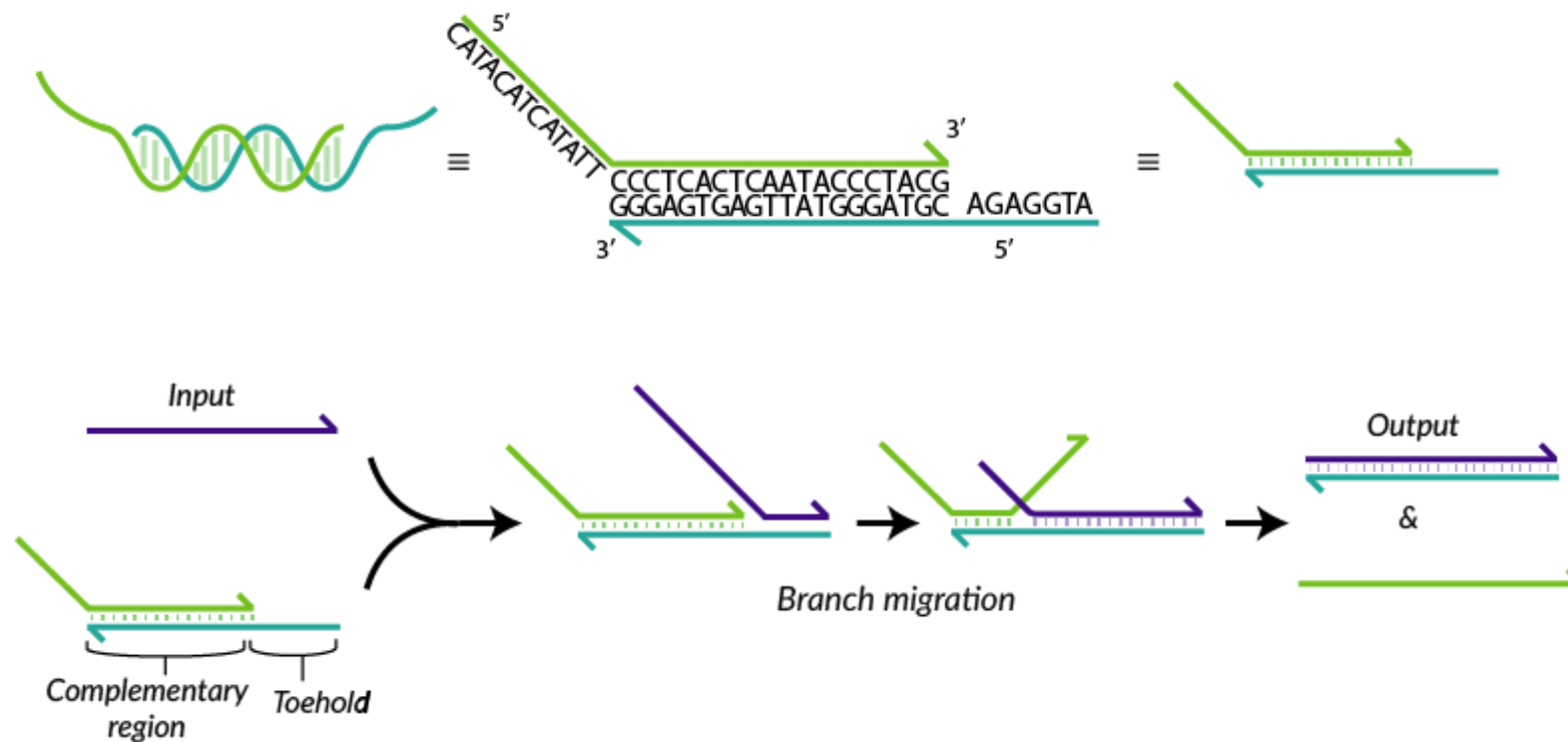
https://www.microsoft.com/en-us/research/video/dna-strand-displacement/

DNA strand displacement

https://www.microsoft.com/en-us/research/video/dna-strand-displacement/

Algorithmic tile self-assembly
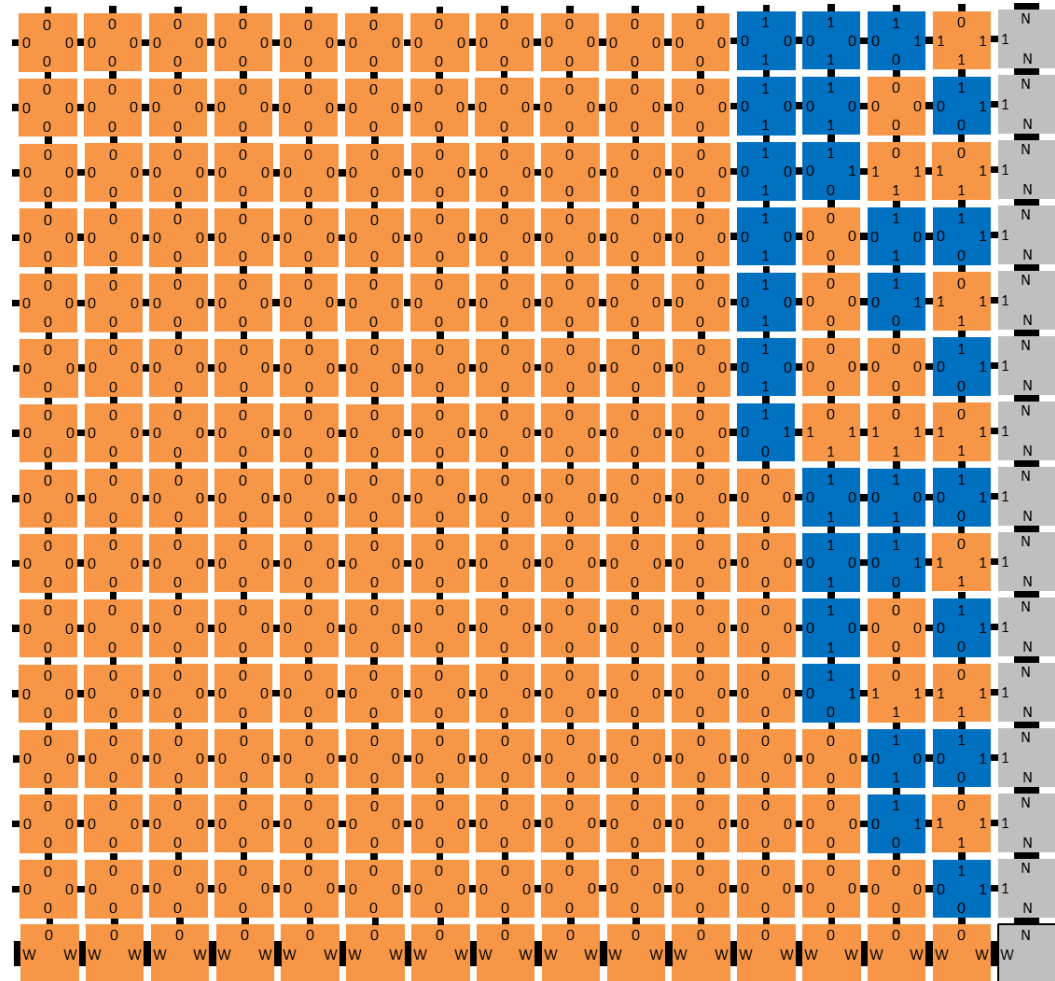
Algorithmic tile self-assembly
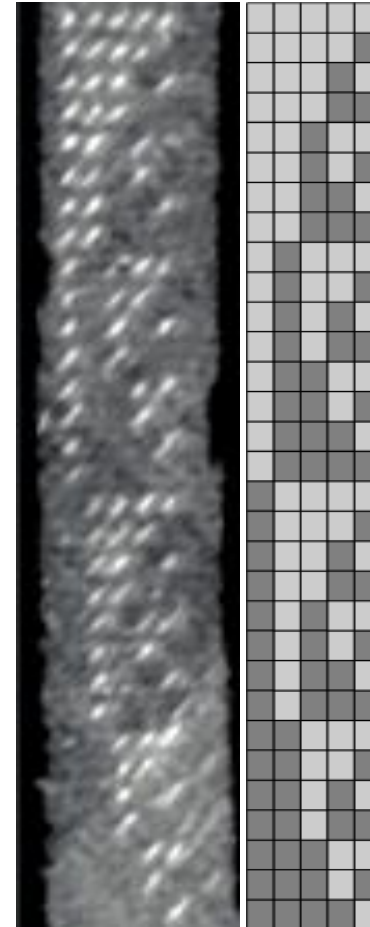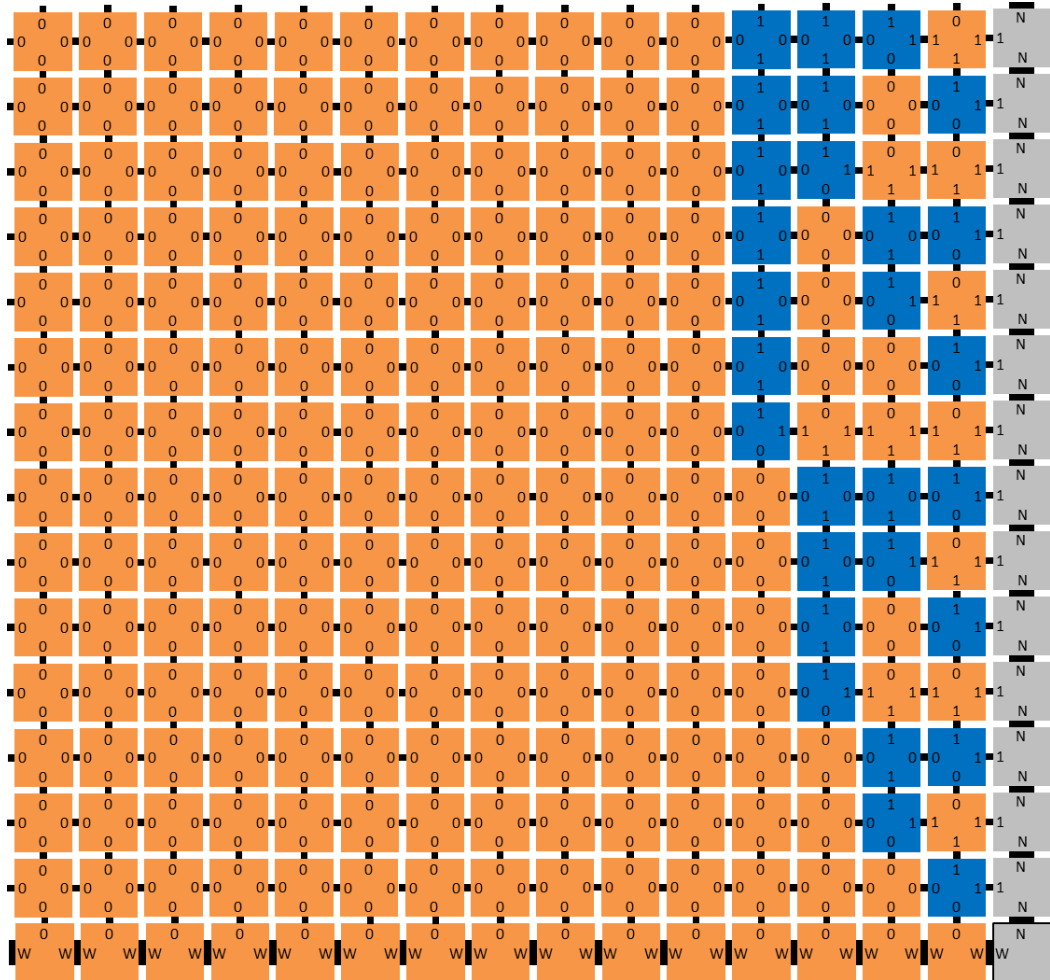
Chemical reaction networks

Chemical reaction networks

reactant(s) $\qquad$ $R \rightarrow P_1 + P_2$ $\qquad$ *product(s)*

Chemical reaction networks

reactant(s) $R \rightarrow P_1 + P_2$ *product(s)*

monomers $M_1 + M_2 \rightarrow D$ *dimer*

Chemical reaction networks

reactant(s) $\qquad R \rightarrow P_1 + P_2$ $\qquad$ *product(s)*

monomers $\qquad M_1 + M_2 \rightarrow D$ $\qquad$ *dimer*

catalyst $\qquad C + X \rightarrow C + Y$

Chemical reaction networks

reactant(s) $R \rightarrow P_1 + P_2$ product(s)

monomers $M_1 + M_2 \rightarrow D$ dimer

catalyst $C + X \rightarrow C + Y$
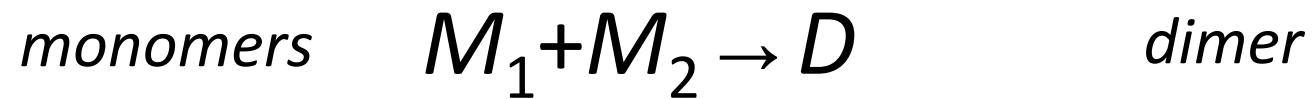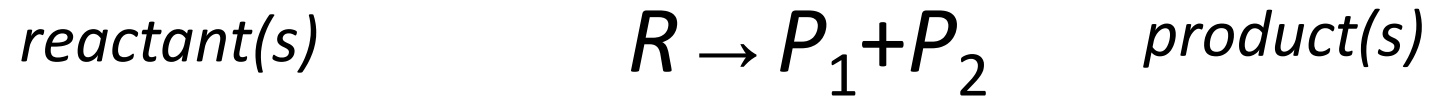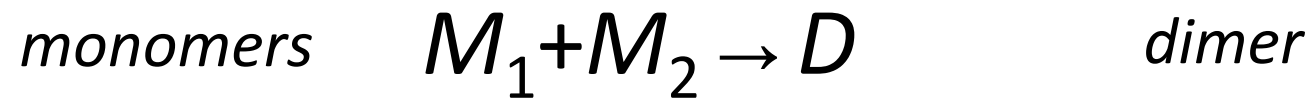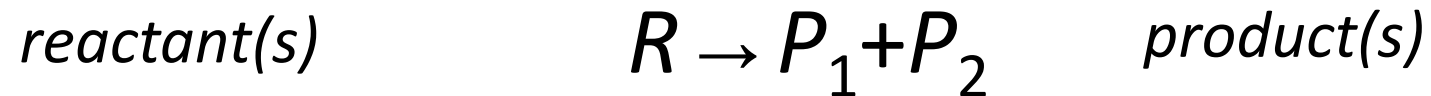
Traditionally a descriptive modeling language…
Let's instead use it as a prescriptive programming language

Chemical caucusing
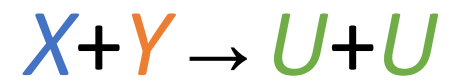
$$X + Y \rightarrow U + U$$

[Angluin, Aspnes, Eisenstat, A simple population protocol for fast robust approximate majority, *DISC* 2007]

Chemical caucusing

$$X + Y \rightarrow U + U$$

$$X + U \rightarrow X + X$$

$$Y + U \rightarrow Y + Y$$

[Angluin, Aspnes, Eisenstat,   A simple population protocol for fast robust approximate majority,   *DISC* 2007]

Chemical caucusing

$X + Y \rightarrow U + U$

$X + U \rightarrow X + X$

$Y + U \rightarrow Y + Y$
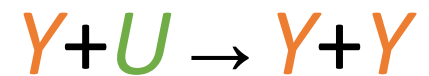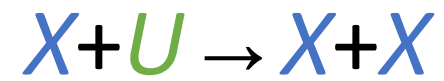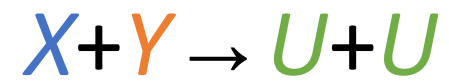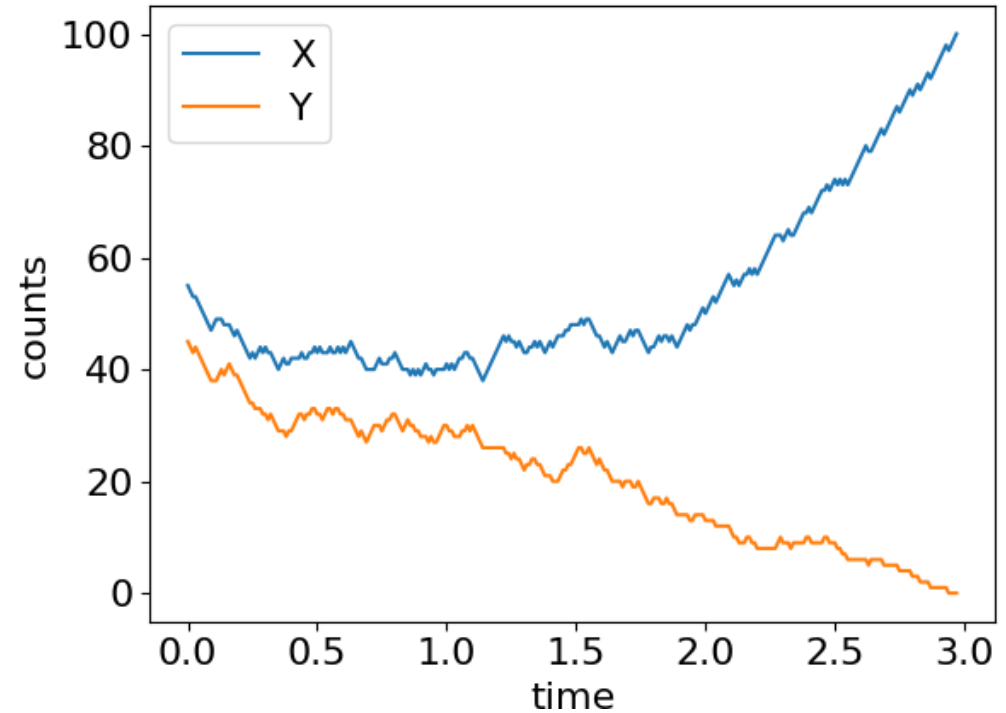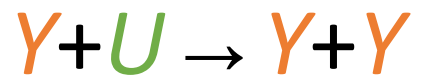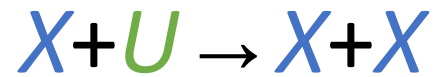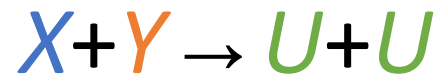
distributed algorithm for *"approximate majority"*:
initial majority (*X* or *Y*) quickly overtakes whole population

[Angluin, Aspnes, Eisenstat,   A simple population protocol for fast robust approximate majority,   *DISC* 2007]

Thermodynamic binding networks

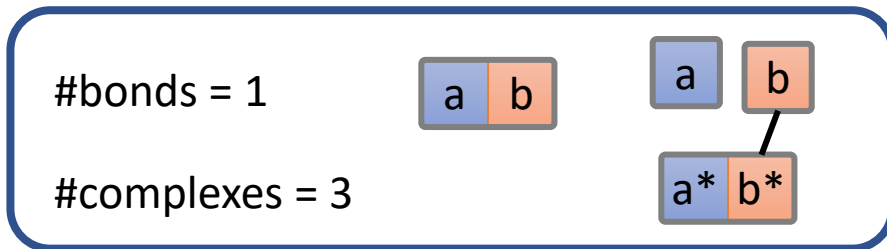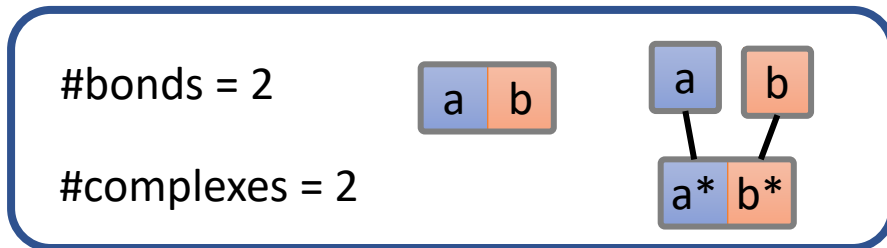<u>Goal of model</u>: abstract away geometry of DNA to understand effect of two contributions to energetics:

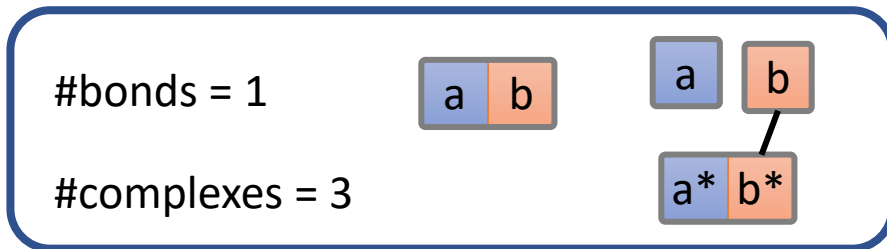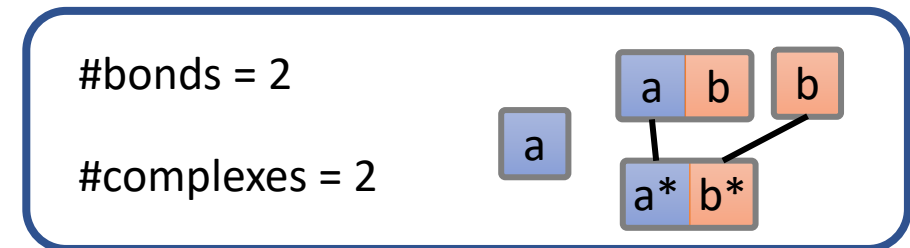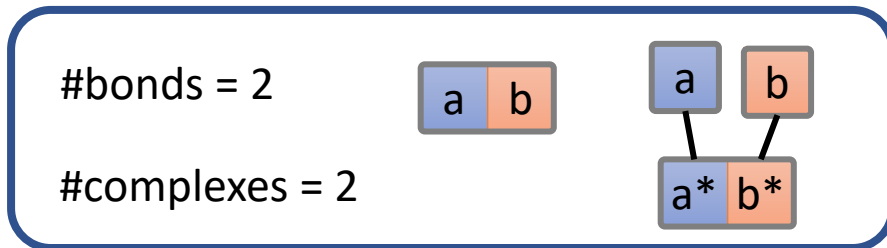1. number of bonds formed ("enthalpy")
2. number of separate complexes ("entropy")

#bonds = 2

#complexes = 2

Thermodynamic binding networks

Goal of model: abstract away geometry of DNA to understand effect of two contributions to energetics:

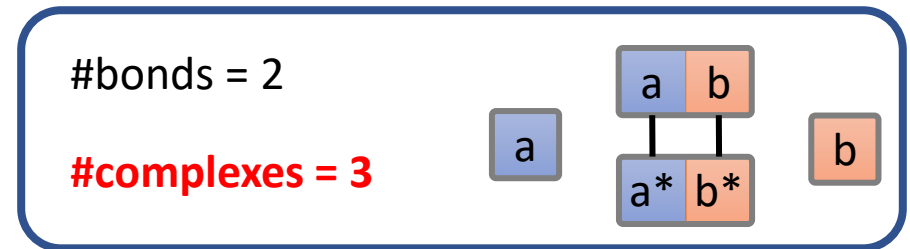1. number of bonds formed ("enthalpy")
2. number of separate complexes ("entropy")

#bonds = 2

#complexes = 2
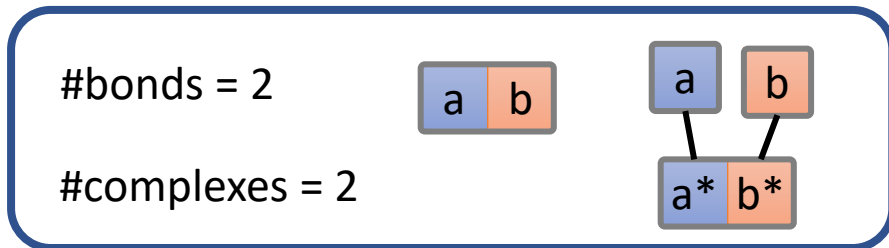
#bonds = 1

#complexes = 3

Thermodynamic binding networks

Goal of model: abstract away geometry of DNA to understand effect of two contributions to energetics:
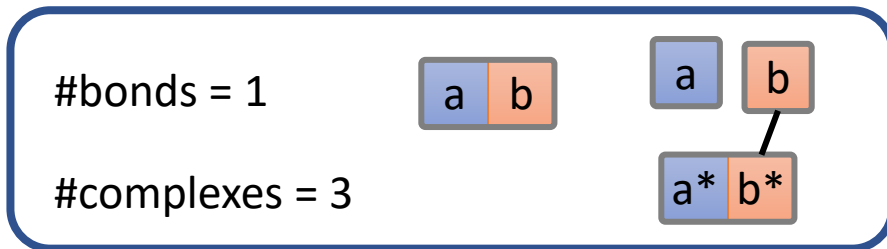
1. number of bonds formed ("enthalpy")
2. number of separate complexes ("entropy")

Thermodynamic binding networks

Goal of model: abstract away geometry of DNA to understand effect of two contributions to energetics:
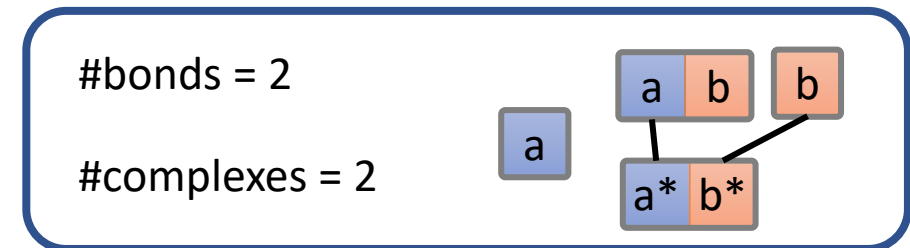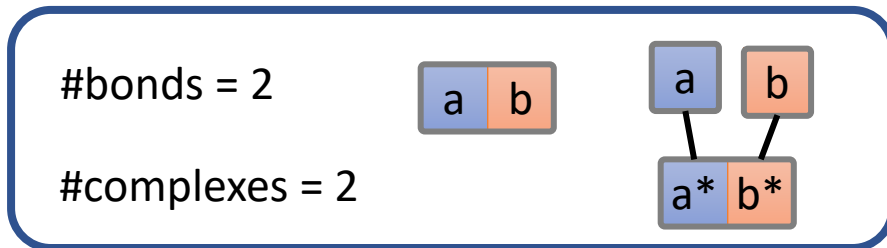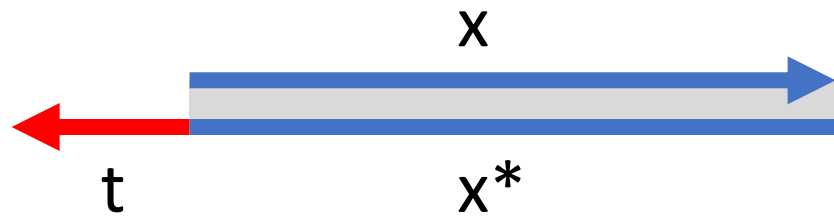
1. number of bonds formed ("enthalpy")
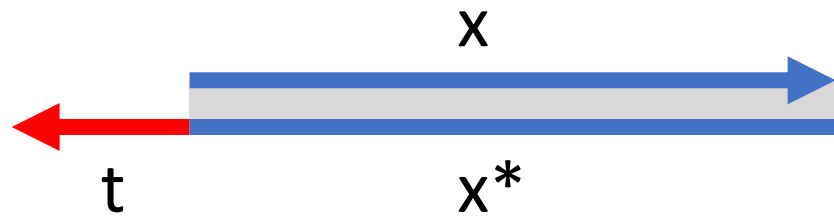2. number of separate complexes ("entropy")

#bonds = 2

#complexes = 2

#bonds = 2

#complexes = 2

#bonds = 1

#complexes = 3

#bonds = 2

#complexes = 3

DNA sequence design

Goal: given strands with
abstract "binding domains"

x

t   x*

https://link.springer.com/chapter/10.1007/978-3-540-74126-8_30

DNA sequence design

Goal: given strands with
abstract "binding domains"

x

t   x*

assign to domains
concrete DNA sequences:

ACGTACGTACGTACGTAC

ACTTATGCATGCATGCATG

DNA sequence design

Goal: given strands with
abstract "binding domains"

while preventing unwanted
interactions (e.g., strand
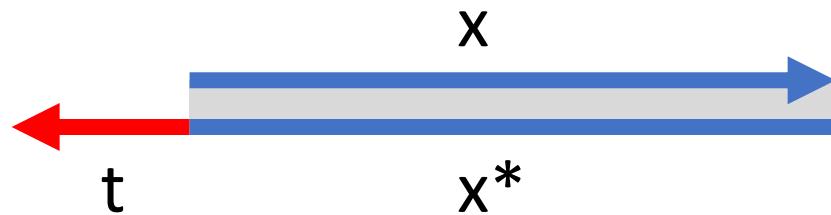folds up on itself)

assign to domains
concrete DNA sequences:

DNA sequence design

Goal: given strands with
abstract "binding domains"
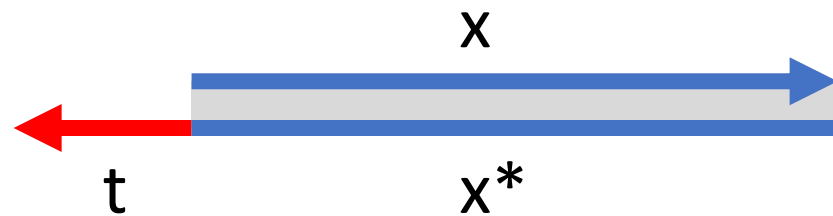

x

x*

t

assign to domains
concrete DNA sequences:

ACGTACGTACGTACGTAC

ACTTATGCATGCATGCATGCATG

while preventing unwanted
interactions (e.g., strand
folds up on itself)

CCGGCCGGCCAAT

GGCCGGCCGG

TTTTT
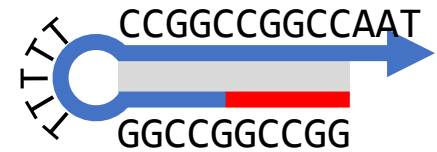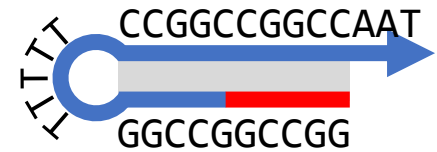
using combinatorial predictive models of DNA energy:

14
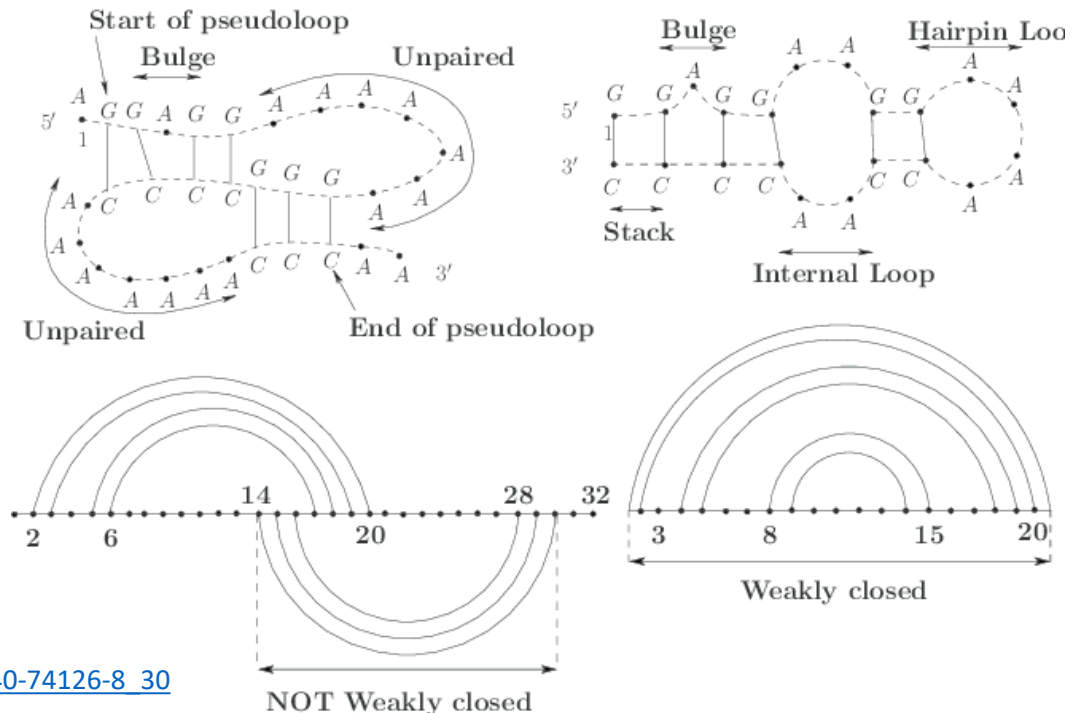
Logistics

Grading

- 60% homework
- 30% final project
- 10% in-class participation

Prerequisites

- Undergraduate theory of computation (ECS 120)
  - experience with proofs, formal definitions, discrete math (*sets, sequences, finite strings, graphs, big-O notation*)
  - finite automata, Turing machines, computability (*halting problem, decidable versus computably enumerable languages*) and complexity theory (*polynomial-time,* **NP***-completeness*)
- Probability
  - events
  - random variables
  - expected value