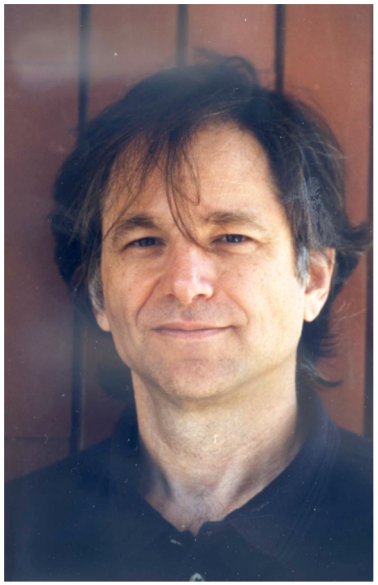
Molecular Programming Models

A little history; three models of computing with DNA; Chemical Reaction Networks

Based on notes by Dave Doty

History

- Feynmann 1959: "There's plenty of room at the bottom" envisioned "manipulating and controlling things on a small scale”
- Adleman 1994: “Computing with DNA” solved simple Hamiltonian path instance in a test tube



“remarkable energy efficiency ... information density of approximately 1 bit per cubic nanometre ... massively parallel...”

History

Lloyd Smith

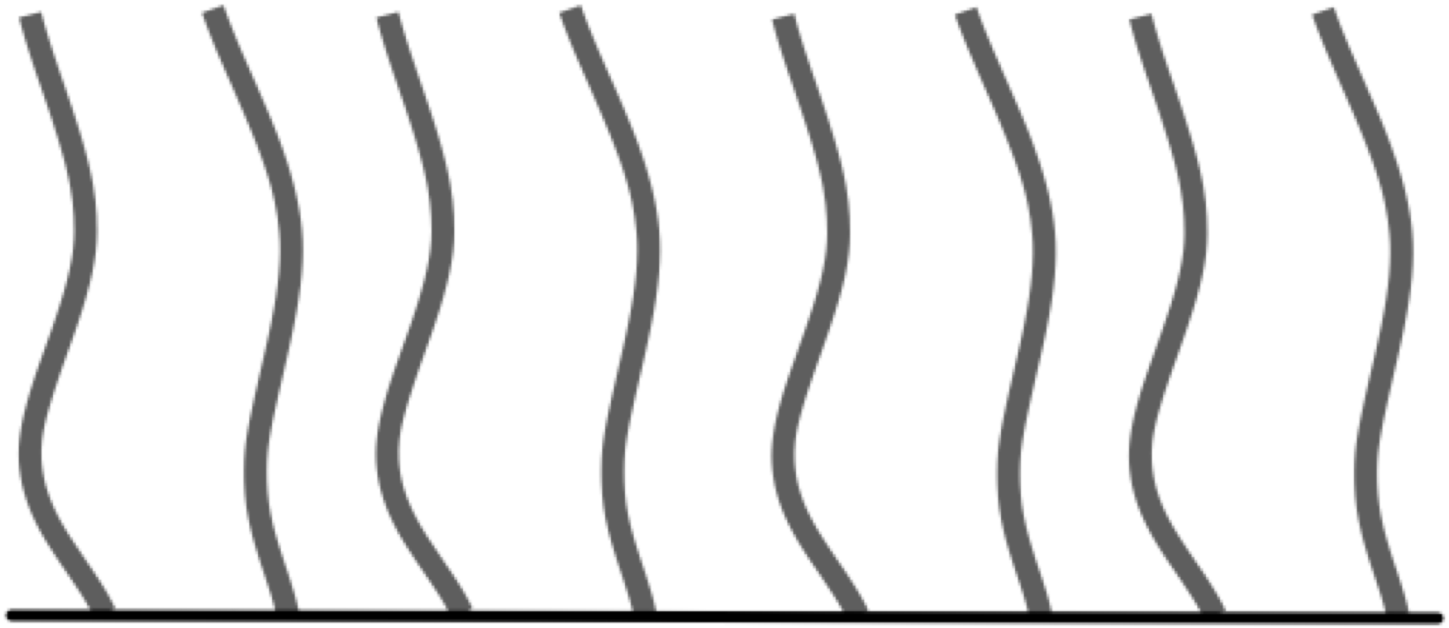
"DNA computing can be put in a form which is very amenable to automation"

Rob Corn

"Surfaces are a great place to keep track of molecules. We are learning to store and manipulate information in DNA"

Surface-based DNA computing

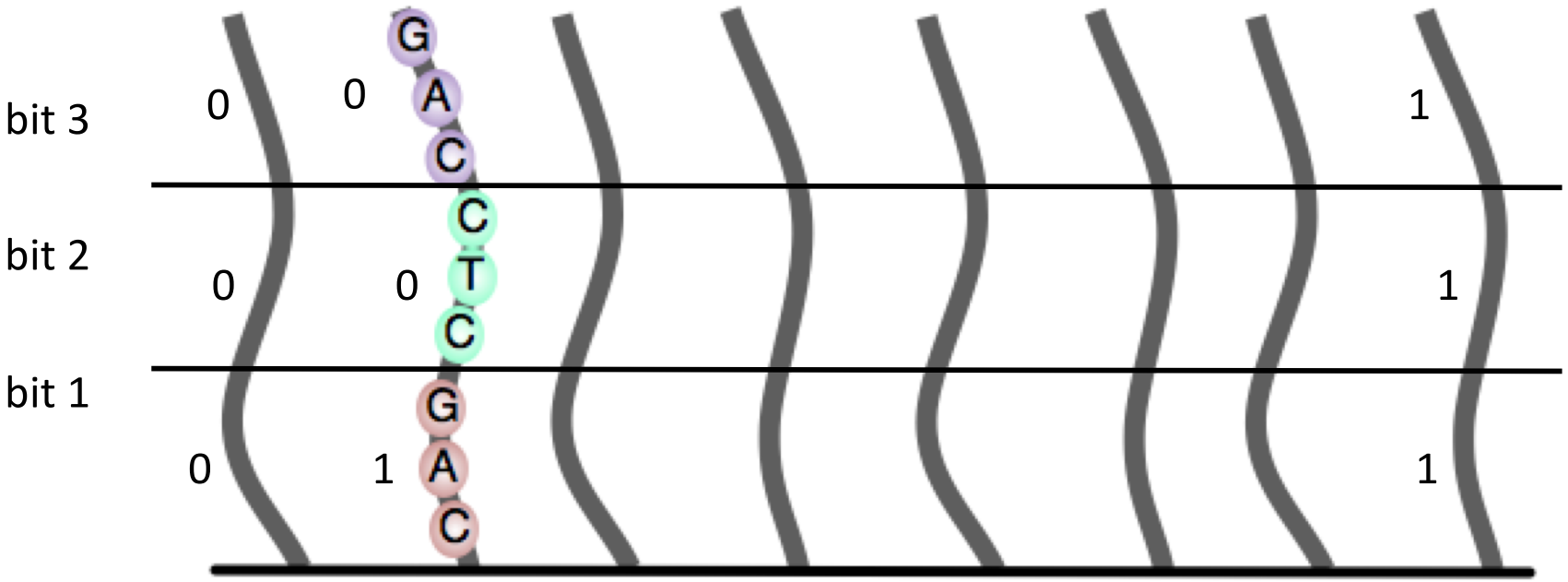
DNA strands are placed on a surface



Liu et al., Nature, 2000

Surface-based DNA computing

Strands encode binary strings

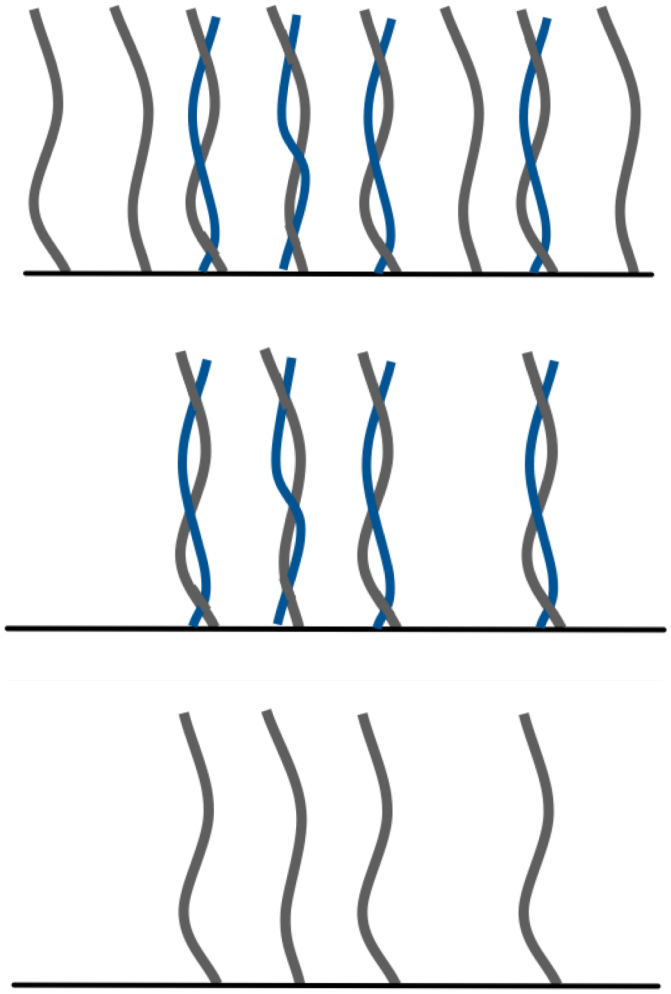


Liu et al., Nature, 2000

Surface-based DNA computing

Three “instructions”:

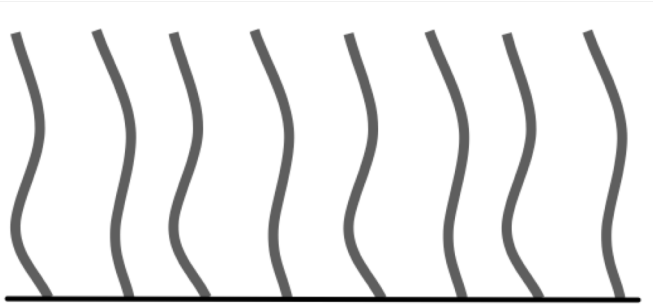
- *mark*(i,b): make all strands with bit i set to b double-stranded
- *destroy*: erase single-stranded molecules
- *unmark*: make all molecules single-stranded



Liu et al., Nature, 2000

Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

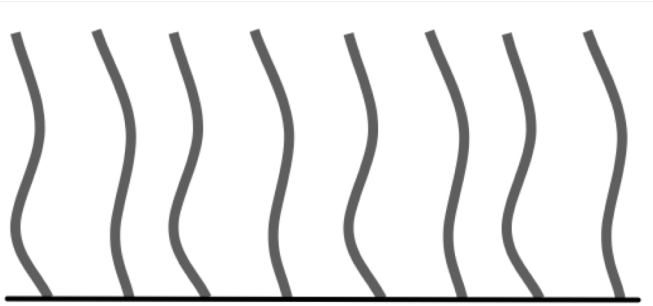


- Initially strands representing all truth assignments are on the surface
- Goal is to erase unsatisfying truth assignments

Liu et al., Nature, 2000

Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

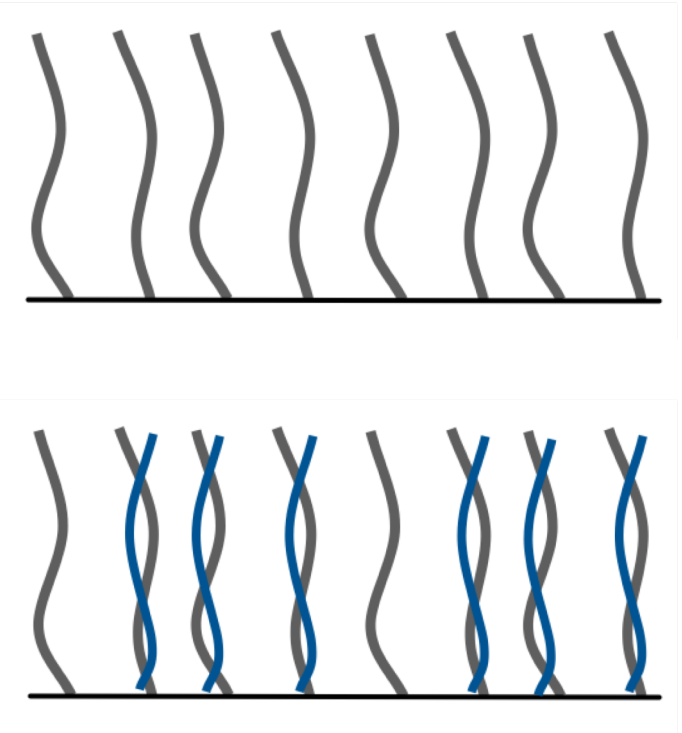


- Clause $(x_1 \vee x_2)$:
  - *mark* strands with bits 1 or 2 set to “1”

Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

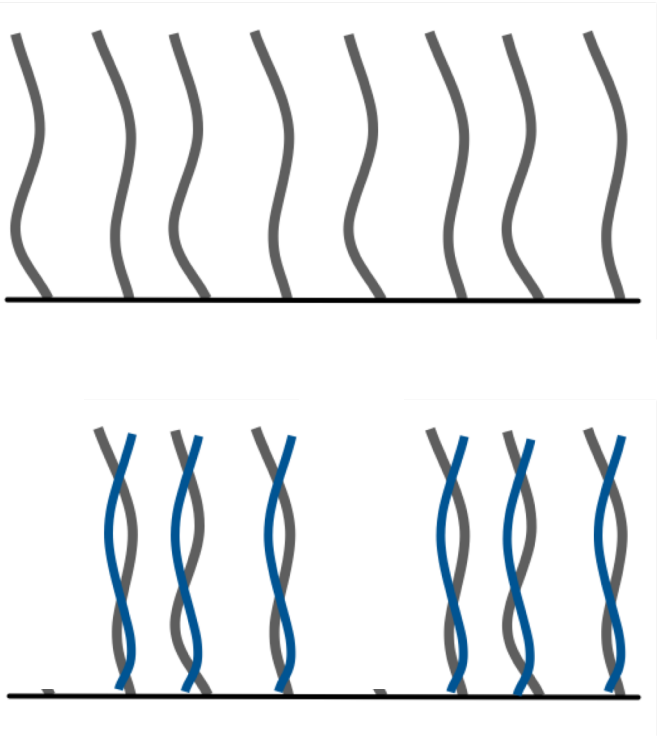
- Clause $(x_1 \vee x_2)$:
  - *mark* strands with bits 1 or 2 set to “1”



Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

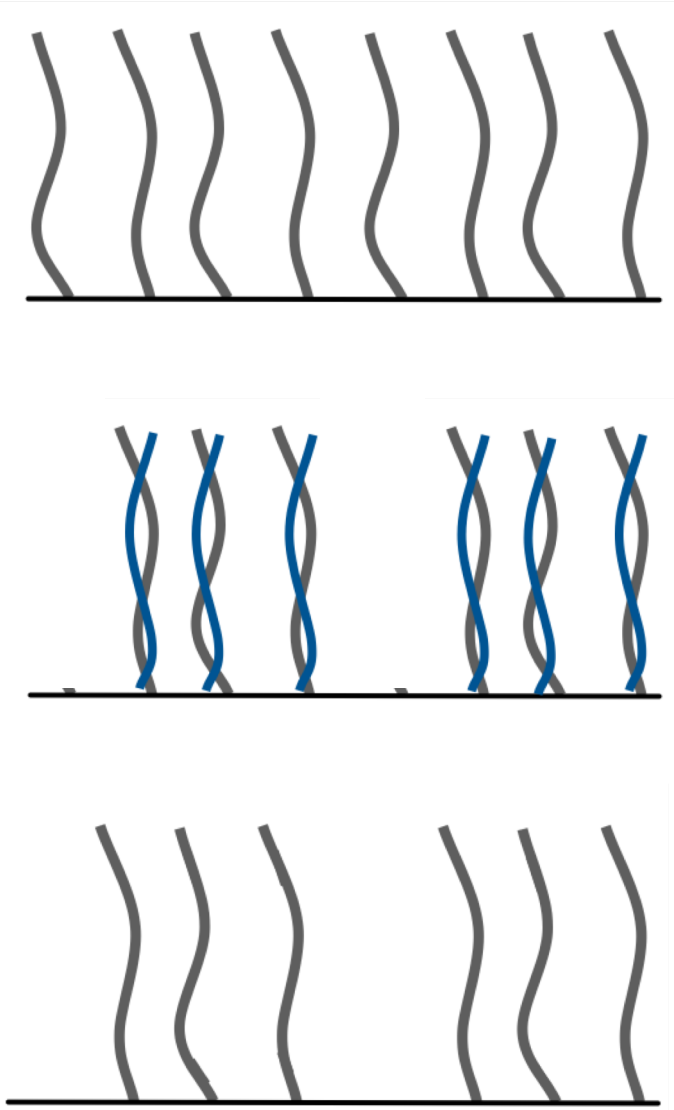
- Clause $(x_1 \vee x_2)$:
  - *mark* strands with bits 1 or 2 set to "1"
  - *destroy*



Surface-based DNA computing

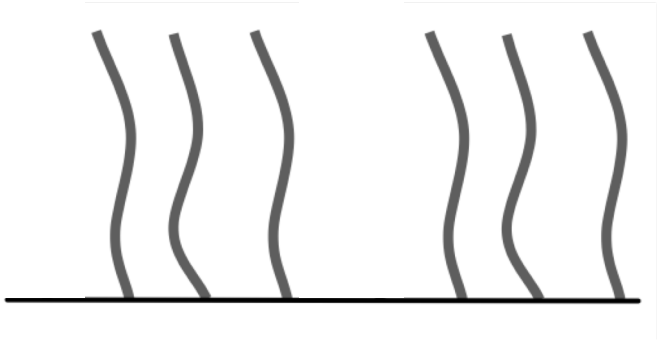
Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

- Clause $(x_1 \vee x_2)$:
  - *mark* strands with bits 1 or 2 set to "1"
  - *destroy*
  - *unmark*



Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

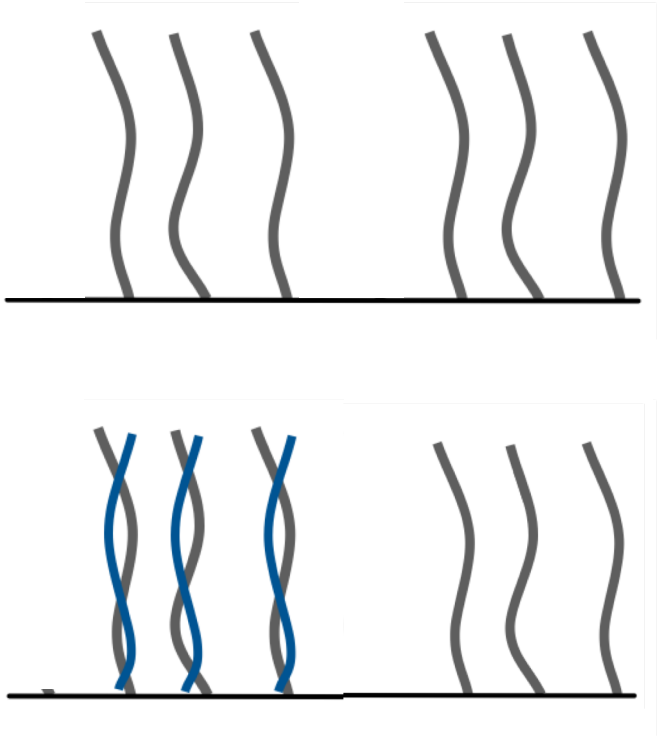


- Clause: $(\bar{x}_3)$:
  - *mark* strands with bit 3 set to “0”
  - *destroy*
  - *unmark*
- The remaining strands satisfy the formula

Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

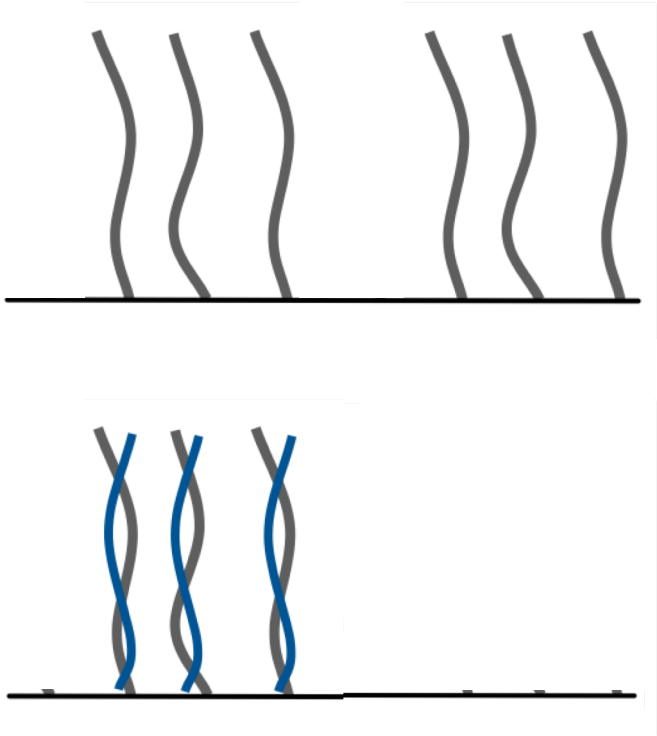
- Clause: $(\bar{x}_3)$:
  - *mark* strands with bit 3 set to "0"



Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

- Clause: $(\bar{x}_3)$:
  - *mark* strands with bit 3 set to “0”
  - *destroy*

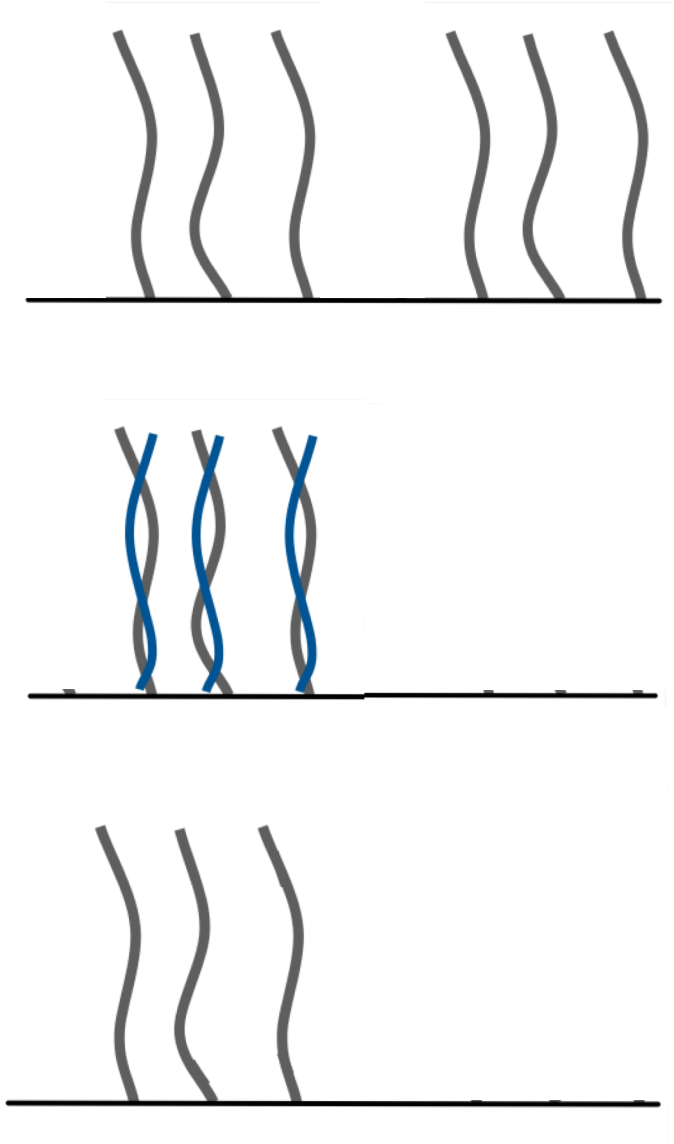


Surface-based DNA computing

Solving $(x_1 \vee x_2) \wedge (\bar{x}_3)$:

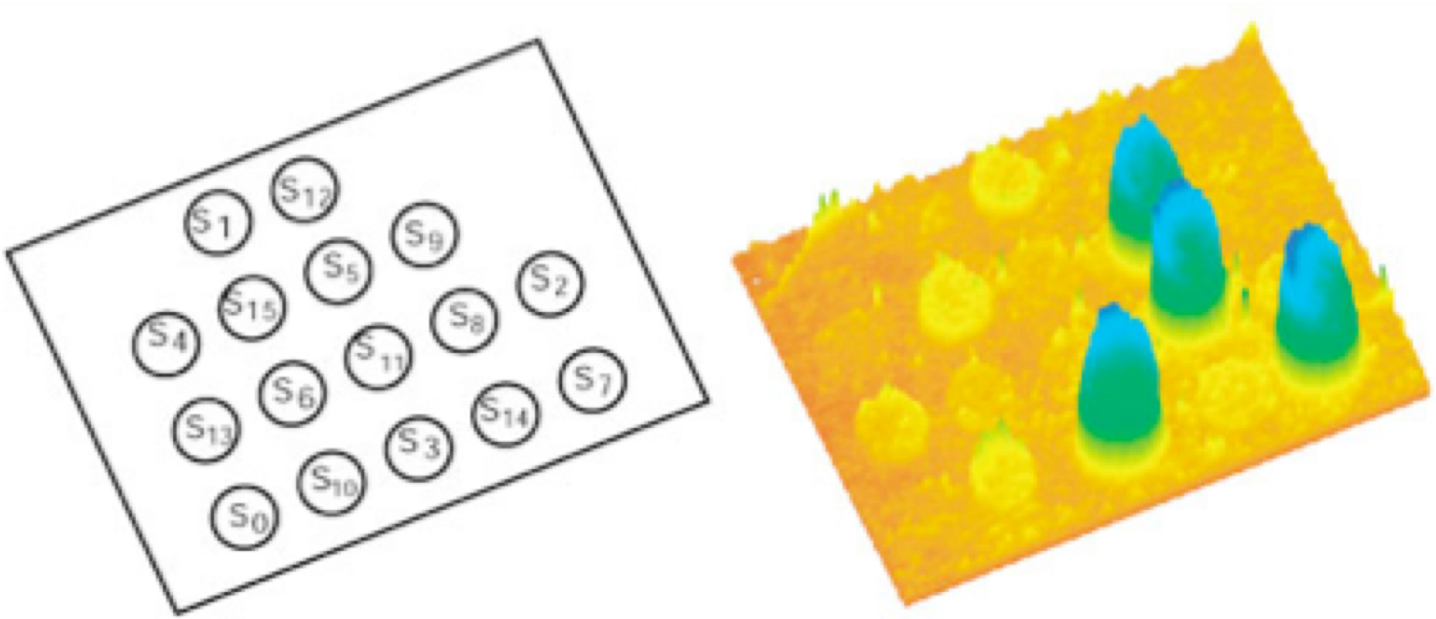
- Clause: $(\bar{x}_3)$:
  - *mark* strands with bit 3 set to "0"
  - *destroy*
  - *unmark*
- The remaining strands satisfy the formula:

  010, 100, 110



Surface-based DNA computing

- In our lab experiment, the logic formula was more complex, involving four variables and a “readout” step

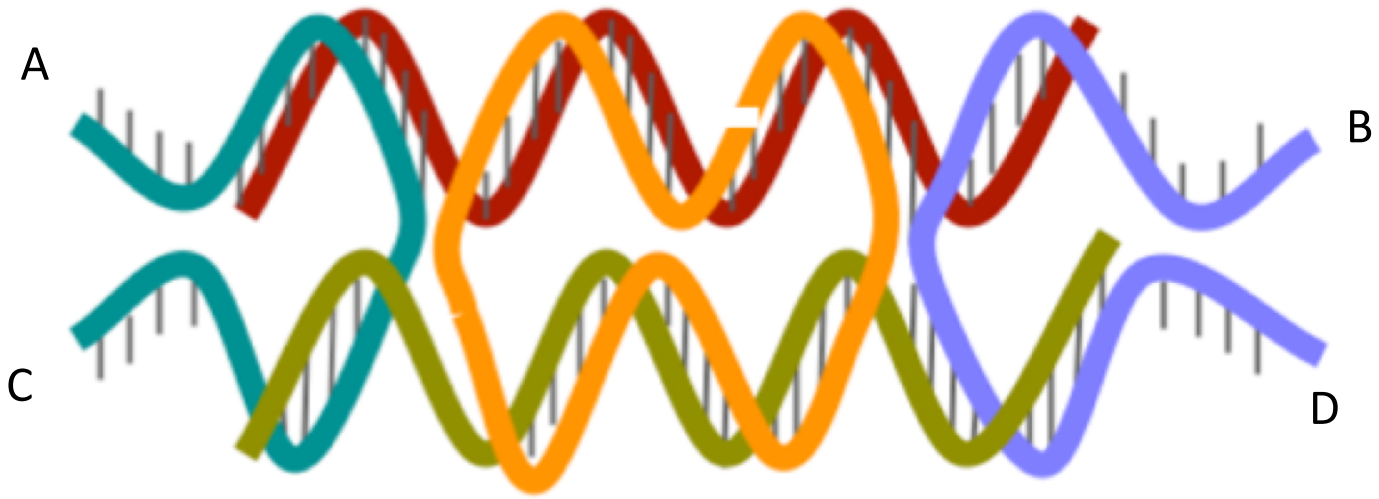


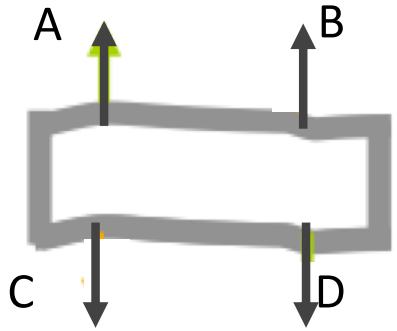
Surface-based DNA computing

- “What is not clear is whether such massive numbers of inexpensive operations can be productively used to solve real computational problems.” - Len Adleman
- "This is still a science-fiction kind of thing" - Lloyd Smith
- Still, this early work also paved the way for exploring many other creative ways to program with DNA

Structure-based DNA computing

- A double-crossover structure with four “sticky ends” (regions of unpaired bases) labeled A, B, C, D, and its tile abstraction





Fu and Seeman, Biochemistry, 1993

Structure-based DNA computing

- Tiles (double-crossover molecules) adhere to a growing assembly if glue strengths (sticky end lengths) are sufficiently strong

Fu and Seeman, Biochemistry, 1993

Structure-based DNA computing

- Tiles (double-crossover molecules) adhere to a growing assembly if glue strengths (sticky end lengths) are sufficiently strong

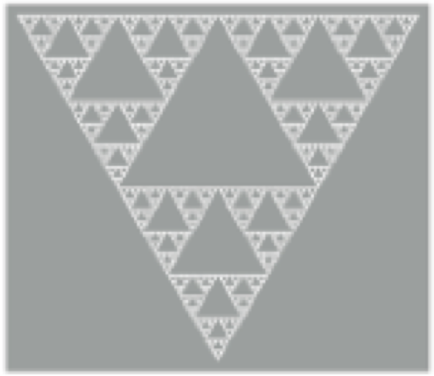
Fu and Seeman, Biochemistry, 1993

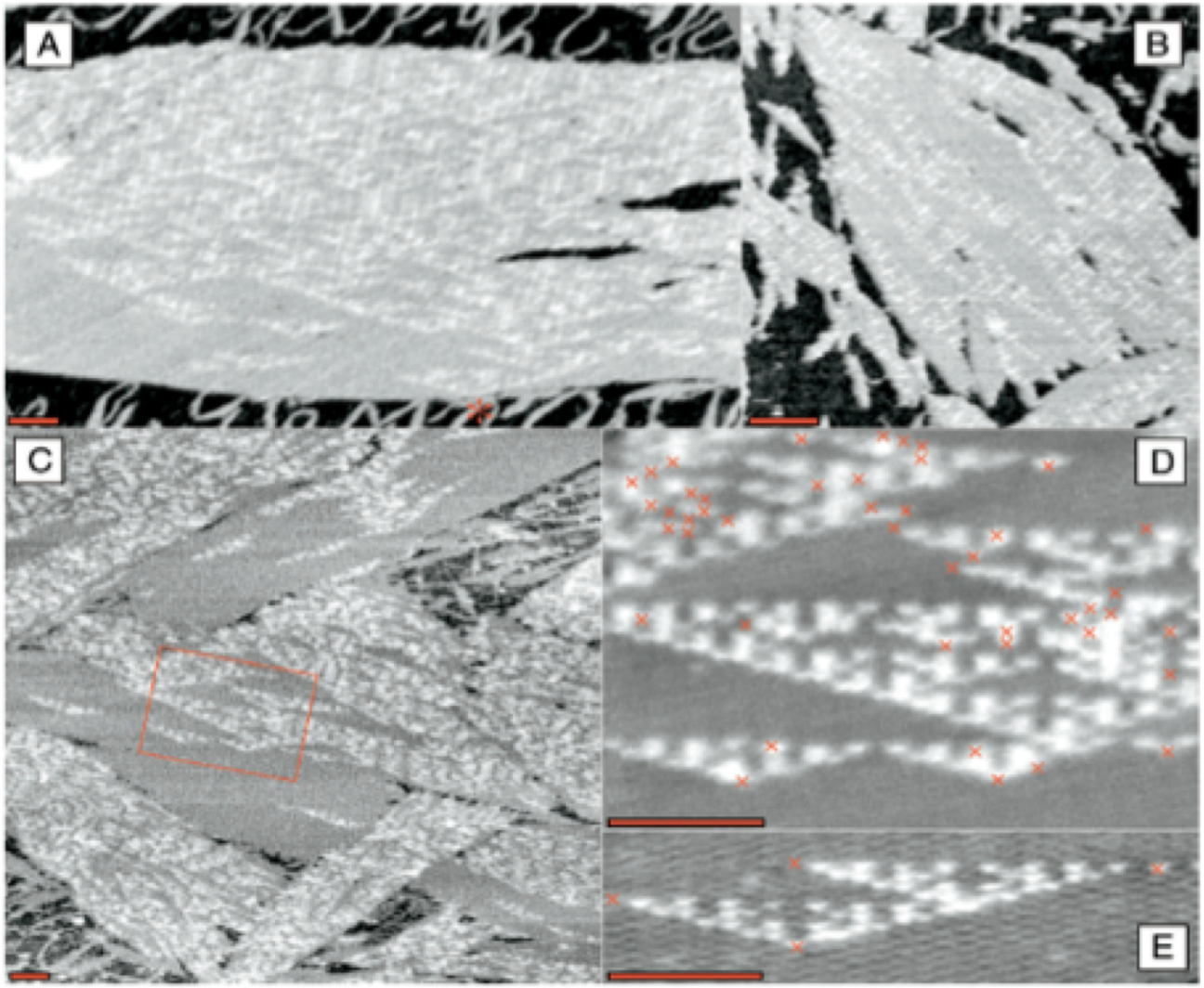
Structure-based DNA computing

- Tiles (double-crossover molecules) adhere to a growing assembly if glue strengths (sticky end lengths) are sufficiently strong

Fu and Seeman, Biochemistry, 1993

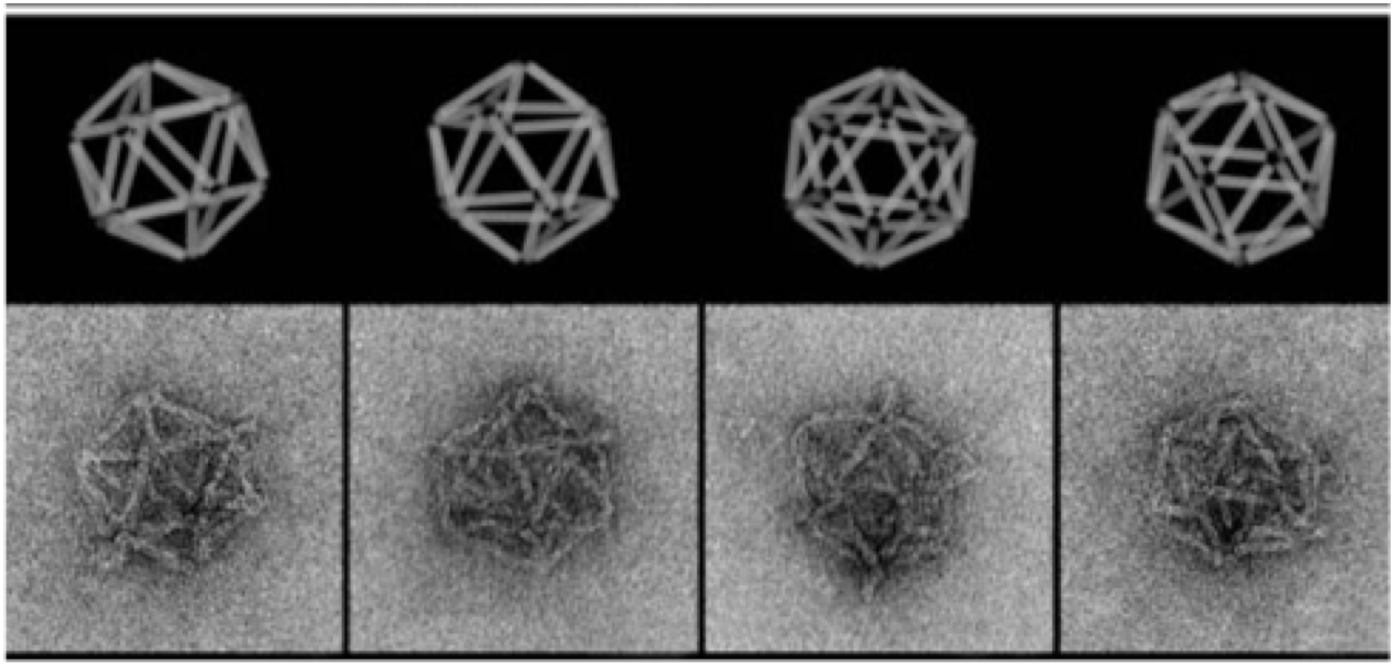
Structure-based DNA computing





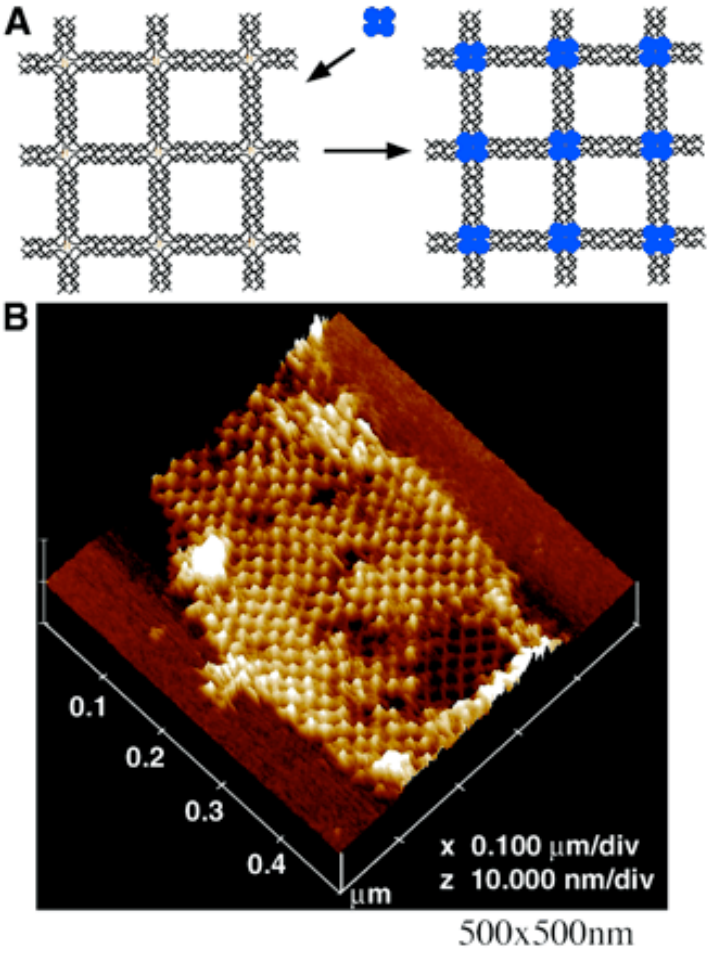
Winfree et al., Nature, 1998; Rothemund et al., Nature, 2004

Structure-based DNA computing



Douglas et al., Nature, 2000

Structure-based DNA computing



- DNA lattice structures have been used to arrange other molecules (proteins) on a surface, making it easier to study their structure
- Other potential applications include miniaturization of electronics circuits, or biosensors

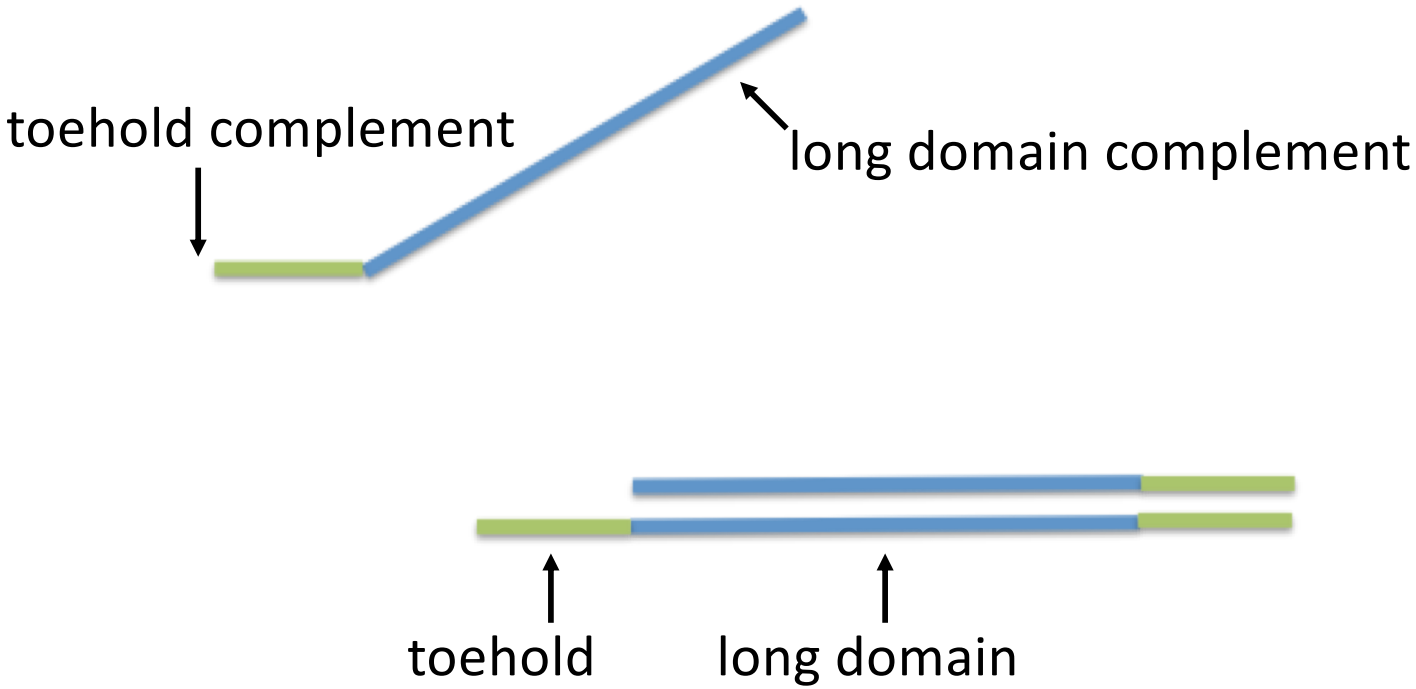
Yan et al, Science, 2003

Structure-based DNA computing

- These experiments show how one can “program” spatial arrangement of matter at the nanoscale
- They take advantage of DNA’s material properties to create artifacts, rather than just producing an answer
- Still, like earlier experiments on DNA sequence, the process results in a static outcome

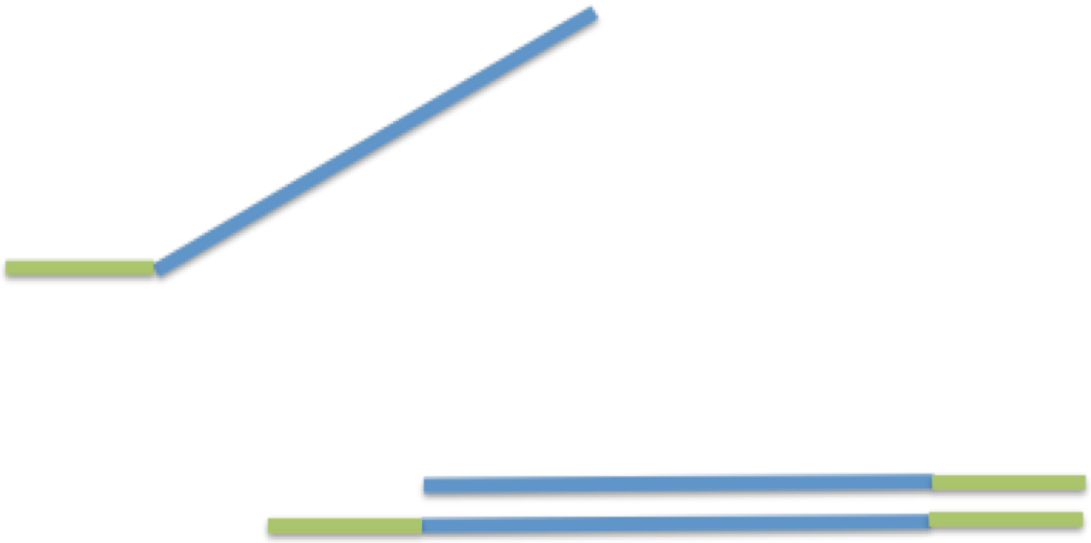
DNA strand displacement systems (DSDs)

DSDs | DNA strand displacements model

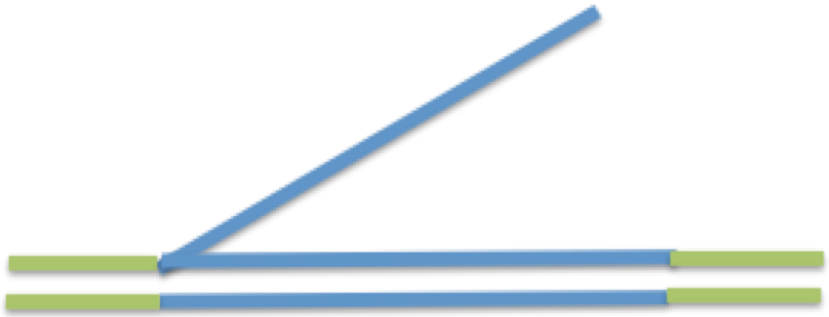


Soloveichik, Seelig, Winfree, PNAS 2010

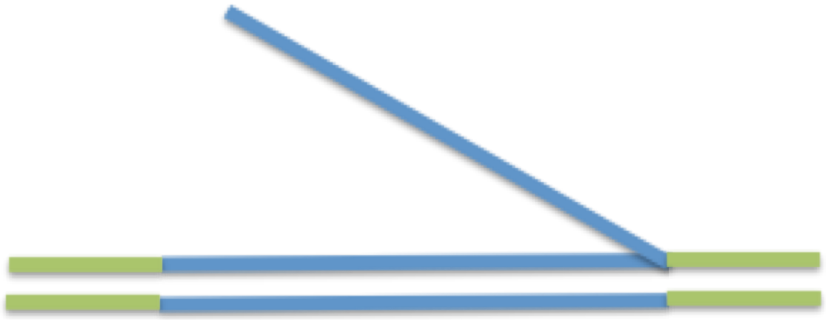
DSDs | DNA strand displacements model



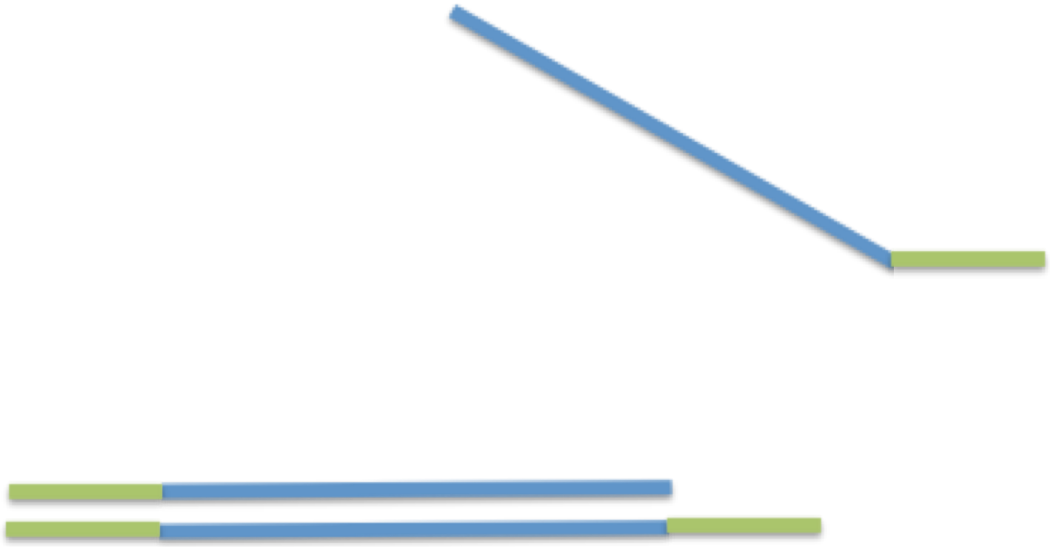
DSDs | DNA strand displacements model



DSDs | DNA strand displacements model



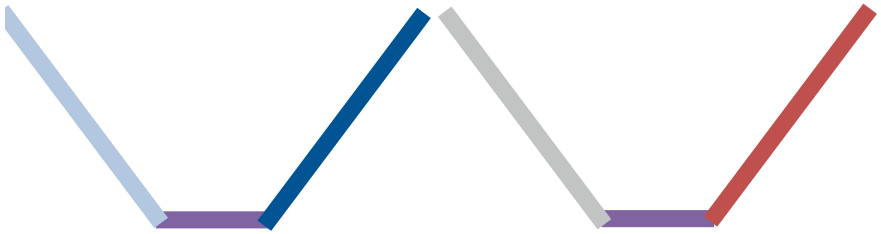
DSDs | DNA strand displacements model



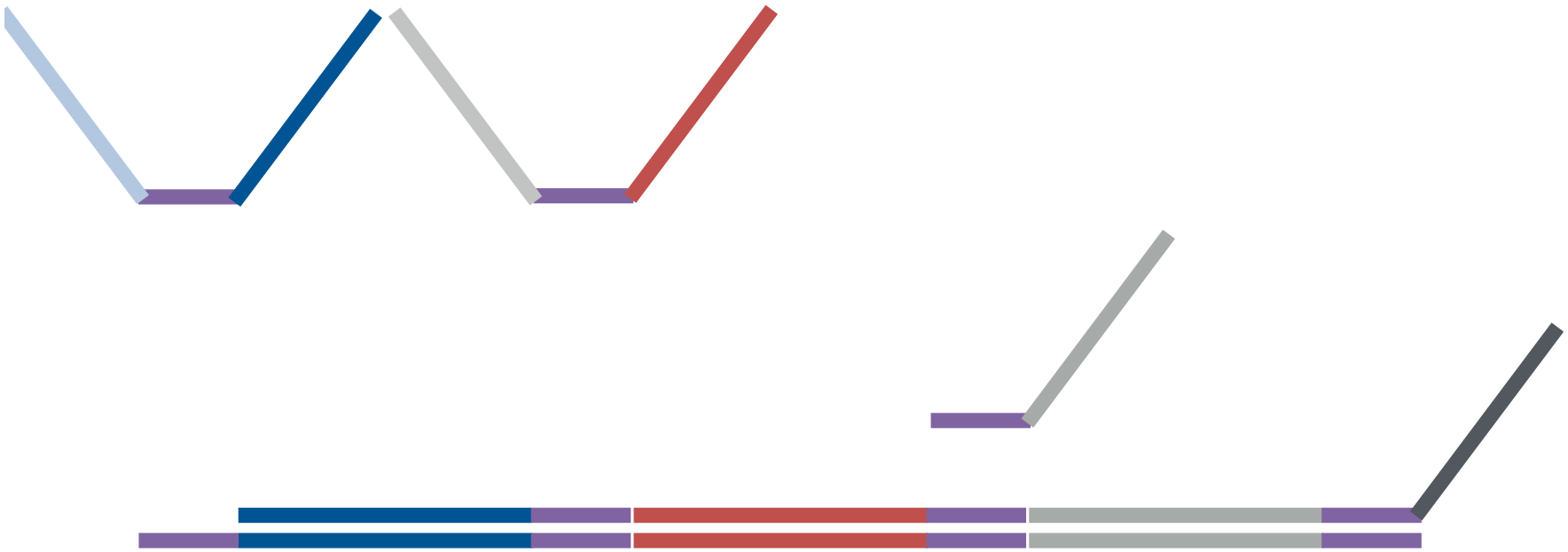
energy efficient
reversible or irreversible variants

DSDs | DNA strand displacements model

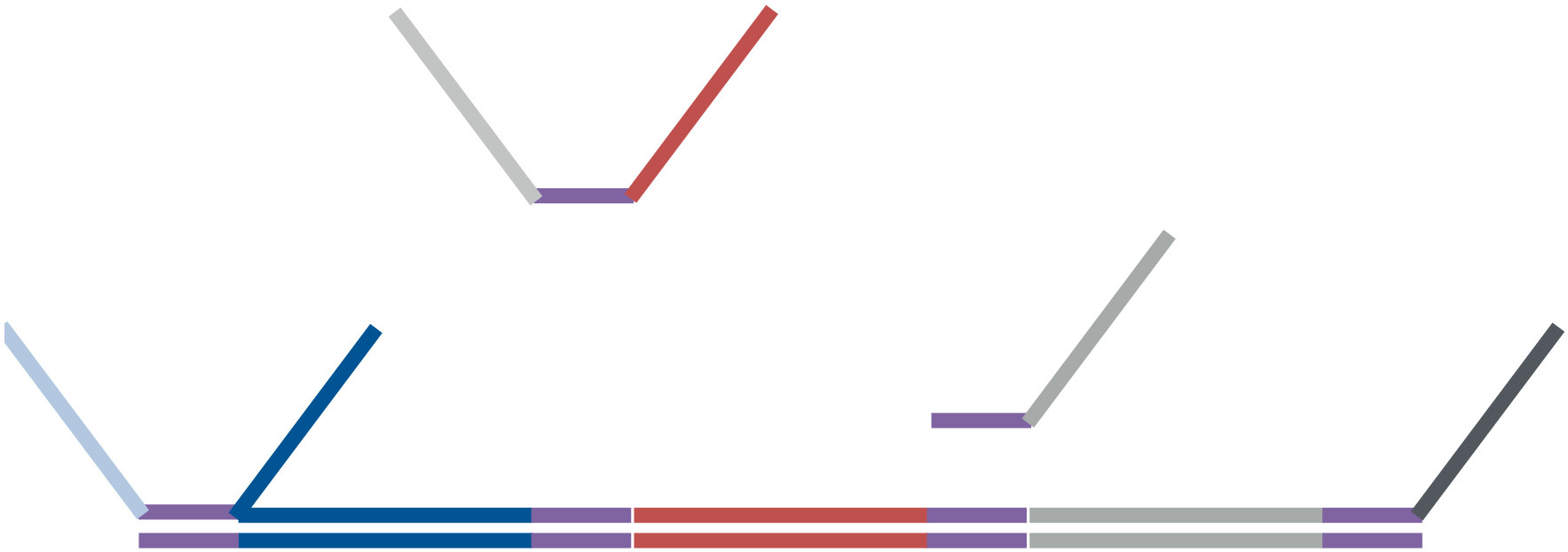
DSDs | DNA strand displacements model



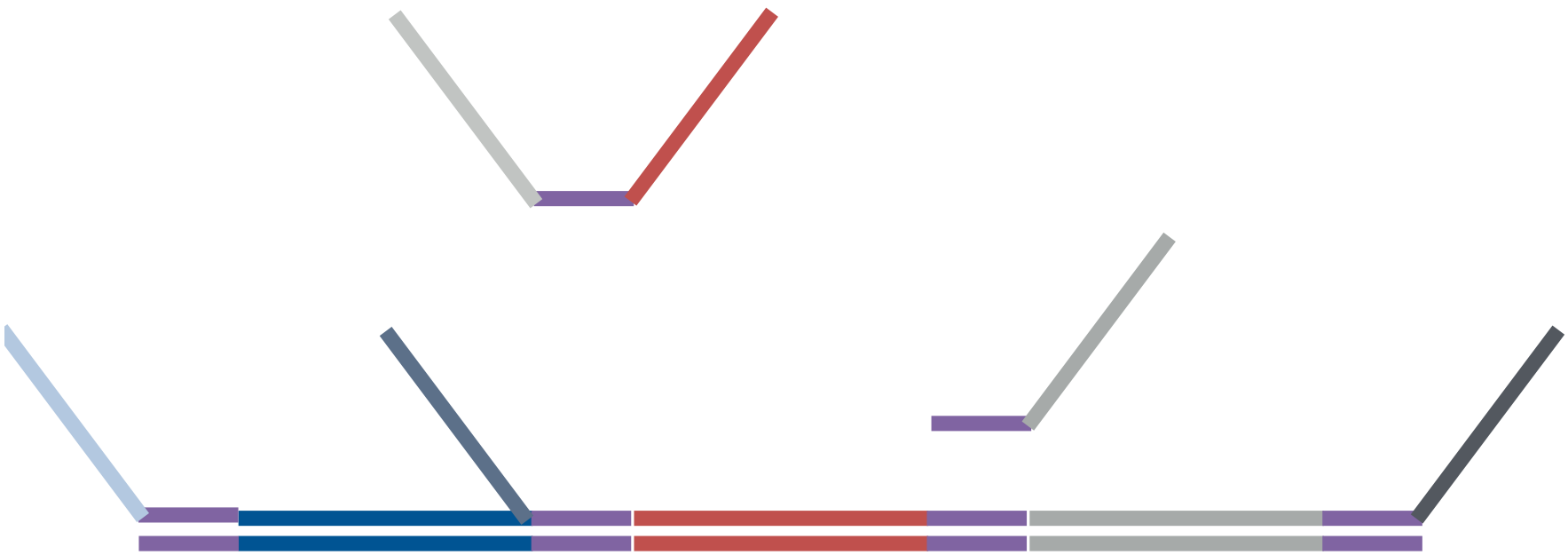
DSDs | DNA strand displacements model



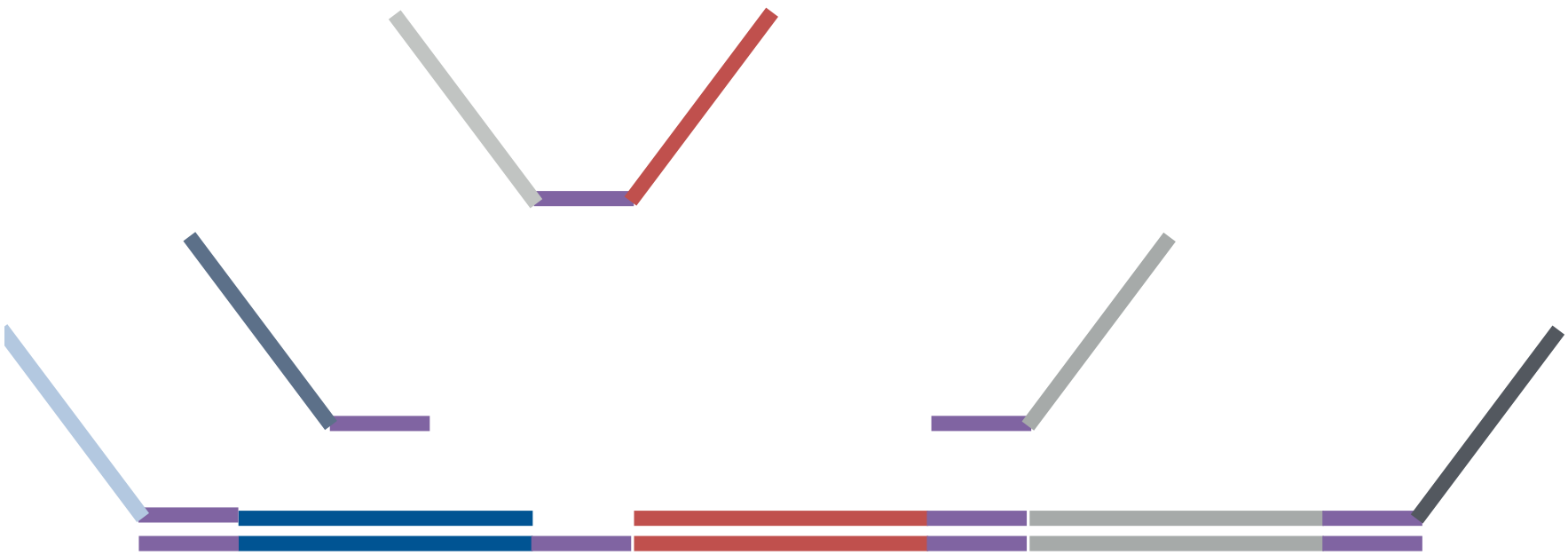
DSDs | DNA strand displacements model



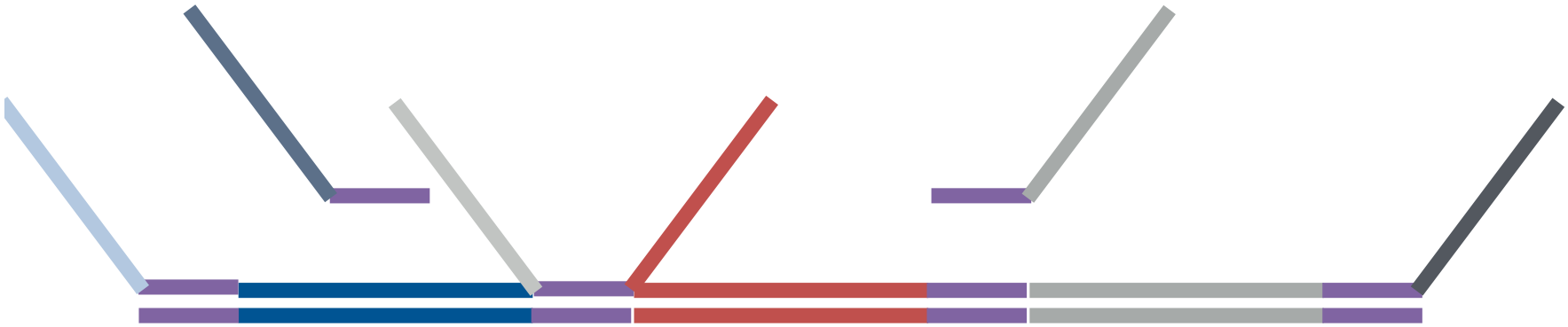
DSDs | DNA strand displacements model



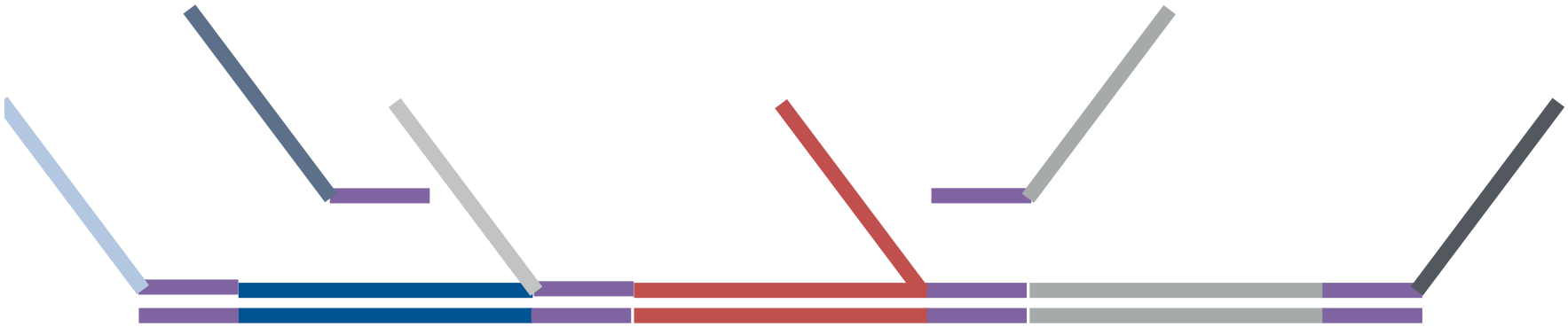
DSDs | DNA strand displacements model



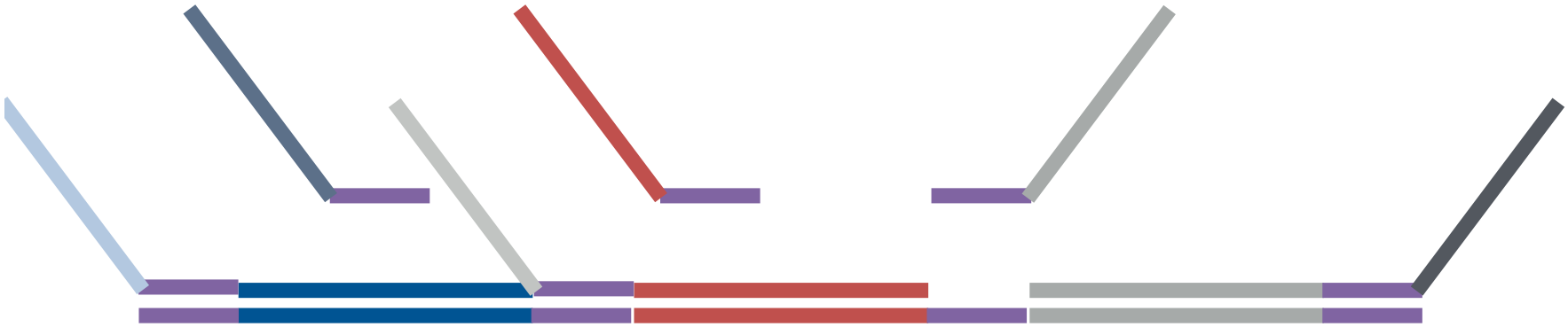
DSDs | DNA strand displacements model



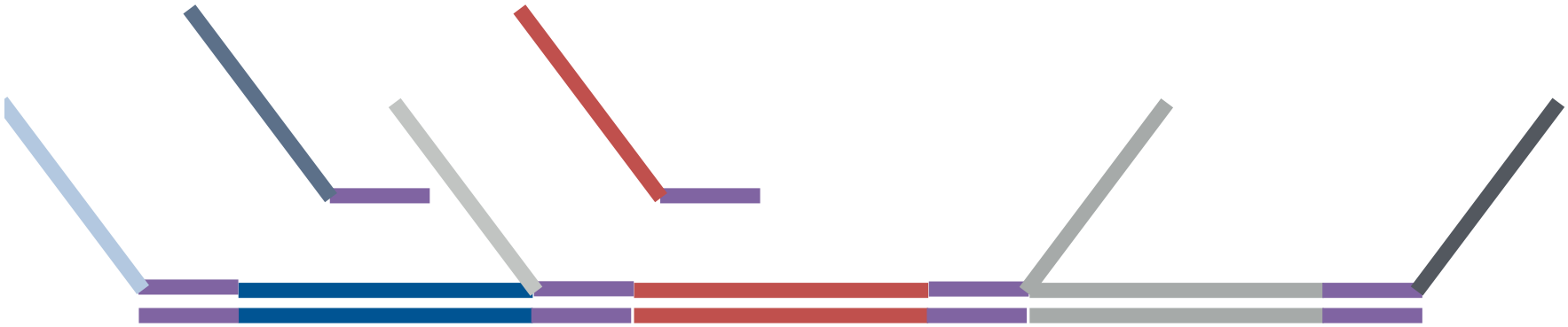
DSDs | DNA strand displacements model



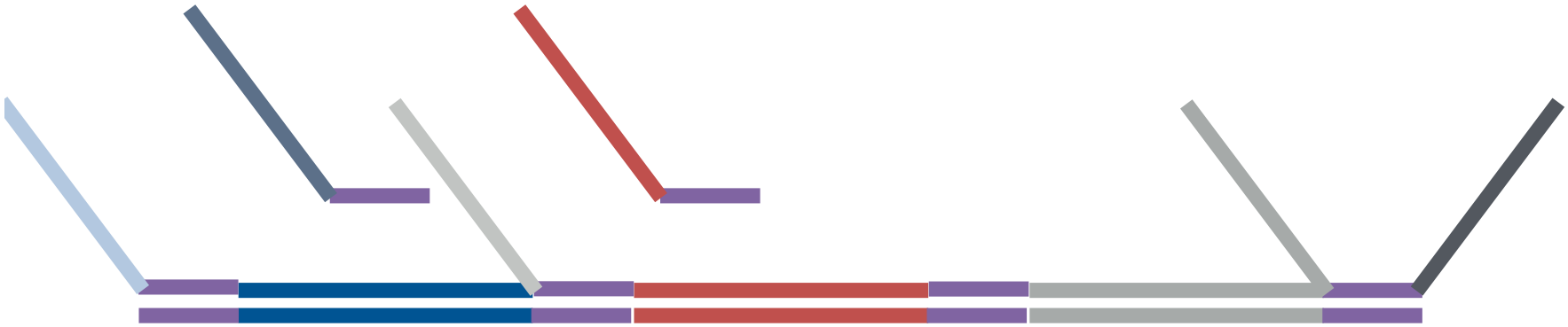
DSDs | DNA strand displacements model



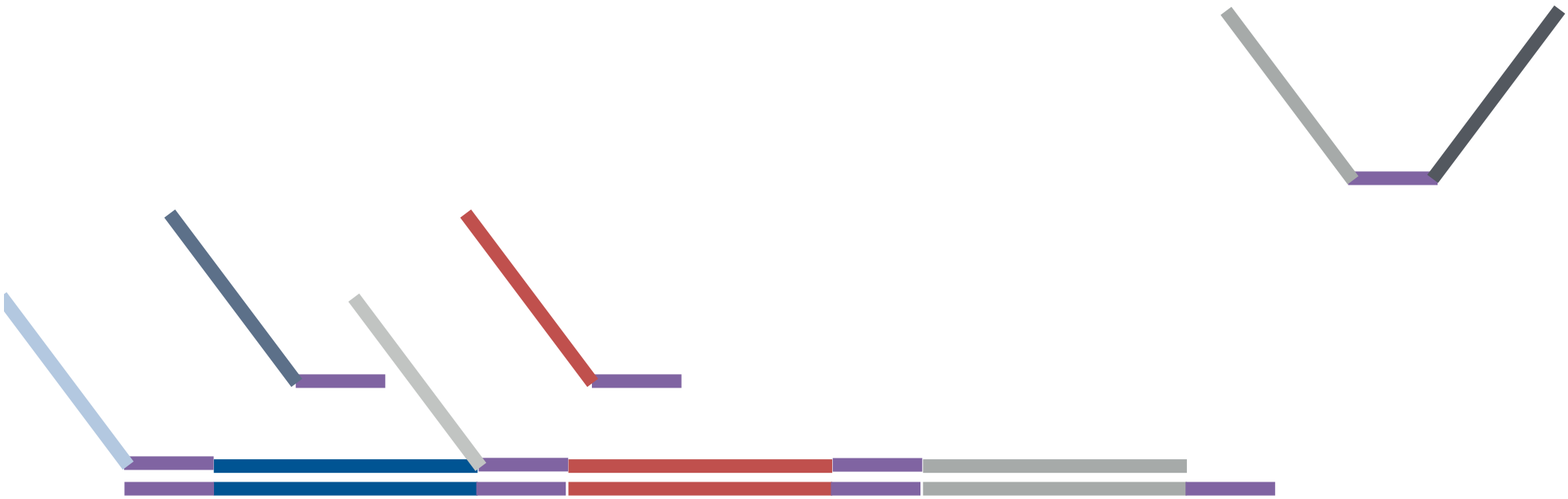
DSDs | DNA strand displacements model



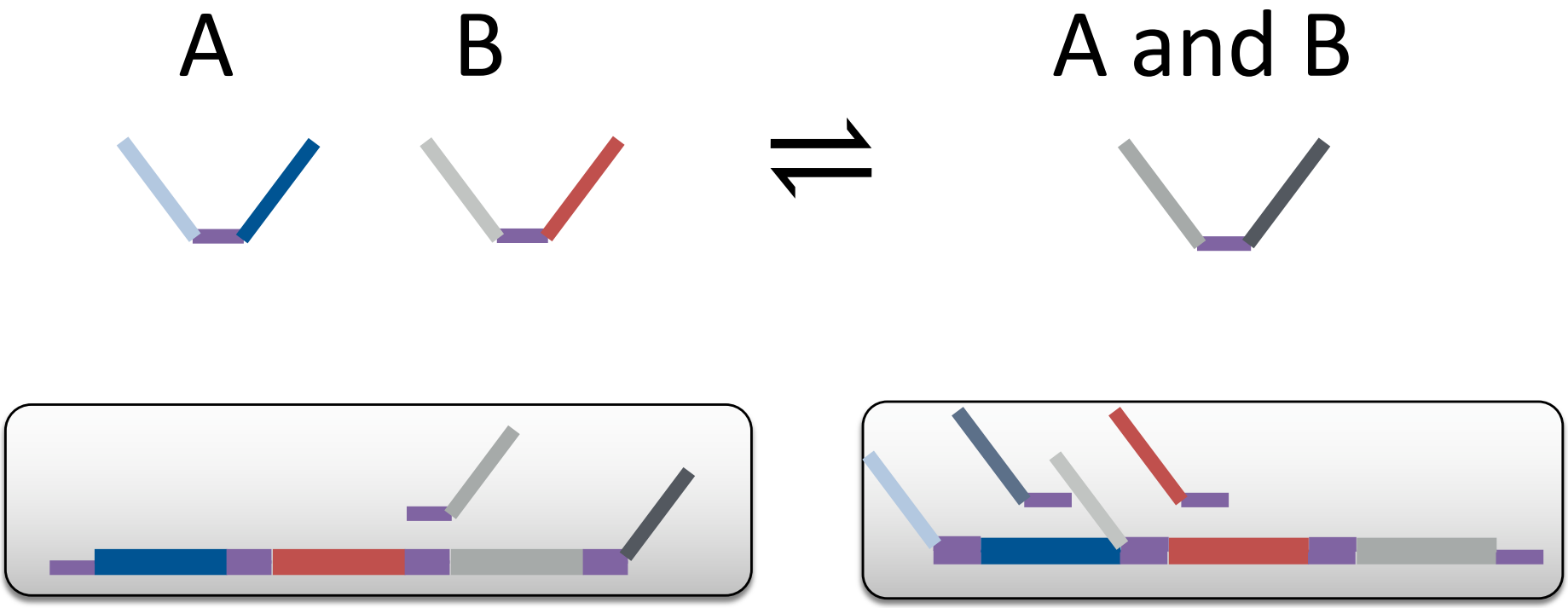
DSDs | DNA strand displacements model



DSDs | DNA strand displacements model



DSDs | DNA strand displacements model

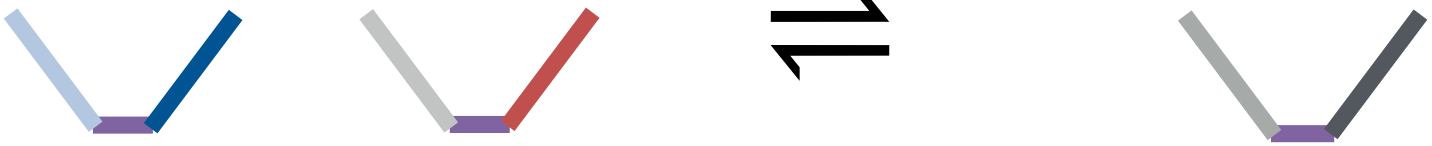


transformer molecules

DSDs | DNA strand displacements model

could you design transformer molecules for A or B?

A B ⇌ A or B?



DSDs | DNA strand displacements model

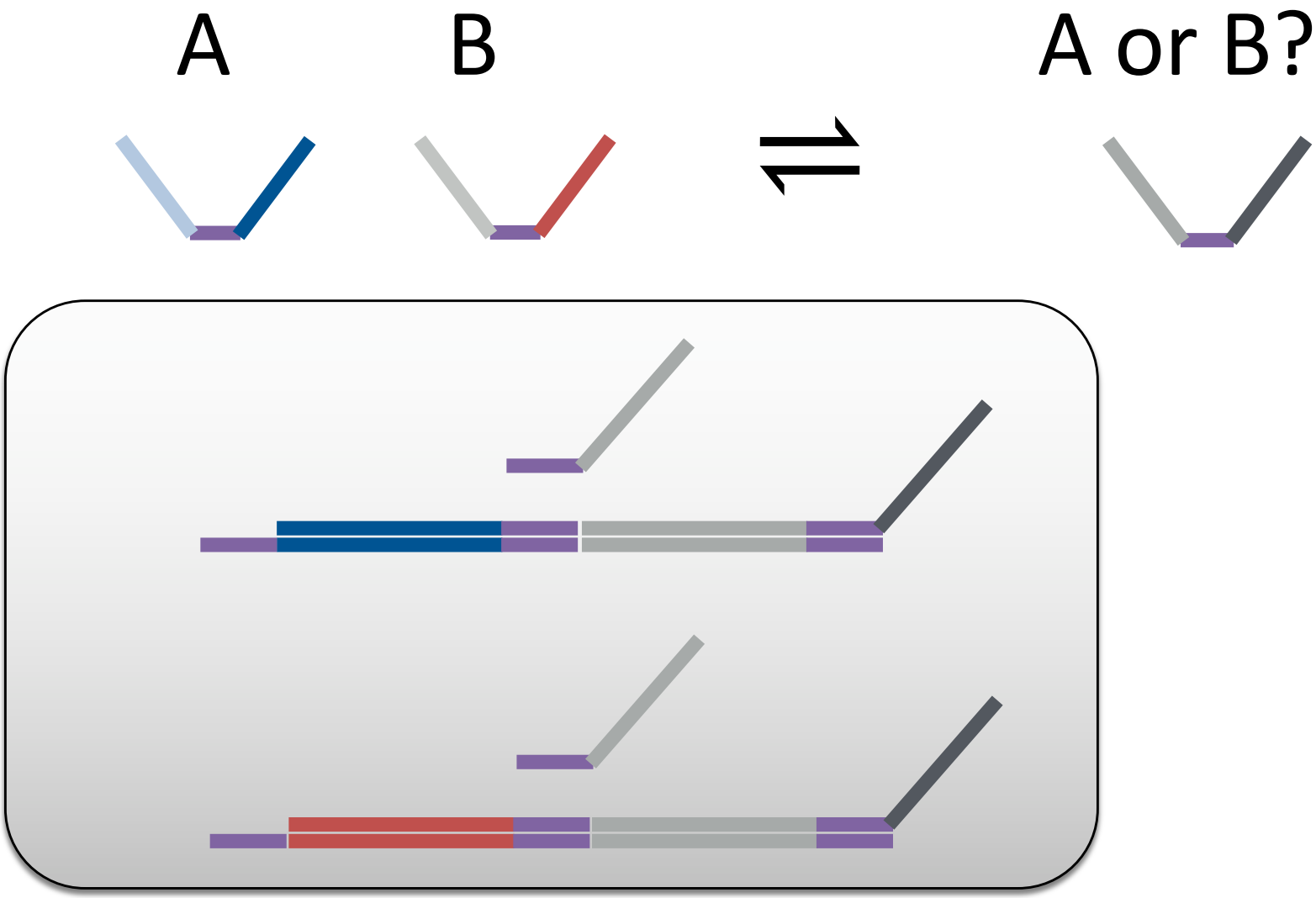
could you design transformer molecules for A or B?

A B ⇌ A or B?

transformer molecules?

DSDs | DNA strand displacements model

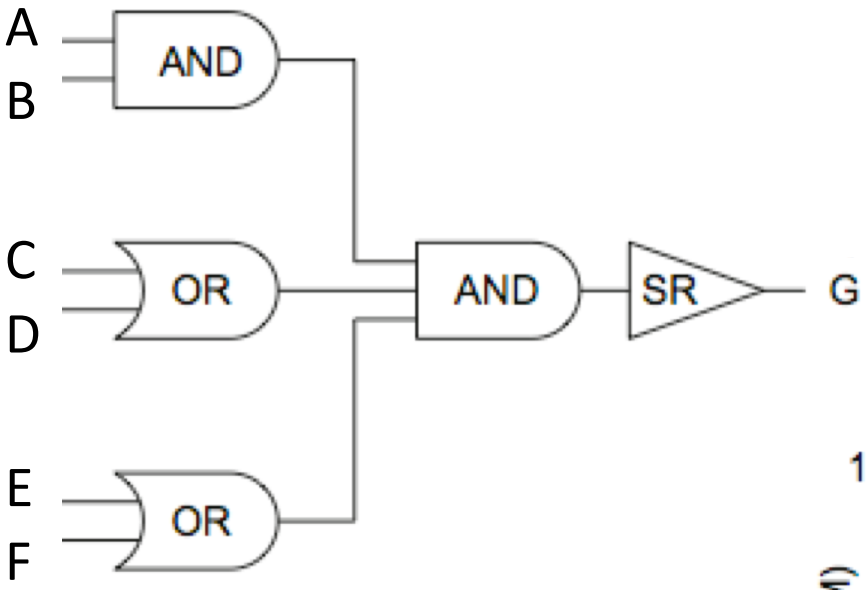
could you design transformer molecules for A or B?

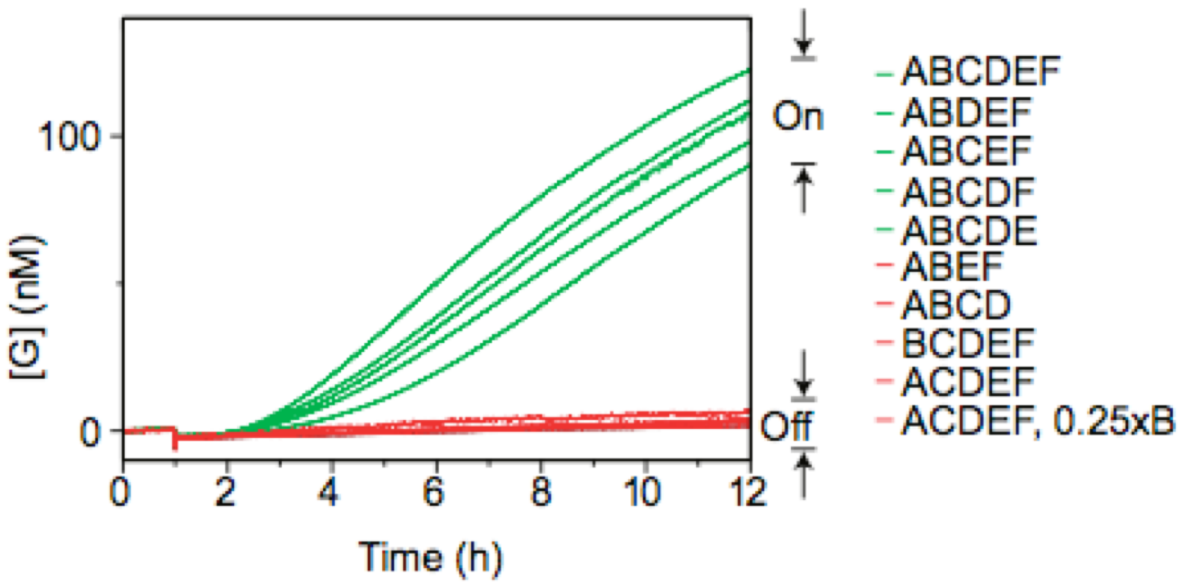


transformer molecules?

DSDs | DNA strand displacements model

experimental demonstration

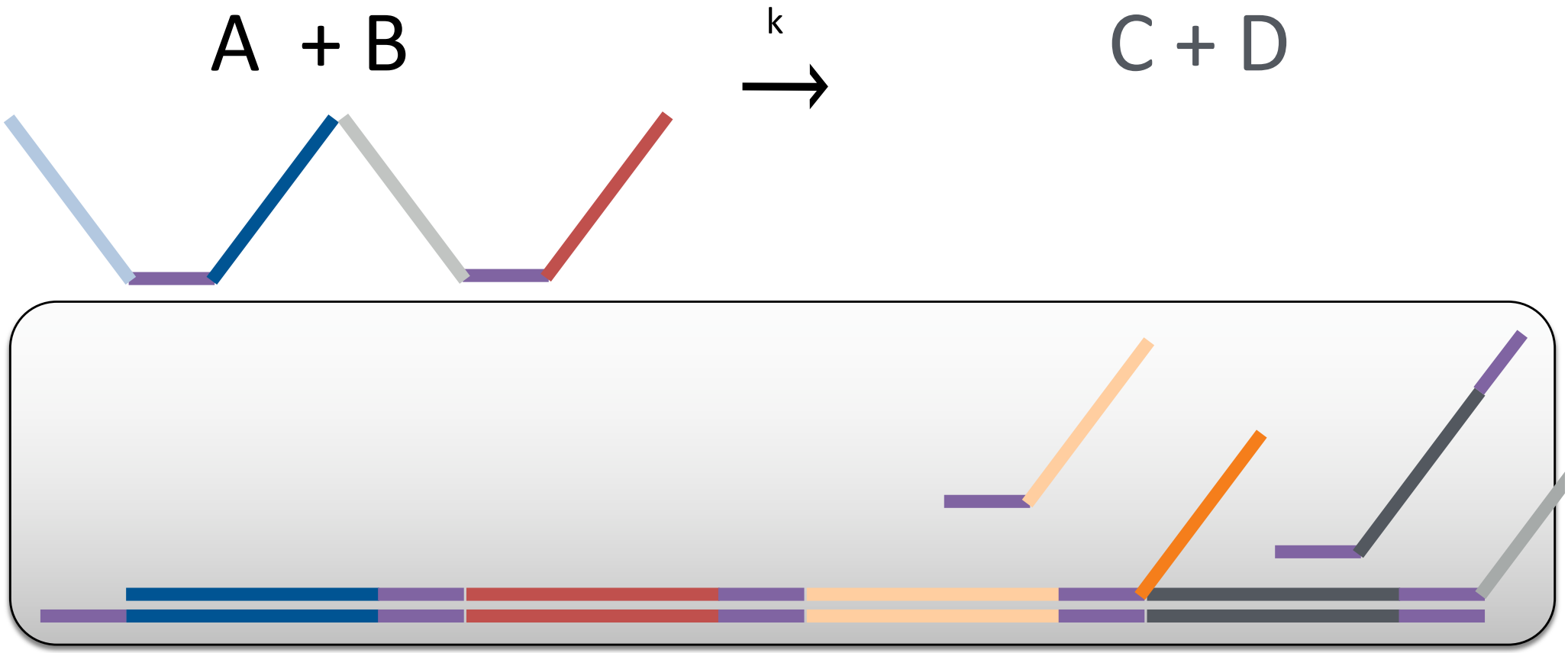




Zhang and Seelig, Nature Chemistry, 2011

DSDs | DNA strand displacements model

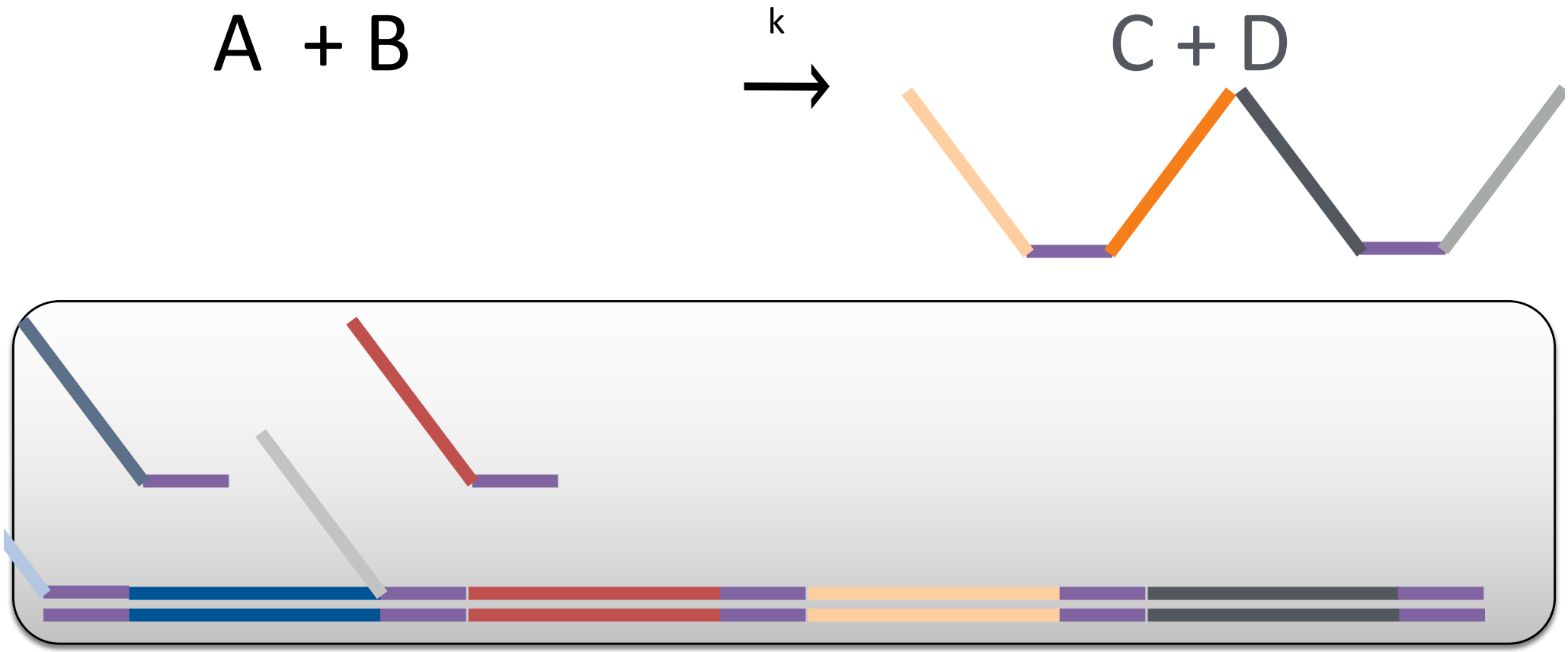
strand displacement implementation of a chemical reaction



Soloveichik, Seelig, Winfree PNAS 2010

DSDs | DNA strand displacements model

strand displacement implementation of a chemical reaction



Soloveichik, Seelig, Winfree PNAS 2010

DSDs | DNA strand displacements model

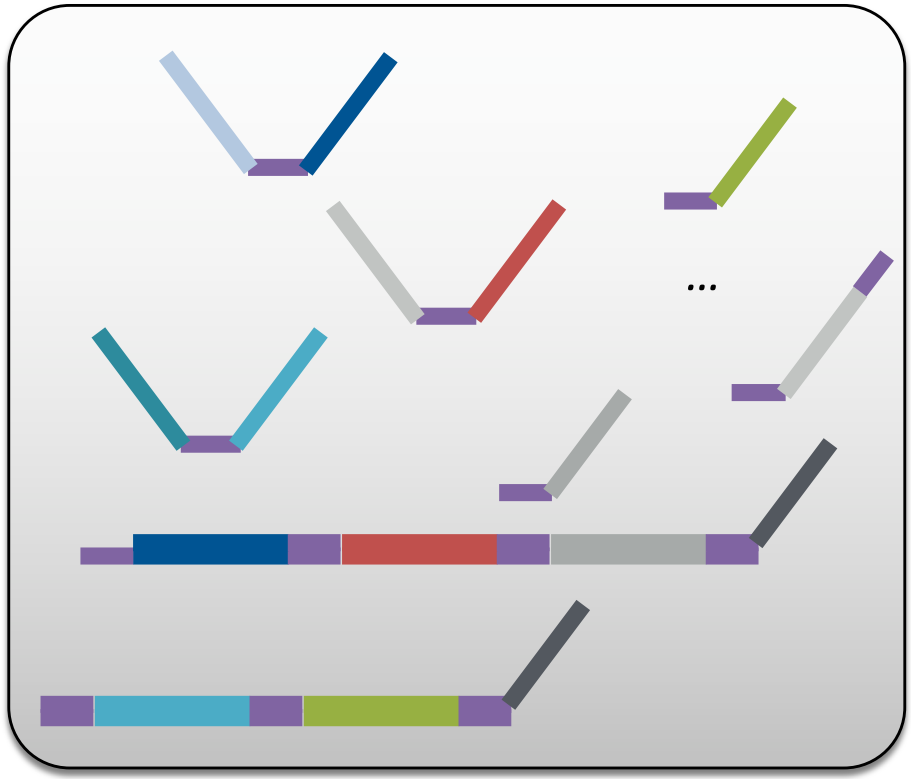
DSDs can implement Chemical Reaction Networks (CRNs)

$$A + B \overset{k_1}{\rightleftharpoons} C$$

$$C + D + E \overset{k_2}{\rightarrow} F + B$$

$$D + F \overset{k_3}{\rightarrow} D + A$$

...



Chen, Dalchau, Srinivas, Phillips, Cardelli, Soloveichik, Seelig

CRNs: A molecular programming model

- CRNs : Chemical Reaction Networks

CRNs: A molecular programming model

- CRN to compute $f(n_1,n_2) = n_1 + n_2$

$$X_1 \rightarrow Y$$
$$X_2 \rightarrow Y$$

(Ignore reaction rate constants for now)

CRNs: A molecular programming model

- We'll consider a model in which data is stored in the integer counts of molecules in a well-mixed solution, and reactions are operations
- Inputs are represented by counts n_i of molecular species X_i, and outputs are represented by counts of species Y_i

CRN examples and model justification

- f(n) = 2n: $X \rightarrow 2Y$

CRN examples and model justification

- f(n) = 2n: $X \rightarrow 2Y$

- What about conservation of mass?

  Molecules that preserve mass need not be described explicitly (but participate in reactions nevertheless)

  (E.g., assume a large supply of a neutral molecule W and replace the above reaction with W + X → 2Y)

- Assume that volume scales over time, in proportion with the total count of molecular species

CRN examples and model justification

- f(n) = n/3:

CRN examples and model justification

- f(n) = n/3: $3X \rightarrow Y$

CRN examples and model justification

- f(n) = n/3: $3X \rightarrow Y$

- Aren't reactions with three or more reactants unrealistic?

  Higher order reactions can rewritten as bimolecular reactions given additional *context*, say a copy of L_0:

$$
\begin{aligned}
L_0 + X &\rightarrow L_1 \\
L_1 + X &\rightarrow L_2 \\
L_2 + X &\rightarrow L_0 + Y
\end{aligned}
$$

CRN examples and model justification

- f(n) = n/3: $3X \rightarrow Y$

- Can this be done with bimolecular reactions, *without* additional context?

CRN examples and model justification

- f(n) = n/3: $3X \rightarrow Y$

- Can this be done with bimolecular reactions, *without* additional context?

X + X → X'
X' + X’ → Y + X
X' + X → Y

CRNs: More examples

- $\min(n_1,n_2)$
- $n_1 - n_2$ (assume that $n_1 \geq n_2$)
- $\max(n_1,n_2)$

CRNs: More examples

- min(n_1,n_2) $X_1 + X_2 \rightarrow Y$
- $n_1 - n_2$ (assume that $n_1 \geq n_2$)
- max(n_1,n_2)

CRNs: More examples

- min(n_1,n_2) $X_1 + X_2 \rightarrow Y$
- $n_1 - n_2$ (assume that $n_1 \geq n_2$)

$$X_1 \rightarrow Y$$
$$X_2 + Y \rightarrow \emptyset$$

- max(n_1,n_2)

CRNs: More examples

- min(n_1,n_2) $X_1 + X_2 \rightarrow Y$
- $n_1 - n_2$ (assume that $n_1 \geq n_2$)

$$\begin{aligned} X_1 &\rightarrow Y \\ X_2 + Y &\rightarrow \emptyset \end{aligned}$$

- max(n_1,n_2)

$$\begin{aligned} X_1 &\rightarrow Y + Z_1 \\ X_2 &\rightarrow Y + Z_2 \\ Z_1 + Z_2 &\rightarrow K \\ K + Y &\rightarrow \emptyset \end{aligned}$$

CRNs: Formal model

- A CRN is a pair ($\mathbf{\Lambda}$,R) where

  $\mathbf{\Lambda}$ is an ordered set of species

  R is a set of reactions (r,p,k), where

  – r and p are vectors of length $|\mathbf{\Lambda}|$ that describe the reactants and products respectively

  – k is the rate constant (omitted if 1)

CRNs: Formal model

- A CRN is a pair ($\mathbf{\Lambda}$,R) where

  $\mathbf{\Lambda}$ is an ordered set of species

  R is a set of reactions (r,p,k), where

  – r and p are vectors of length $|\mathbf{\Lambda}|$ that describe the reactants and products respectively

  – k is the rate constant (omitted if 1)

- *Example*: ({X1, X2, Y}, {((1,1,0),(0,0,1))})

CRNs: Formal model

- A CRN is a pair ($\mathbf{\Lambda}$,R) where

  $\mathbf{\Lambda}$ is an ordered set of species

  R is a set of reactions (r,p,k), where

  – r and p are vectors of length $|\mathbf{\Lambda}|$ that describe the reactants and products respectively

  – k is the rate constant (omitted if 1)

- *Example*: ({X1, X2, Y}, {((1,1,0),(0,0,1))})

  (denotes X1 + X2 → Y)

CRNs: Formal model

- A *configuration* c is a vector of non-negative integers of length $|\Lambda|$ where c(X) denotes the count of species X
- A reaction (r,p) is *applicable* to c if r ≤ c and the result of the reaction is c – r + p
- *Example*: reaction X1 + X2 → Y
  - is applicable to configuration c = (2,3,0)
  - is not applicable to configuration c' = (3,0,1)

CRNs: Formal model

- A *configuration* c is a vector of non-negative integers of length $|\Lambda|$ where c(X) denotes the count of species X
- A reaction (r,p) is *applicable* to c if $r \leq c$ and the result of the reaction is $c - r + p$
- An *execution* is a sequence of configurations, such that for each consecutive pair c, c', some reaction applicable to c results in c'
- If (c1, c2, …, ck) is an execution sequence then we say that ck *is reachable from* c1 (c1 → ck)

CRNs: Function computation

- Let C be a CRN with *input species* X_1, X_2, …, X_k, *output species* Y (and possibly other species)
- A *valid initial configuration* C_{init} is one in which the counts of all but the input species is 0
- We'll denote $C_{init}(X_i)$ by n_i (initial count of input species X_i), and let $n = n_1 + n_2 + \ldots + n_k$
- A *configuration* o is *output stable* if for all c such that o → c, o(Y) = c(Y)

CRNs: Function computation

- Let f: $\mathbb{N}^k \rightarrow \mathbb{N}$

- We say that *C computes f* if for all valid initial configurations C_{init} and configurations c, if $C_{init} \rightarrow c$ then $c \rightarrow o$ where o is output stable and $o(Y) = f(n_1, n_2, \ldots, n_k)$.

Summary

- Formal model of stable function computation with chemical reaction networks
- Next time: define stochastic behaviour of CRNs, and time complexity