Tutorial on Chemical Reaction Networks

Part II

DISC'14

David Soloveichik

Outline

Distributed Algorithms in Biological Regulatory Networks

Molecular Implementation of CRNs with Strand Displacement Cascades

David Soloveichik

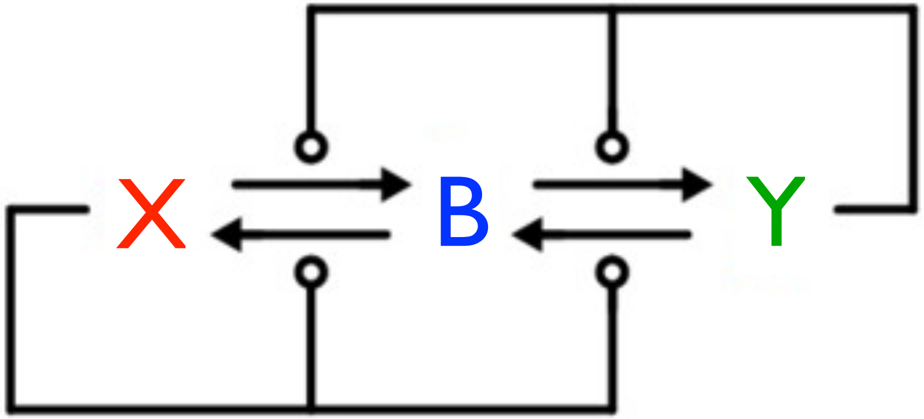
(3 Species) Approximate Majority

$X + Y \rightarrow X + B$

$X + Y \rightarrow B + Y$

$X + B \rightarrow X + X$

$Y + B \rightarrow Y + Y$



n = total number of molecules (X, Y, B)

- Fast/efficient: O(n log n) interactions to converge (optimal)
- Robust: above a threshold, the initial majority wins whp; even with some "byzantine agents"

[Angluin, Aspnes, Eisenstat DISC'07]

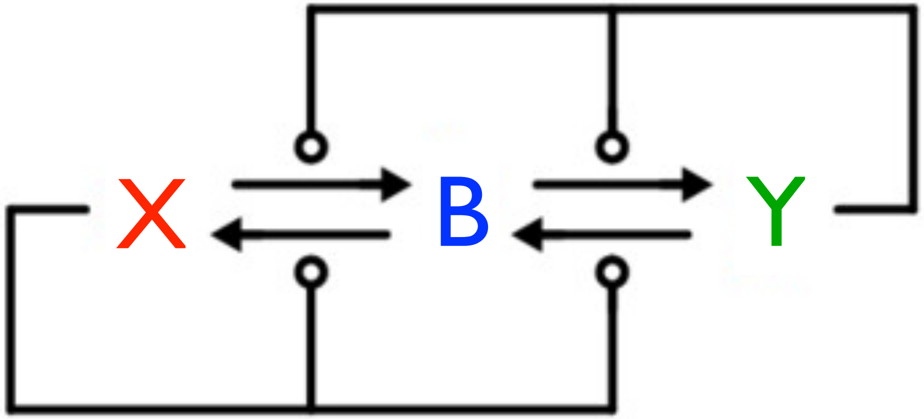
(3 Species) Approximate Majority

$X + Y \rightarrow X + B$

$X + Y \rightarrow B + Y$

$X + B \rightarrow X + X$

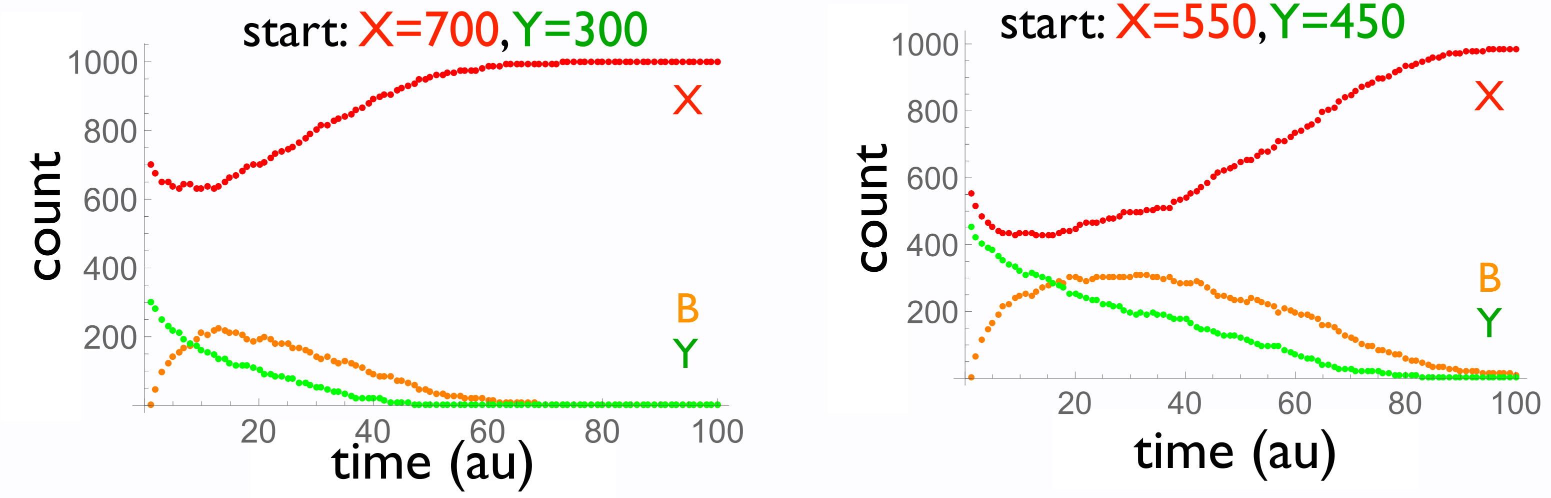
$Y + B \rightarrow Y + Y$



n =

- Fa mal)
- R ;
  ev

Example simulations:

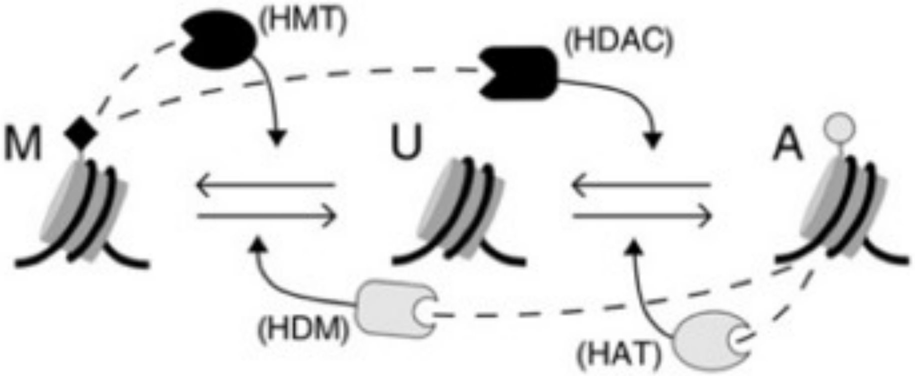


[Angluin, Aspnes, Eisenstat DISC'07]

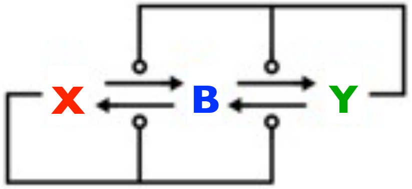
Example: Approximate Majority in a Biological Regulatory Network

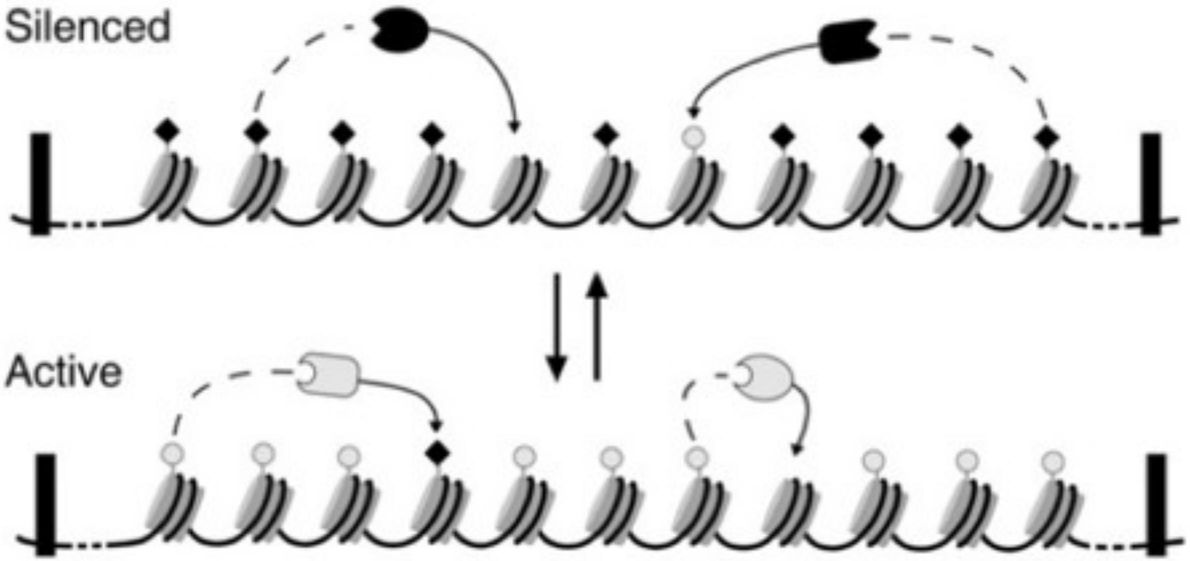
"Epigenetic Memory by Nucleosome Modification"

methylated unmodified acetylated



=

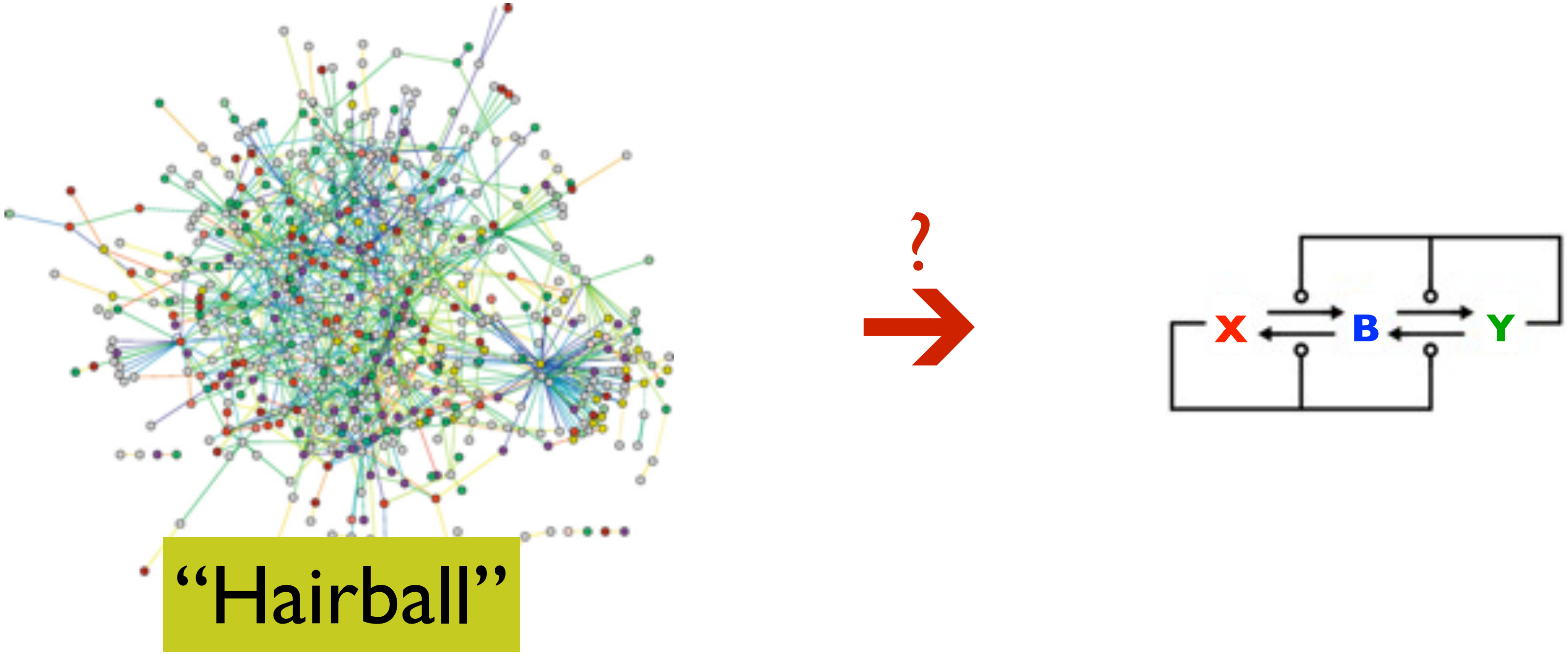




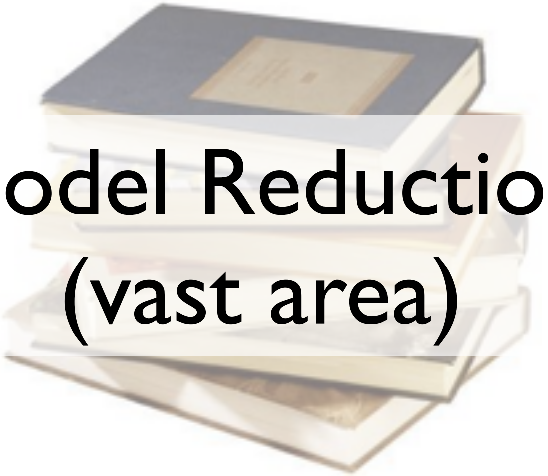
Dodd, Micheelsen, Sneppen, Thon, *Cell* 129, 813-822 (2007)

How Can We Identify CRN Algorithms in Biology?

Does a biologically messy network X “implement” some ideal algorithm Y?



How Can We Identify CRN Algorithms in Biology?

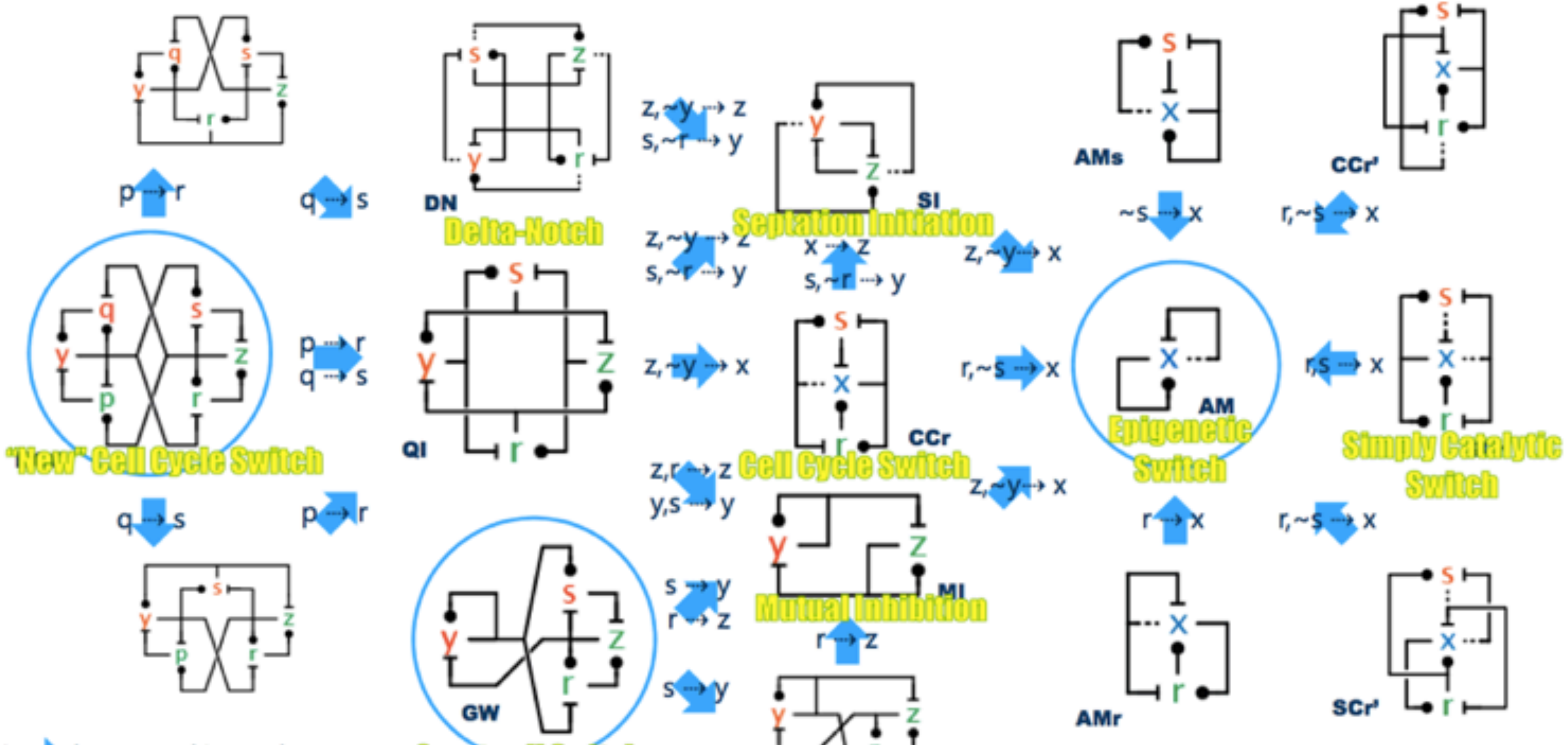


Intermediary Species → Model Reduction (vast area)

Symmetries → CRN Morphisms

[Cardelli, "Morphisms of reaction networks that couple structure to function" 2014]

Approximate Majority Emulation Zoo



(→ homomorphism and stoichiomorphism (transitive))

Slide credit: Luca Cardelli

Outline

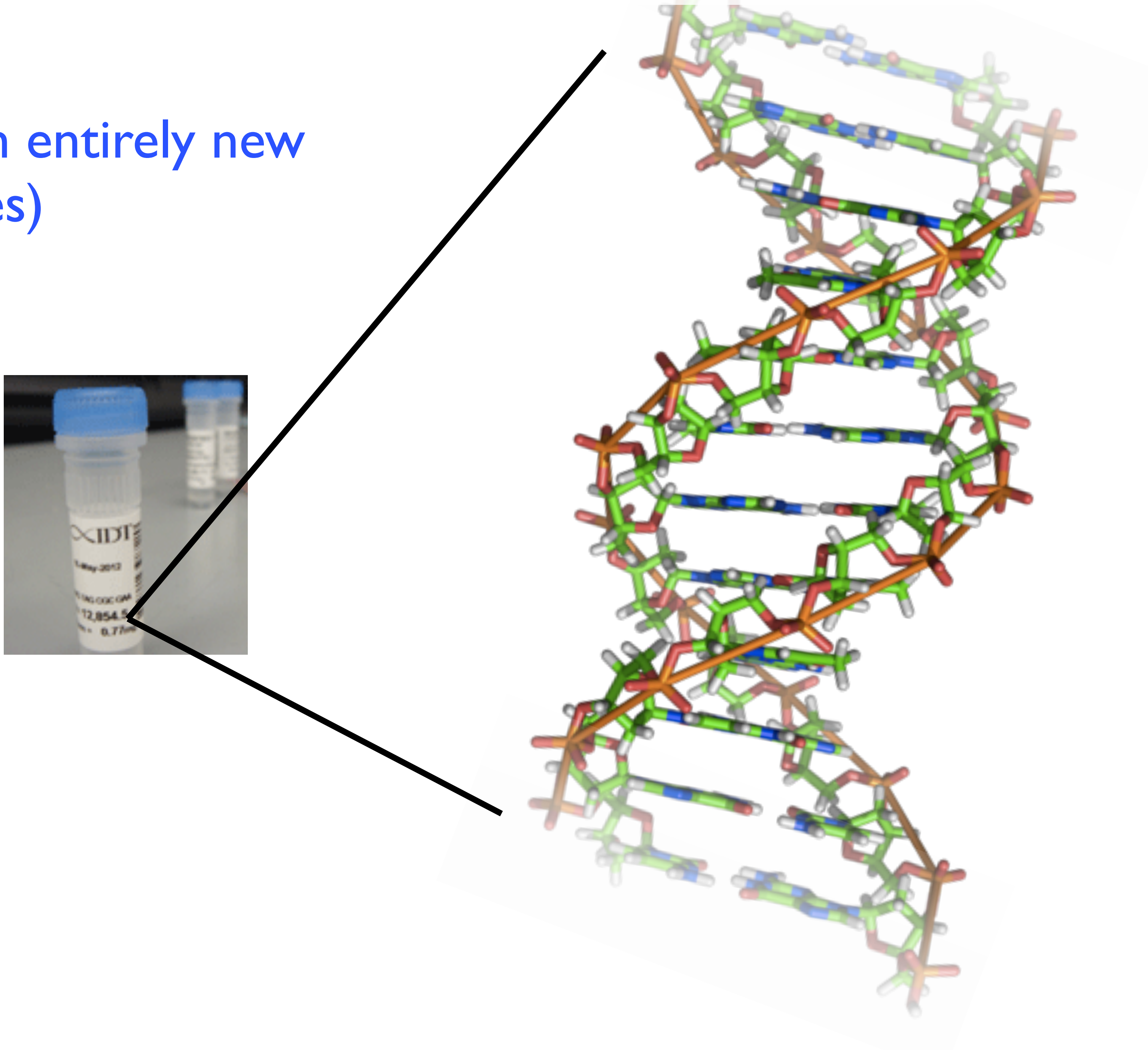
Distributed Algorithms in Biological Regulatory Networks

Molecular Implementation of CRNs with Strand Displacement Cascades

David Soloveichik

Strand Displacement Cascades

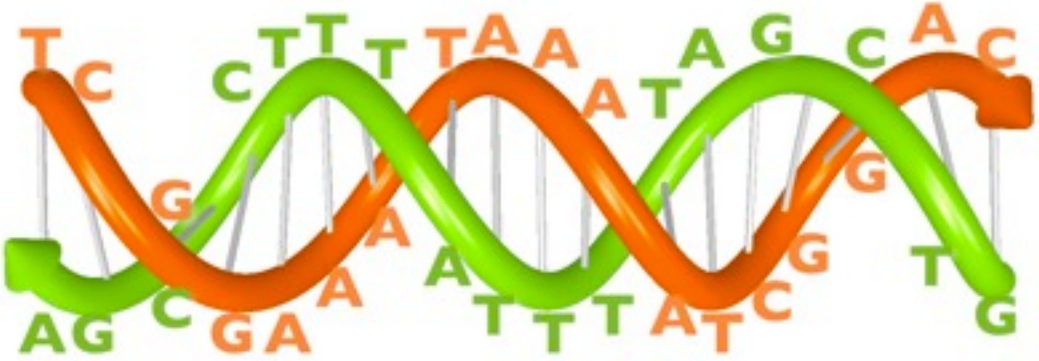
- DNA used in an entirely new way (NOT genes)



Basics of DNA

Nucleotides

A	Adenine
T	Thymine
C	Cytosine
G	Guanine


T C C T T T T T A A A A T A G C A C
G A G G T
A G C G A A A T T T A T C G T G

strand 1

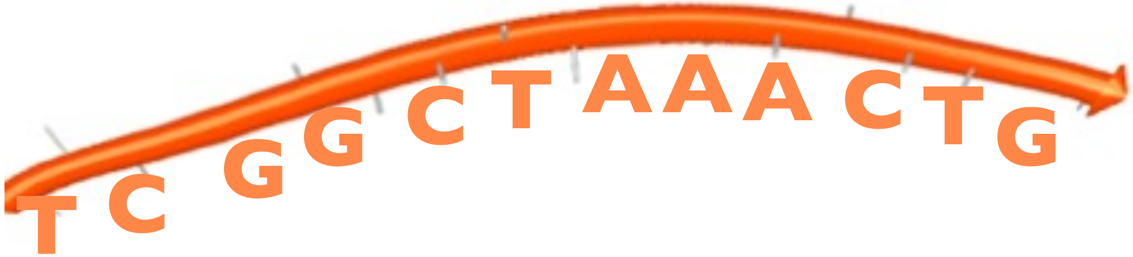
strand 2=(strand 1)*

Binding

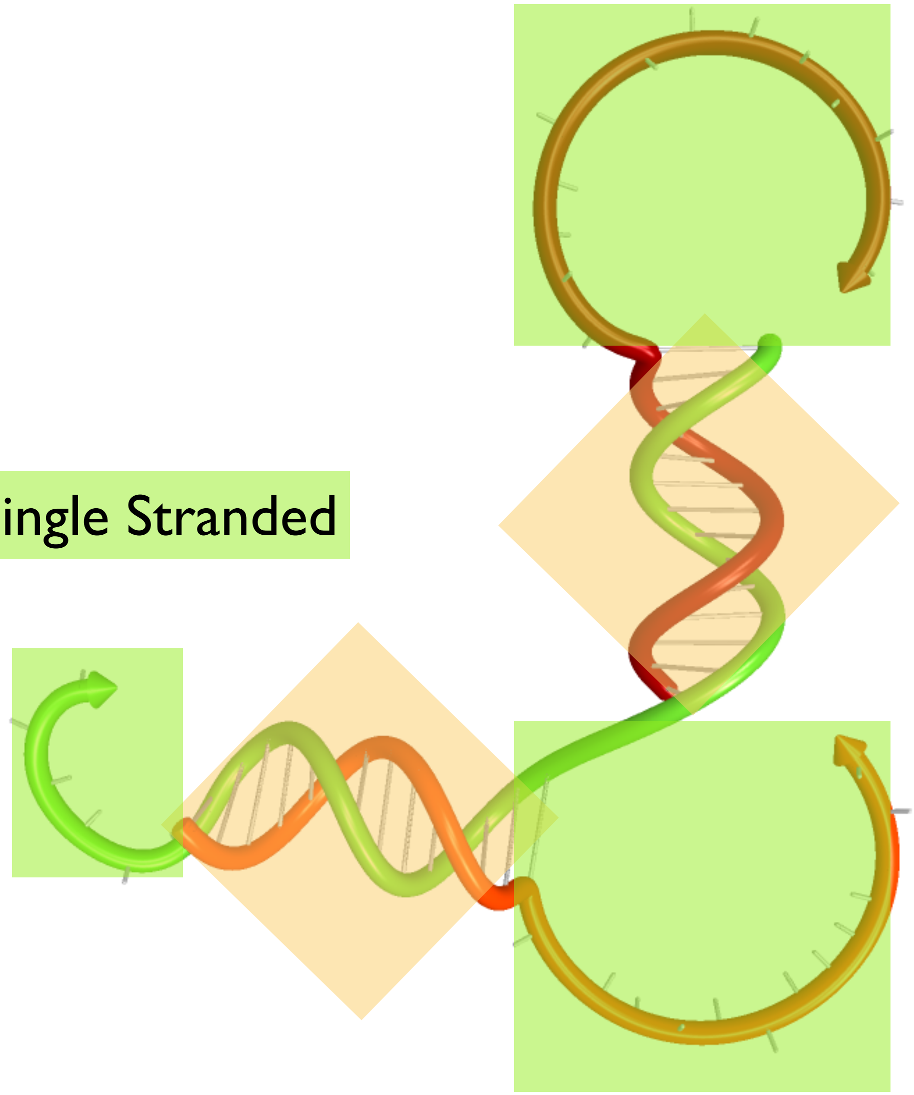
A — T

C — G

Basics of DNA



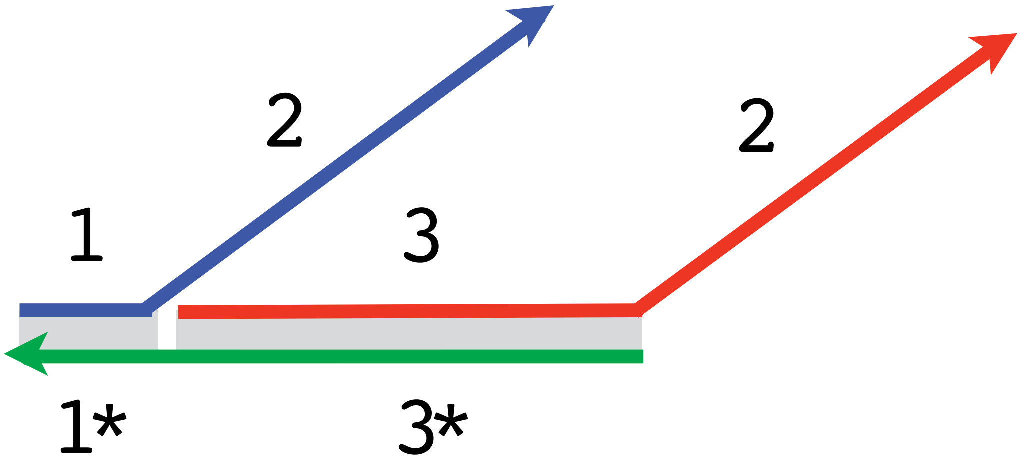
Multi-stranded **Complex**



Single Stranded

Double Stranded

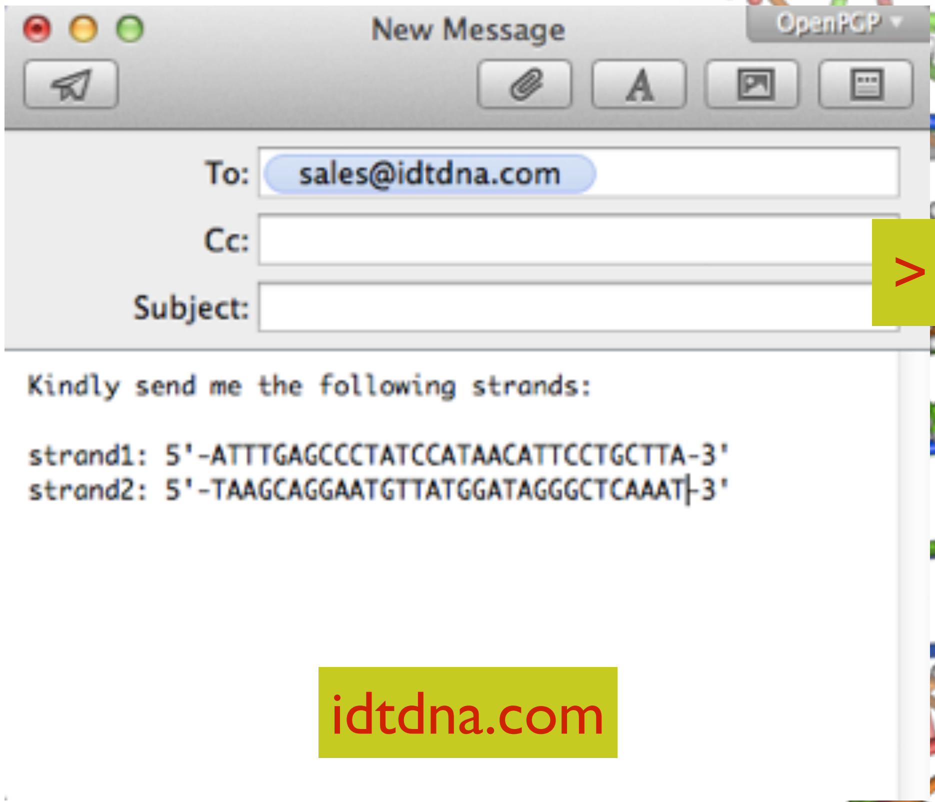
Multi-stranded **Complex**



1	=	CCGGGAA
2	=	GCCAGTGCTCTACACA
3	=	CTAATGACAGTCTGGC

domains

DNA = Commodity Chemical



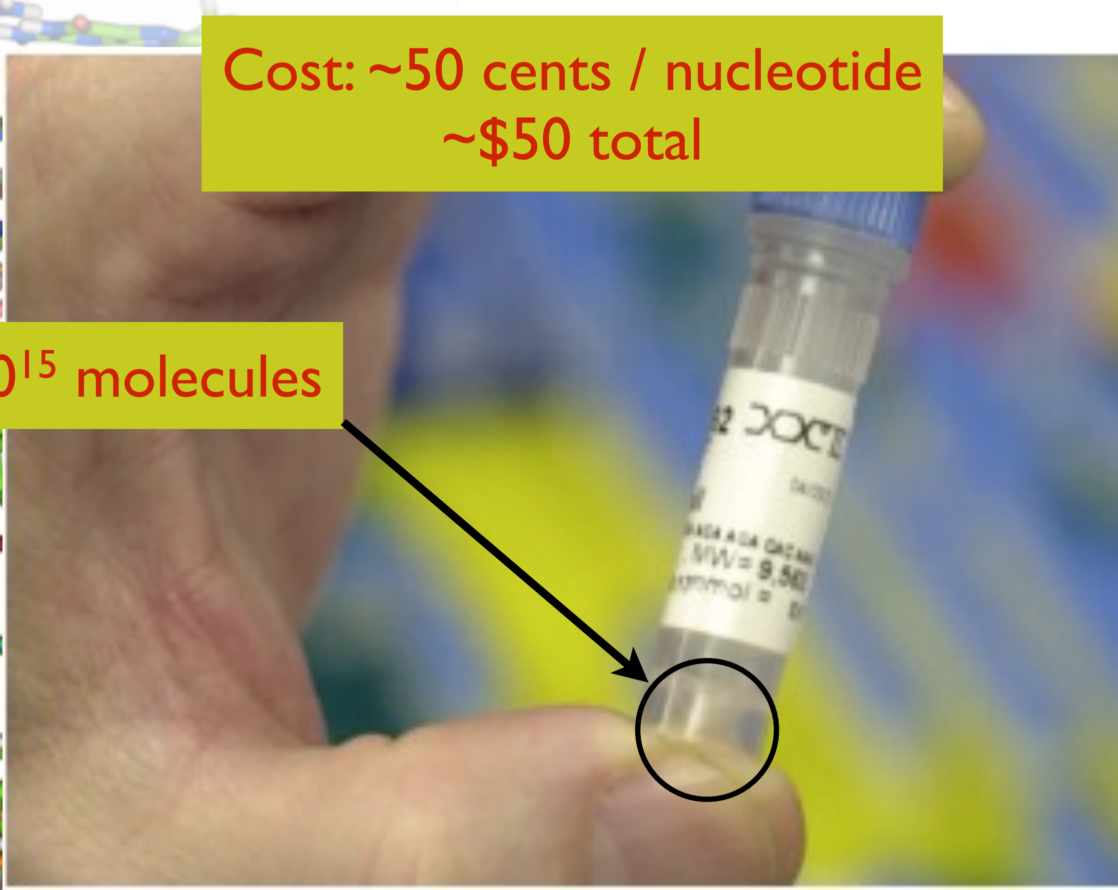
To: sales@idtdna.com

Cc:

Subject:

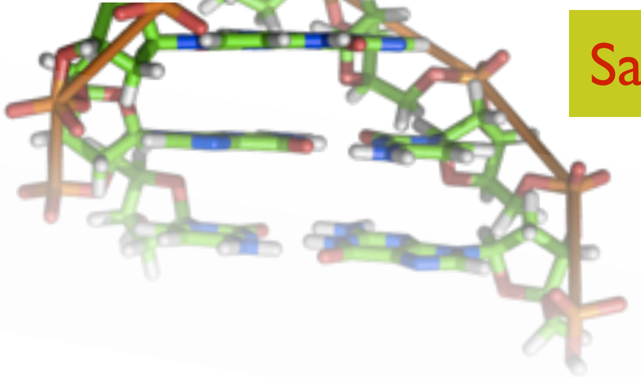
```
Kindly send me the following strands:

strand1: 5'-ATTTGAGCCCTATCCATAACATTCCTGCTTA-3'
strand2: 5'-TAAGCAGGAATGTTATGGATAGGGCTCAAAT-3'
```

idtdna.com



Cost: ~50 cents / nucleotide
~$50 total

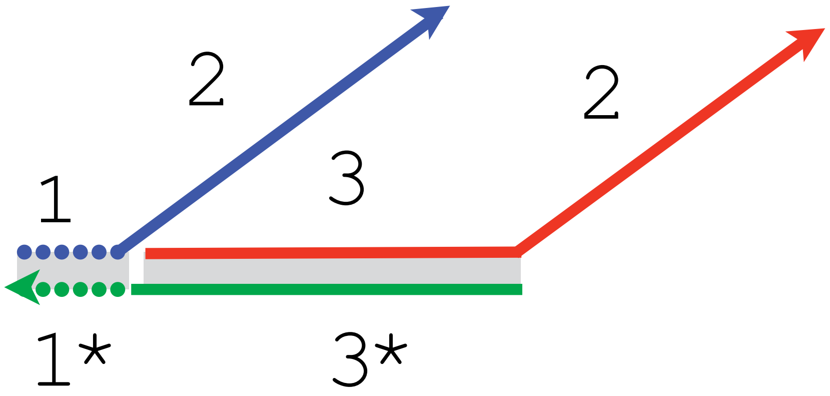
>10^{15} molecules



Same day synthesis

Strand Displacement Cascades

Complexes Should Contain Two Types of Domains: Short and Long



••••••
••••••

short domains: < 8 nucleotides

bind weakly

———
———
———

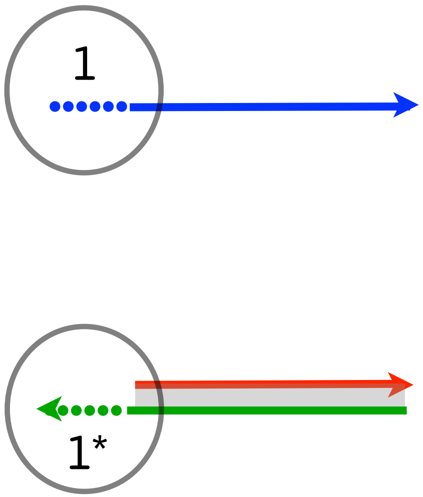
long domains: > 15 nucleotides

bind strongly

Design Complexes To Obey **3 Rules**

Rule 1: Bind

Example

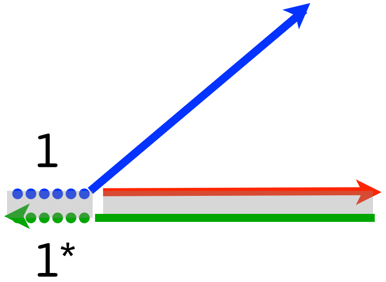


single-stranded
complementary
domains

Design Complexes To Obey **3 Rules**

Rule 1: Bind

Example

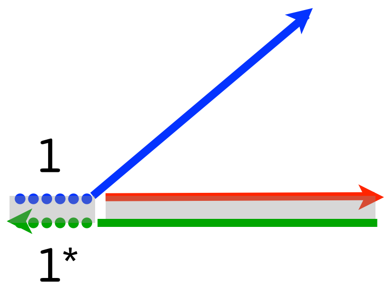


Design Complexes To Obey **3 Rules**

Rule 1: Bind

Two single-stranded complementary domains can **bind**

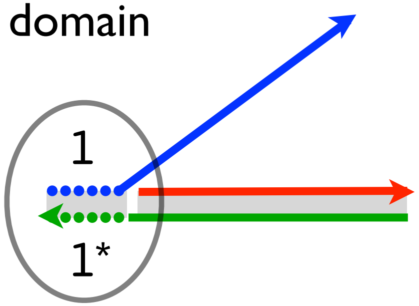
Example



Design Complexes To Obey **3 Rules**

Rule 2: Release

Example

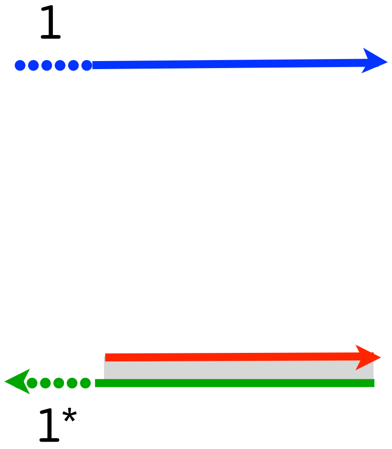


blue strand bound by only a short domain

Design Complexes To Obey **3 Rules**

Rule 2: Release

Example

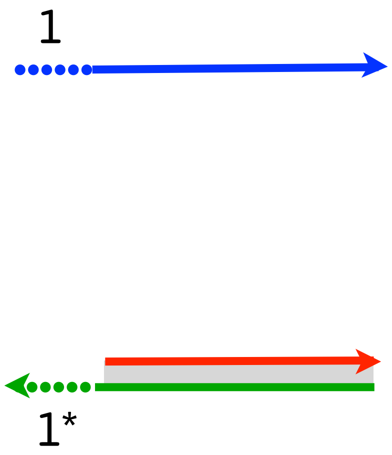


Design Complexes To Obey **3 Rules**

Rule 2: Release

Any strand bound by only a short domain can **release**

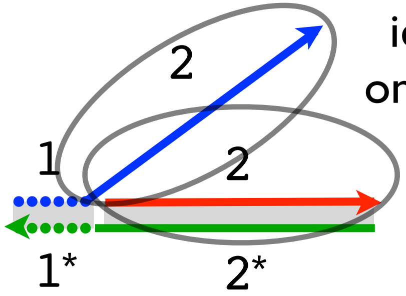
Example



Design Complexes To Obey **3 Rules**

Rule 3: Displace

Example

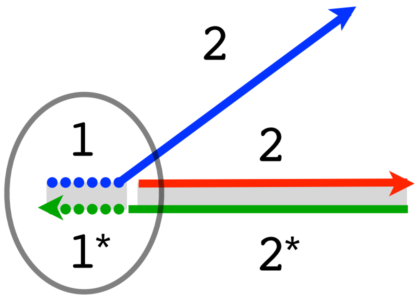


identical domains,
one bound, one free

Design Complexes To Obey **3 Rules**

Rule 3: Displace

Example

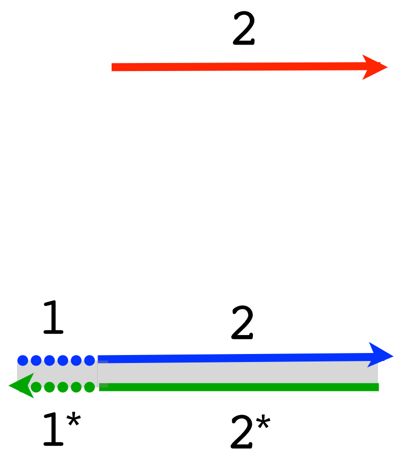


already bound

Design Complexes To Obey **3 Rules**

Rule 3: Displace

Example

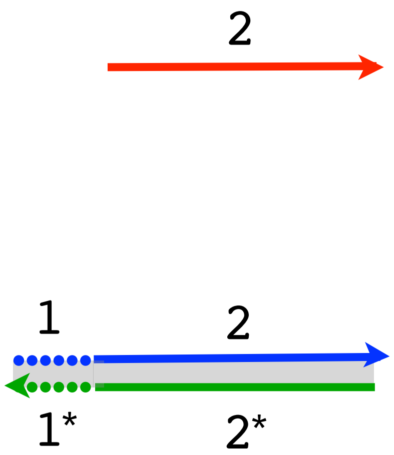


Design Complexes To Obey **3 Rules**

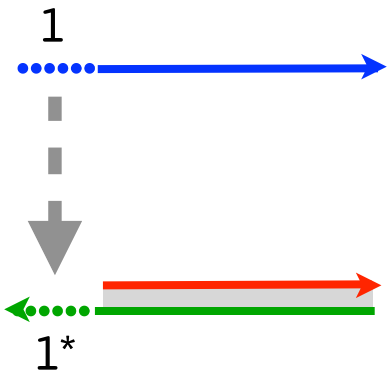
Rule 3: Displace

A domain can **displace** an identical domain of another strand, *if neighboring domains are already bound*

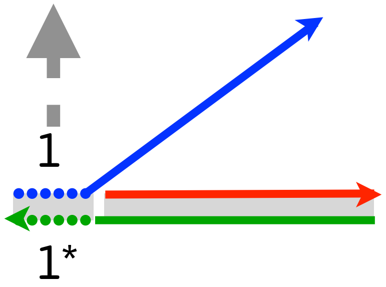
Example



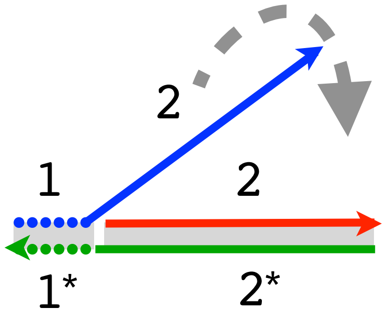
Design Complexes To Obey **3 Rules**



Bind



Release

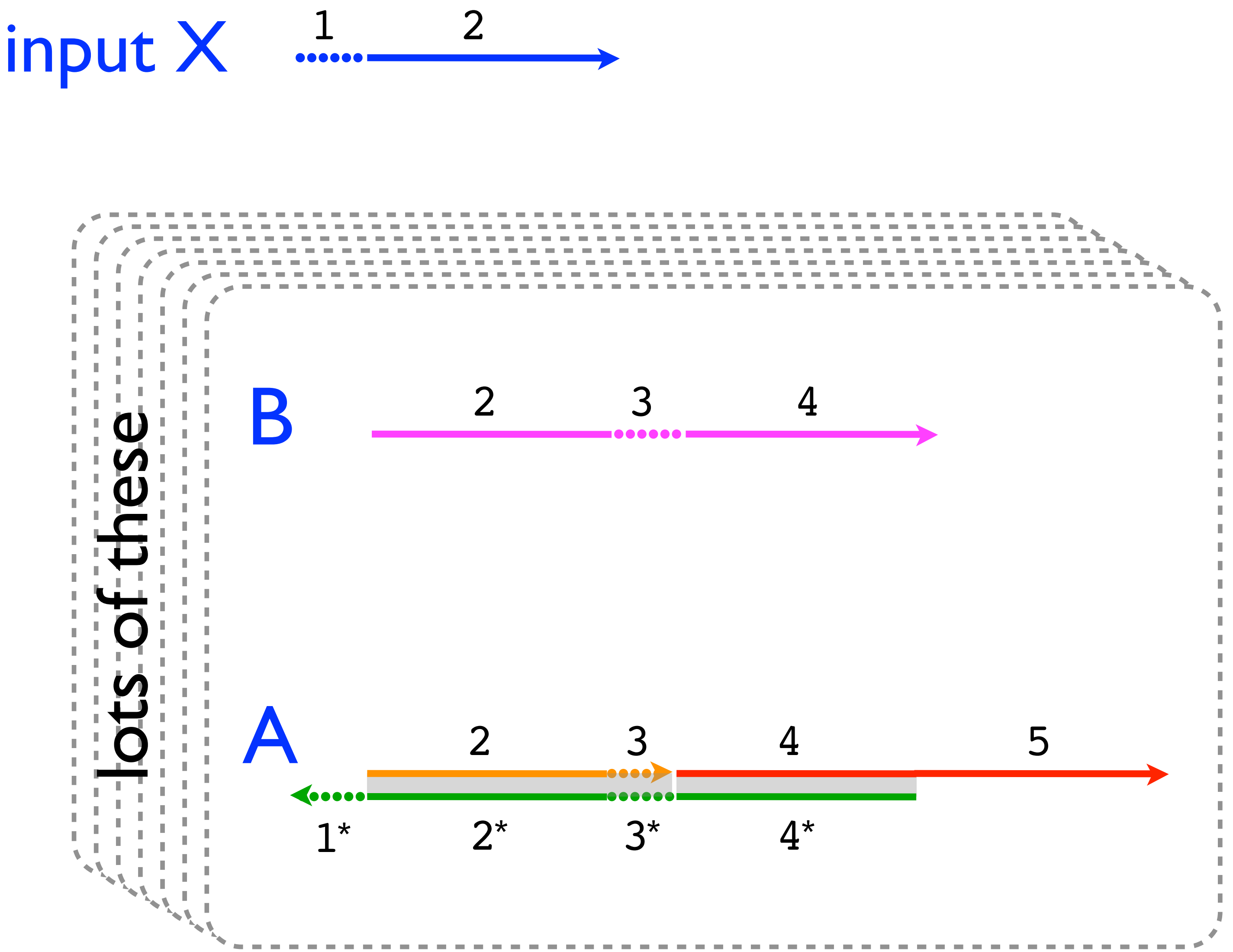


Displace

rate designable by
short domain sequences
(over 6 orders of magnitude)

Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present

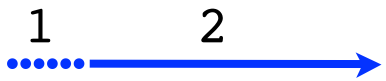


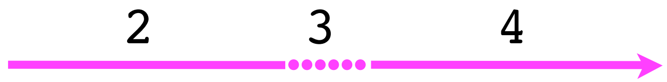
Based on: Zhang, Turberfield, Yurke, Winfree, *Science* 2007

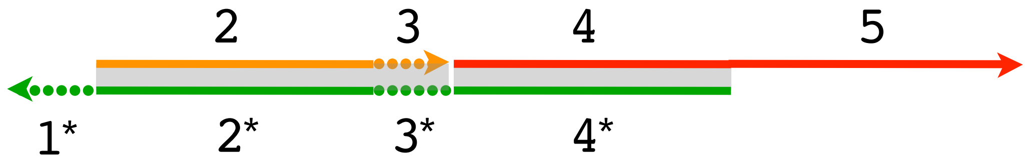
Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present

input X

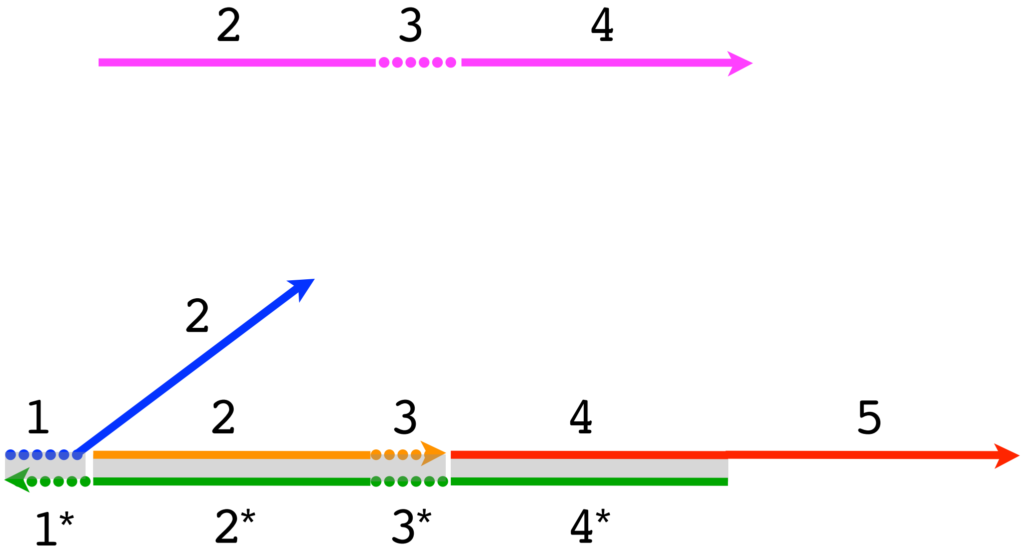






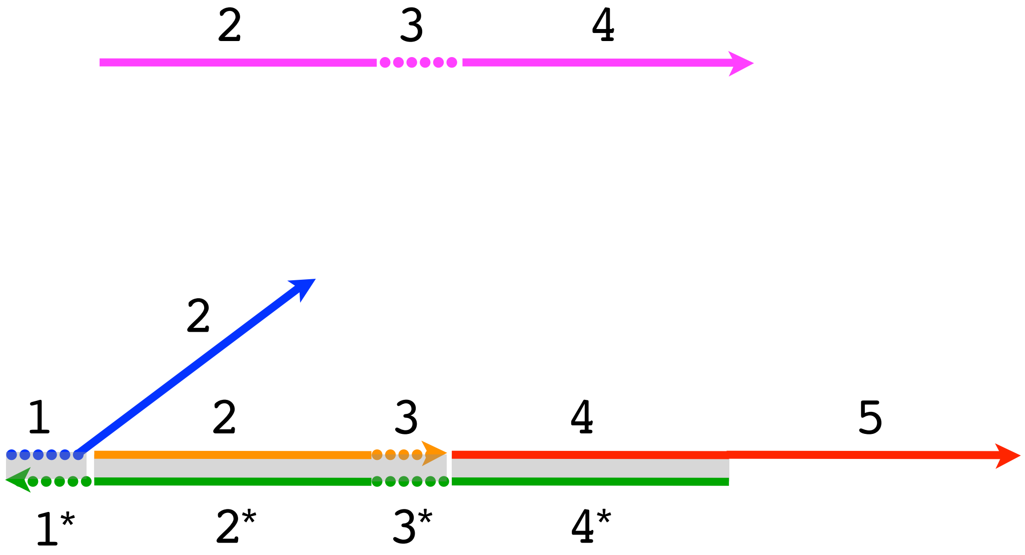
Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present



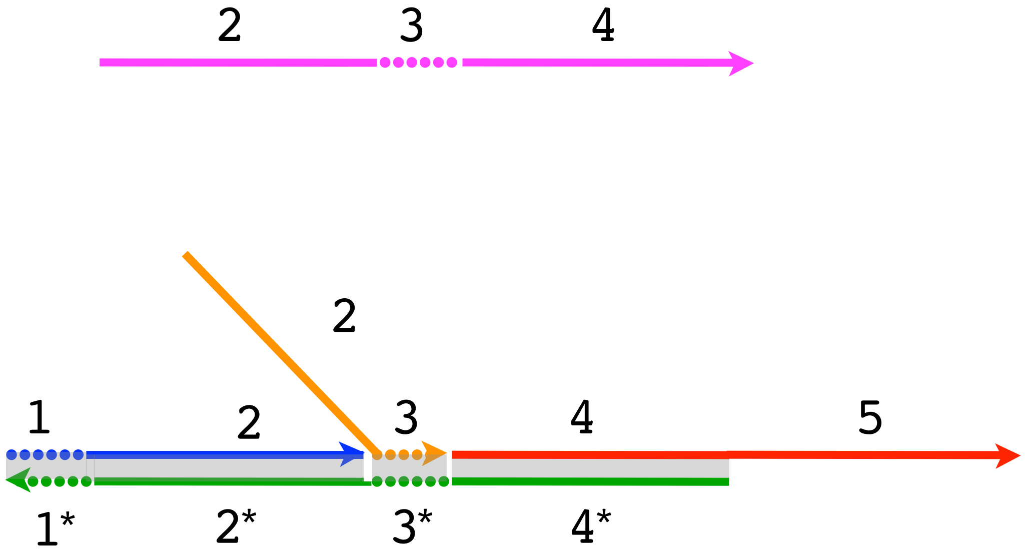
Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present



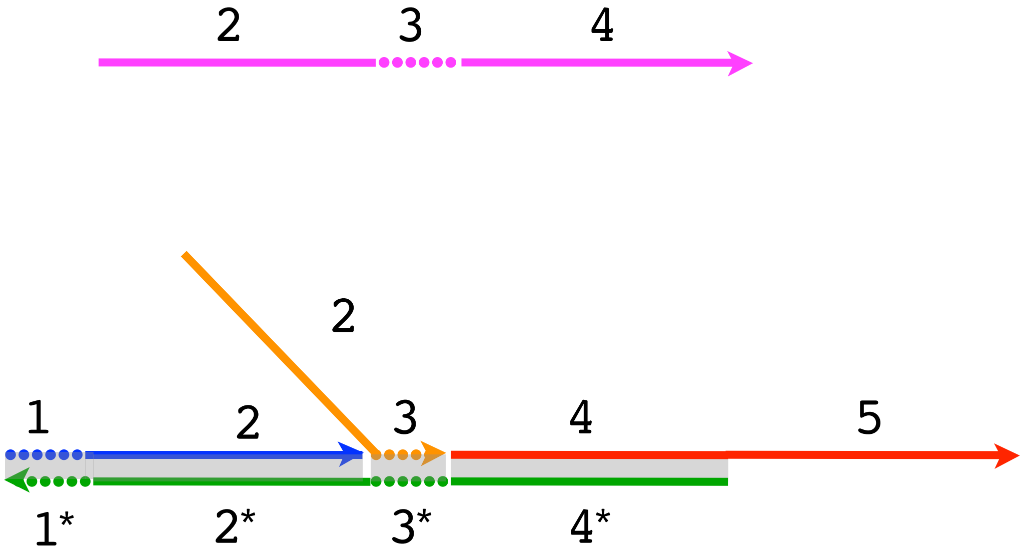
Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present



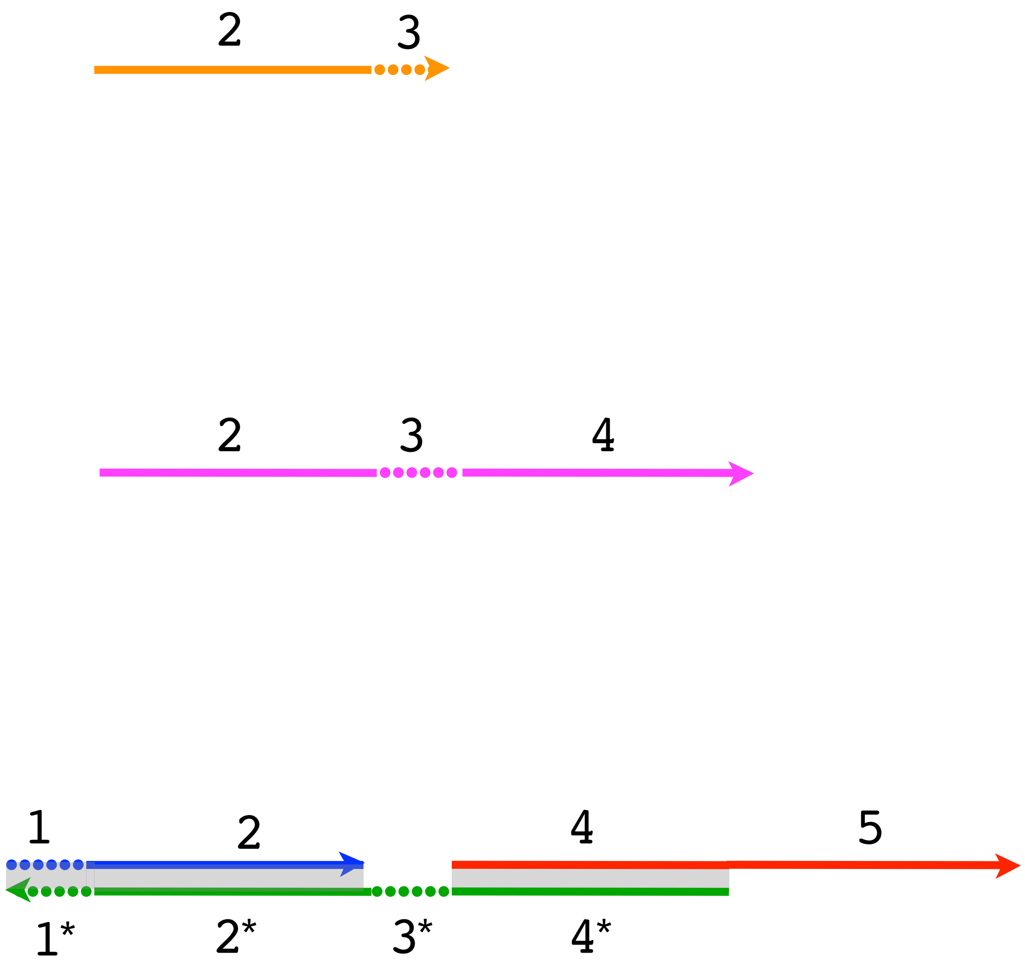
Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present



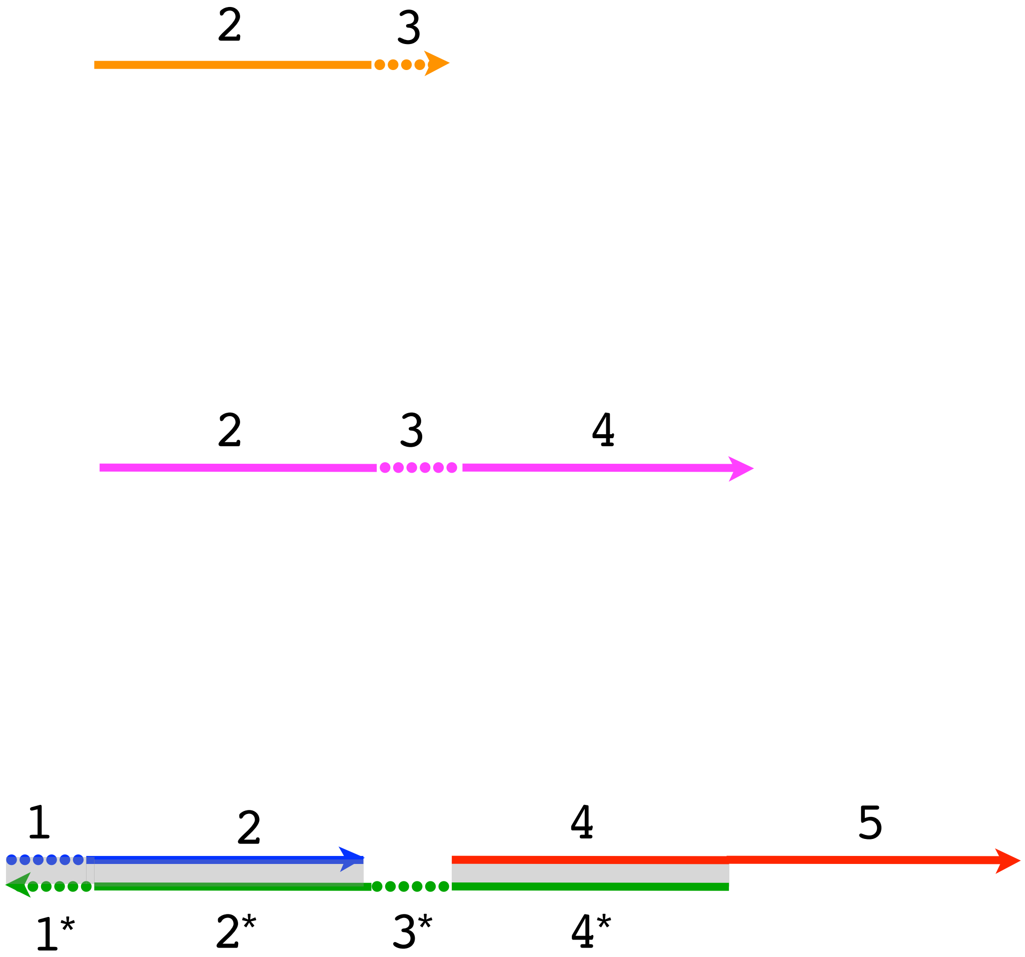
Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present



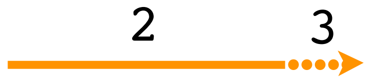
Strand Displacement Cascades Example: Amplifier

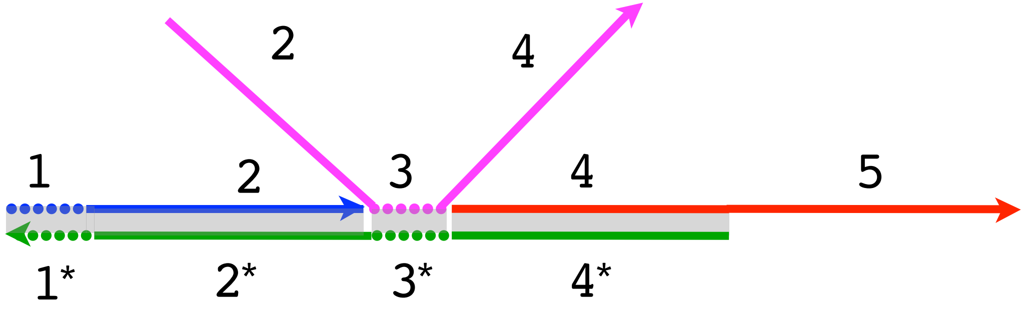
generate a lot of output Y if even a little of input X is present



Strand Displacement Cascades Example: Amplifier

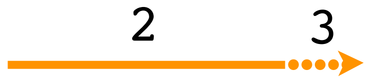
generate a lot of output Y if even a little of input X is present

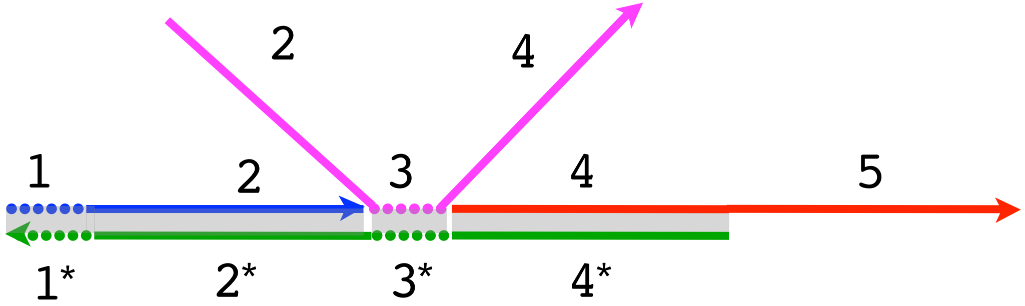




Strand Displacement Cascades Example: Amplifier

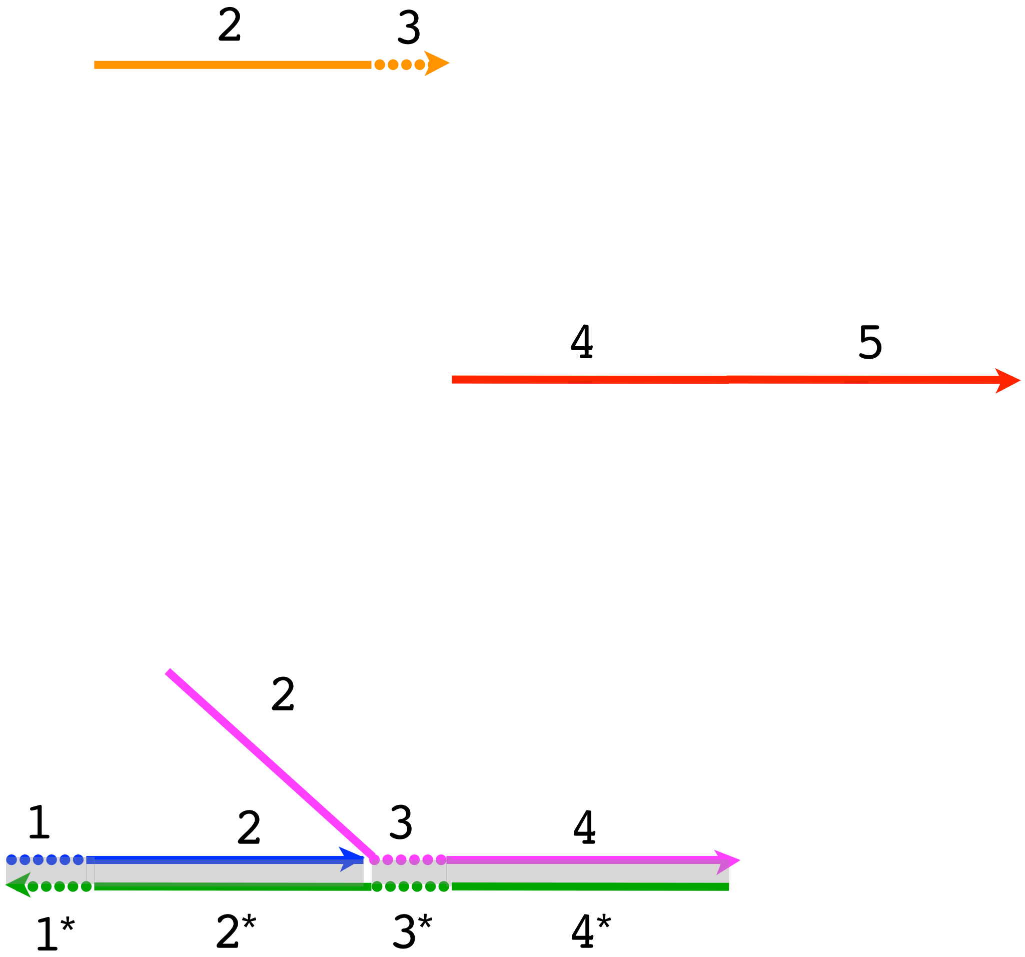
generate a lot of output Y if even a little of input X is present





Strand Displacement Cascades Example: Amplifier

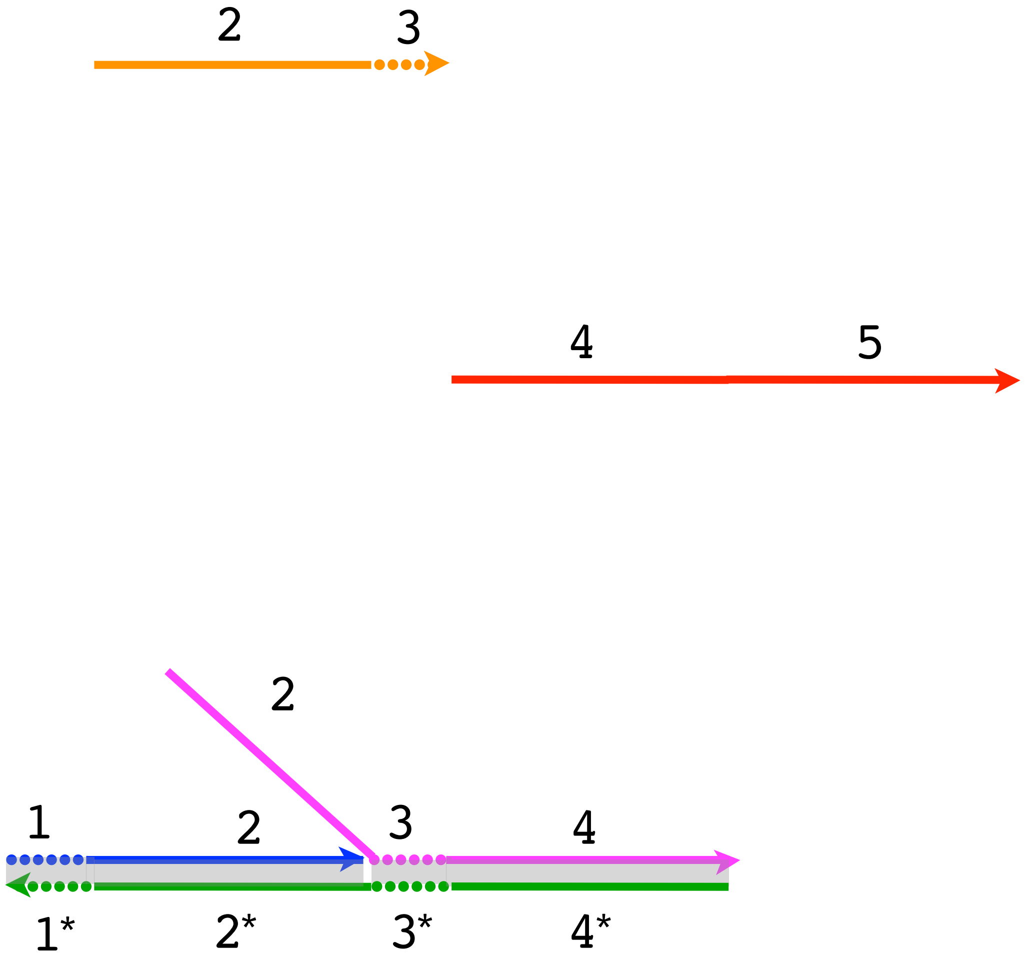
generate a lot of output Y if even a little of input X is present



output Y

Strand Displacement Cascades Example: Amplifier

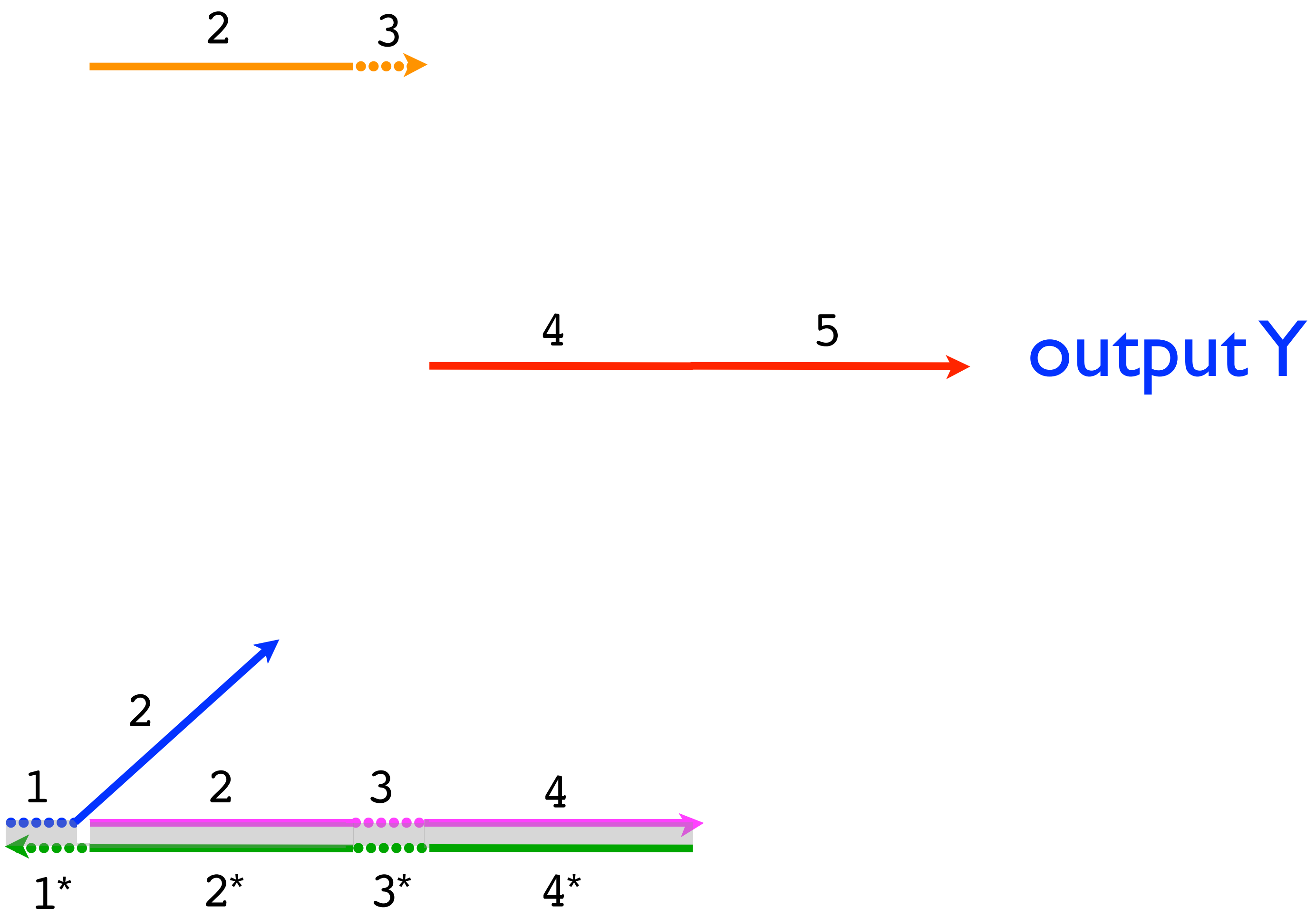
generate a lot of output Y if even a little of input X is present



output Y

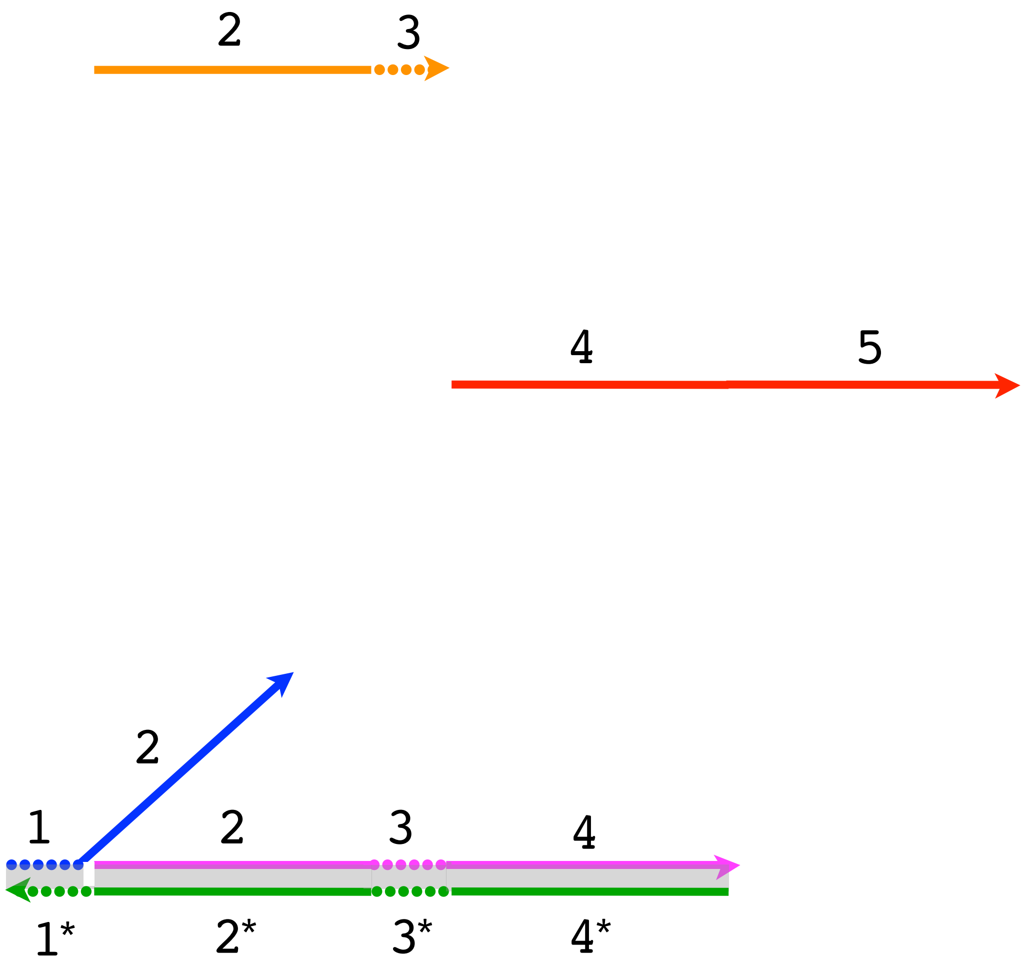
Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present



Strand Displacement Cascades Example: Amplifier

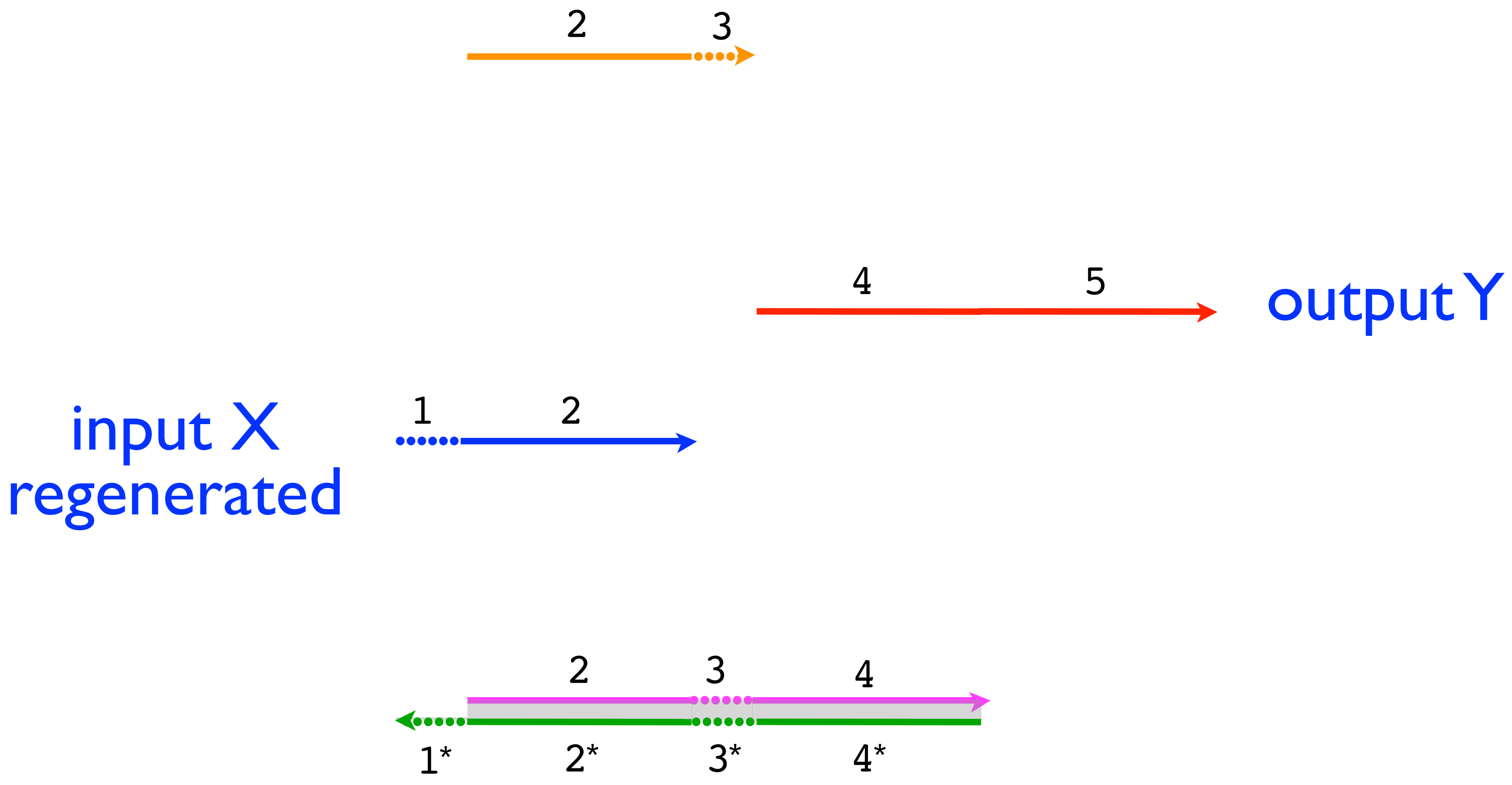
generate a lot of output Y if even a little of input X is present



output Y

Strand Displacement Cascades Example: Amplifier

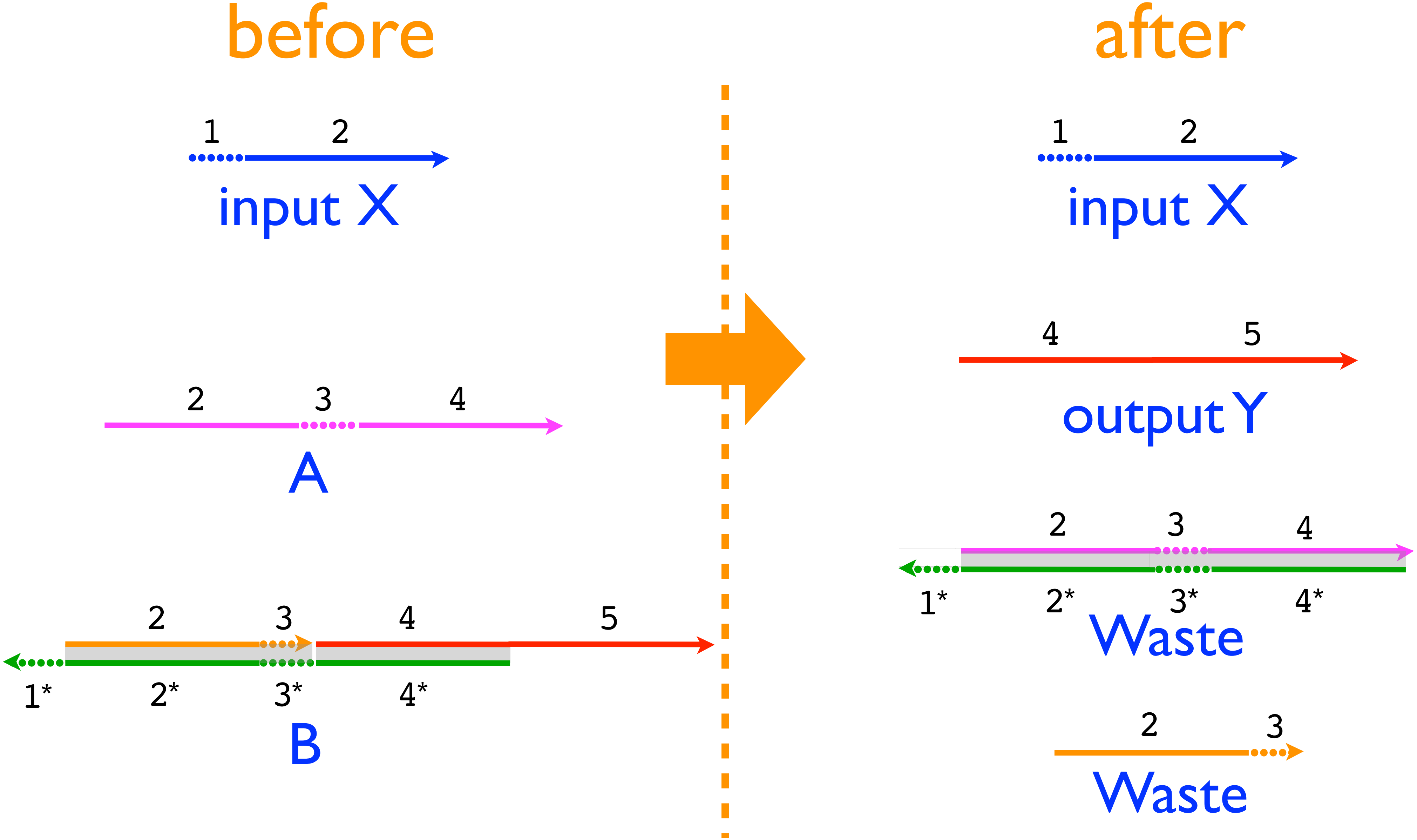
generate a lot of output Y if even a little of input X is present



Strand Displacement Cascades Example: Amplifier

generate a lot of output Y if even a little of input X is present

$$X \rightarrow X+Y$$

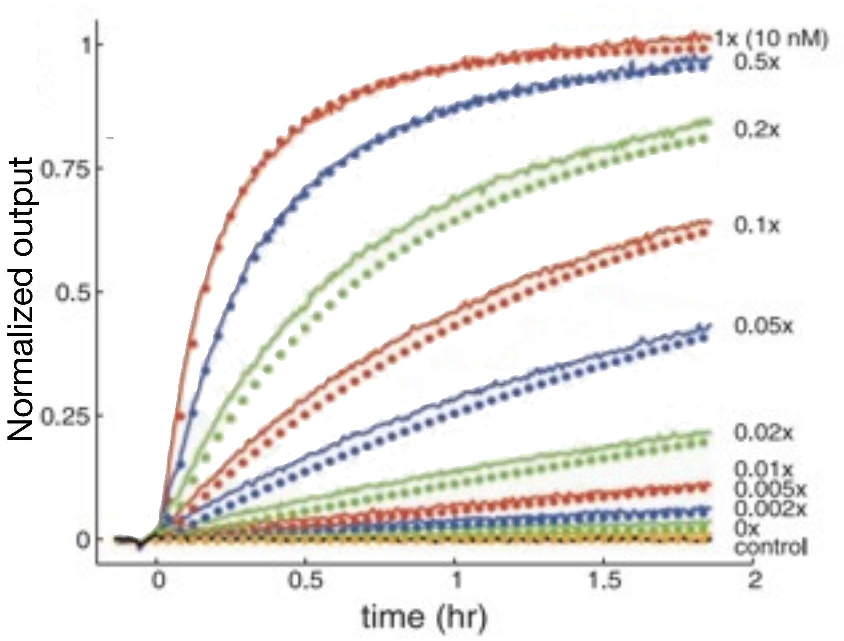


Wet-lab implementation of amplifier

generate a lot of output Y if even a little of input X is present

varying amount of input strand

more output than input produced



---- 3 rules model

—— measurement

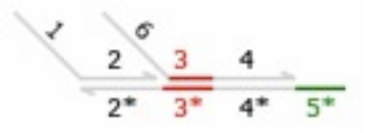
Zhang, Turberfield, Yurke, Winfree, *Science* 2007

Formal Analysis of Strand Displacement Cascades

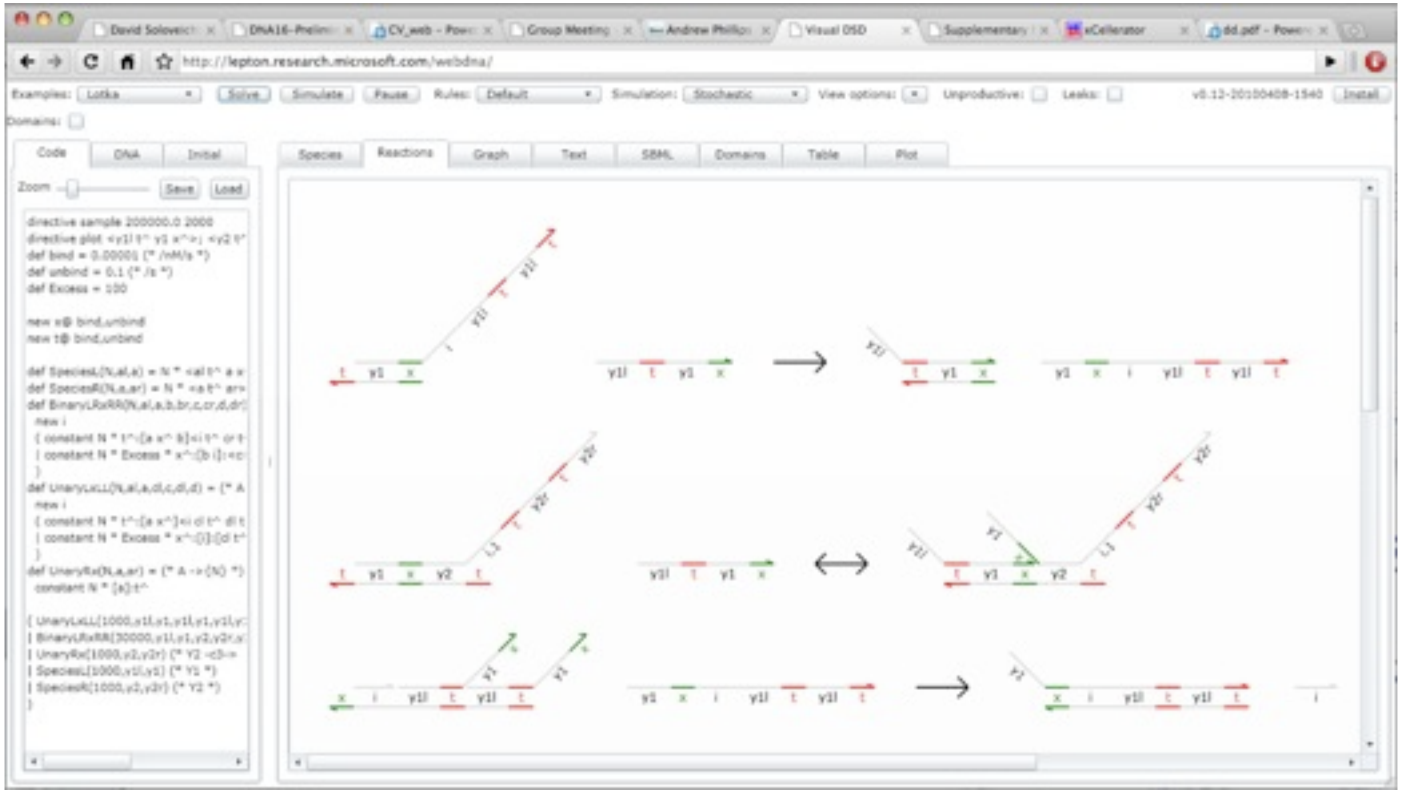
DSD: formal language for describing and modeling strand displacement cascades

<1>[2]:<6>[3^ 4]:5^*

=



http://lepton.research.microsoft.com/webdna/



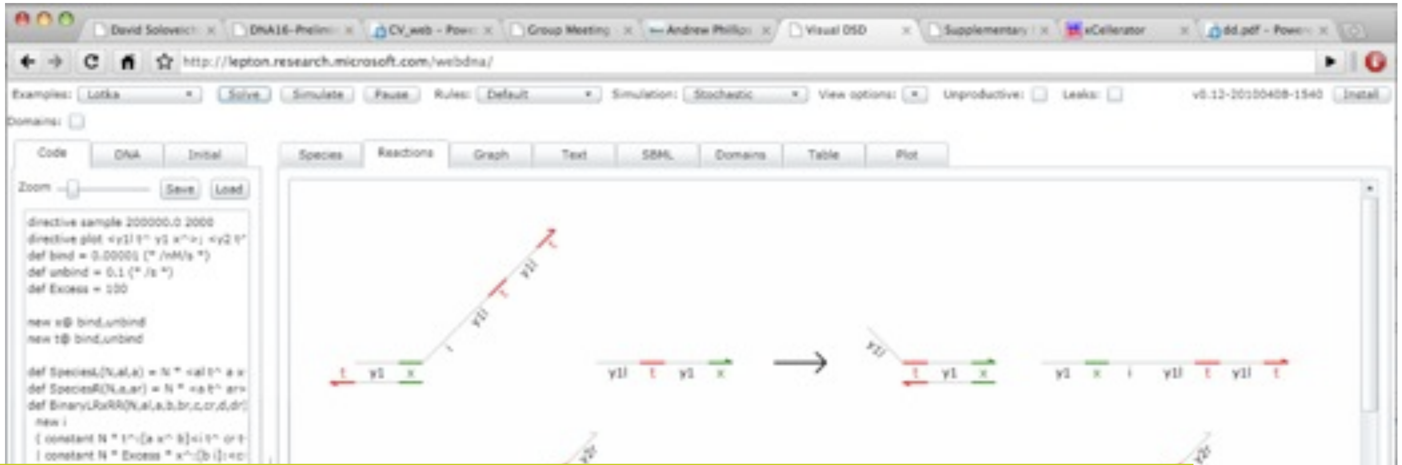
```
directive sample 200000.0 2000
directive plot <y1l t^ y1 x^>; <y2 t^
def bind = 0.00001 (* /nM/s *)
def unbind = 0.1 (* /s *)
def Excess = 100

new x@ bind,unbind
new t@ bind,unbind

def SpeciesL(N,al,a) = N * <al t^ a x
def SpeciesR(N,a,ar) = N * <a t^ ar>
def BinaryLRxRR(N,al,a,b,br,c,cr,d,dr)
  new i
  ( constant N * t^:[a x^ b]<i t^ cr t
  | constant N * Excess * x^:[b i]:<c
  )
def UnaryLxLL(N,al,a,dl,c,dl,d) = (* A
  new i
  ( constant N * t^:[a x^]<i dl t^ dl t
  | constant N * Excess * x^:[i]:[d t^
  )
def UnaryRx(N,a,ar) = (* A ->{N} *)
  constant N * [a]:t^

( UnaryLxLL(1000,y1l,y1,y1l,y1,y1l,y
| BinaryLRxRR(30000,y1l,y1,y2,y2r,x
| UnaryRx(1000,y2,y2r) (* Y2 ->
| SpeciesL(1000,y1l,y1) (* Y1 *)
| SpeciesR(1000,y2,y2r) (* Y2 *)
)
```

Phillips, Cardelli, *Journal of Royal Society Interface*, 2009

Formal Analysis of Strand Displacement Cascades

DSD: formal language for describing and modeling strand displacement cascades

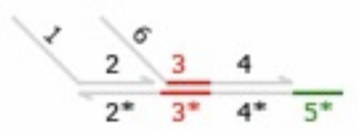
http://lepton.research.microsoft.com/webdna/



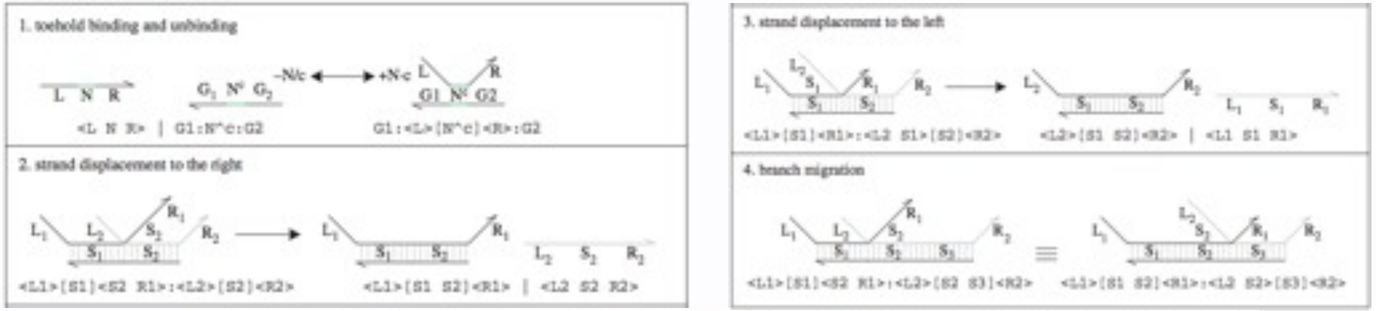
```
directive sample 200000.0 2000
directive plot <y1! t^ y1 x^>; <y2 t^
def bind = 0.00001 (* /nM/s *)
def unbind = 0.1 (* /s *)
def Excess = 100

new x@ bind,unbind
new t@ bind,unbind

def SpeciesL(N,al,a) = N * <al t^ a x
def SpeciesR(N,a,ar) = N * <a t^ ar>
def BinaryLRxRR(N,al,a,b,br,c,cr,d,dr)
  new i
  ( constant N * t^:[a x^ b]<i t^ cr t
  | constant N * Excess * x^:[b i]:<c
```

<1>[2]:<6>[3^ 4]:5^*

=



formal semantics

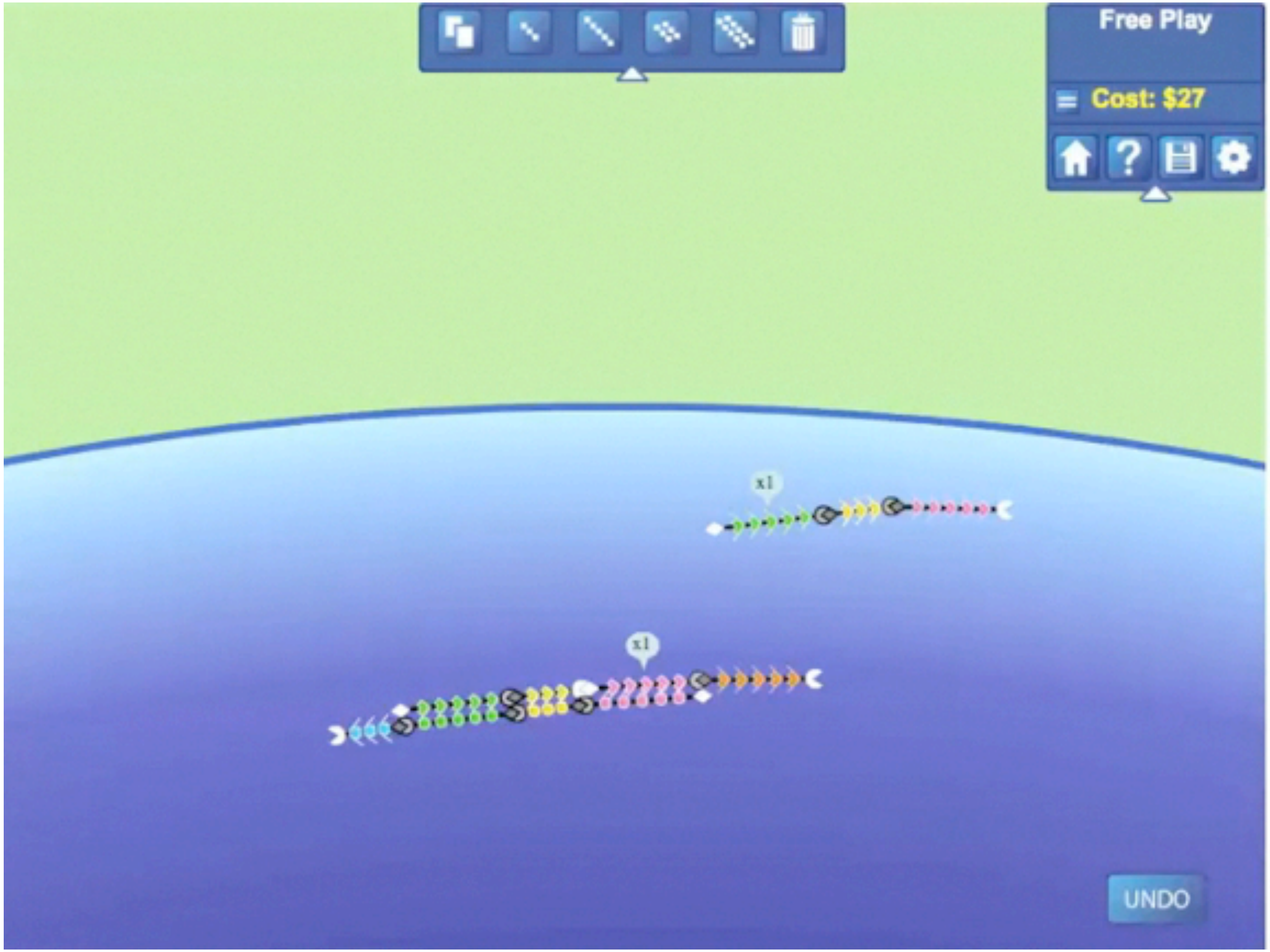


```
<L N R> | G1:N^c:G2        G1:<L>[N^c]<R>:G2
<L1>[S1]<S2 R1>:<L2>[S2]<R2>        <L1>[S1 S2]<R1> | <L2 S2 R2>
<L1>[S1]<R1>:<L2 S1>[S2]<R2>        <L2>[S1 S2]<R2> | <L1 S1 R1>
<L1>[S1]<S2 R1>:<L2>[S2 S3]<R2>   ≡   <L1>[S1 S2]<R1>:<L2 S2>[S3]<R2>
```

Figure 2. Reduction and branch migration rules of the strand displacement language. For each rule, the graphical representation at the top is equivalent to the program code at the bottom.

Phillips, Cardelli, *Journal of Royal Society Interface*, 2009

Diverse Design Possibilities Make for a Game



(Beta version)

Rich Snider, Dmitry Danilov and Zoran Popovic, in collaboration with Georg Seelig, David Baker

http://nanocrafter.org/

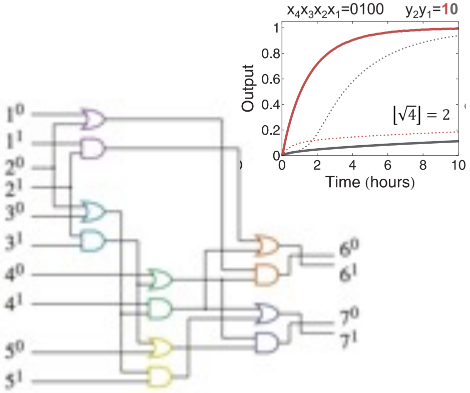
- FoldIt team
- crowd-sourcing

Strand displacement has stimulated multiple research directions in the wet-lab

strand displacement cascades

Strand displacement has stimulated multiple research directions in the wet-lab

molecular logic circuits



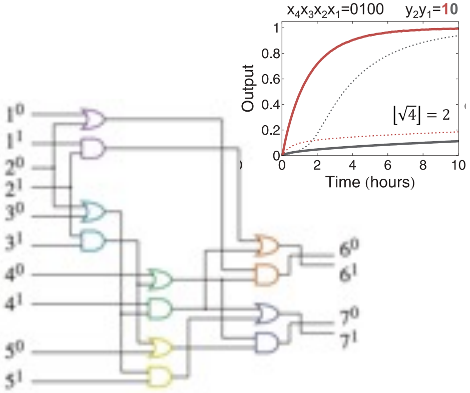
- Largest autonomous biochemical networks built from scratch

Qian, Winfree, *Science* 2011

strand displacement cascades

Strand displacement has stimulated multiple research directions in the wet-lab

molecular logic circuits

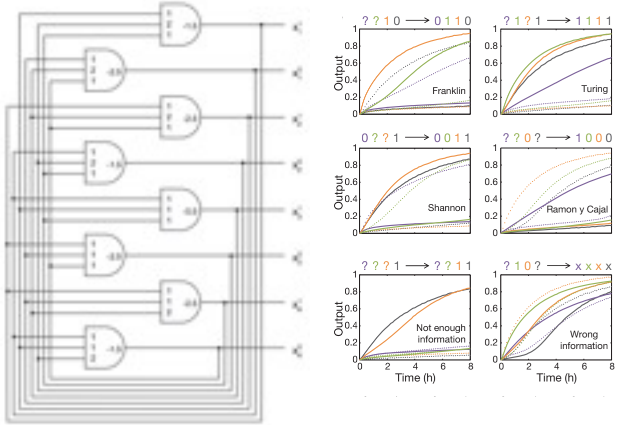


- Largest autonomous biochemical networks built from scratch

Qian, Winfree, *Science* 2011

strand displacement cascades

molecular artificial neural networks

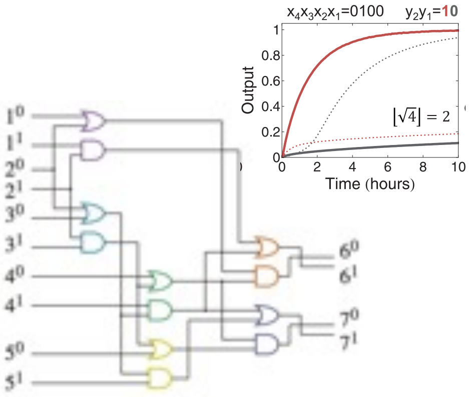


- Biochemical system doing inference

Qian, Winfree, Bruck *Nature* 2011

Strand displacement has stimulated multiple research directions in the wet-lab

molecular logic circuits

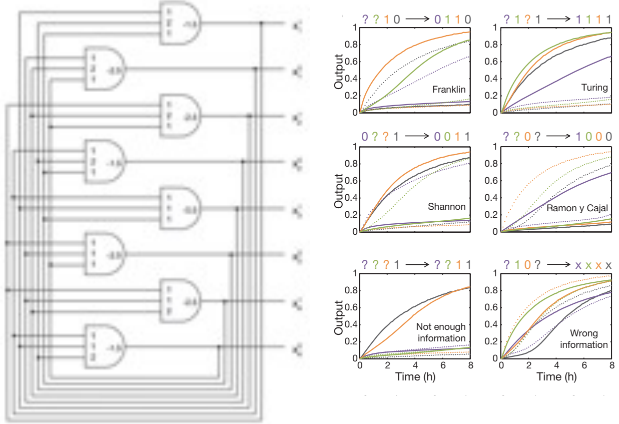


- Largest autonomous biochemical networks built from scratch

Qian, Winfree, *Science* 2011

strand displacement cascades

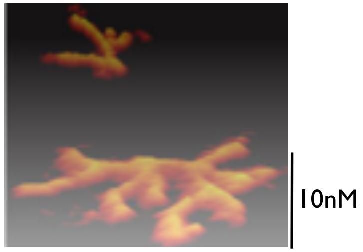
molecular artificial neural networks



- Biochemical system doing inference

Qian, Winfree, Bruck *Nature* 2011

controlling assembly of nanoscale structures

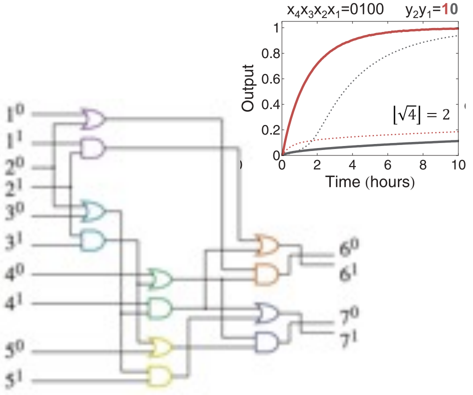


- Prescribed nanoscale structures seen under atomic force microscope

Yin, Choi, Calvert, Yurke, Pierce *Nature* 2008

Strand displacement has stimulated multiple research directions in the wet-lab

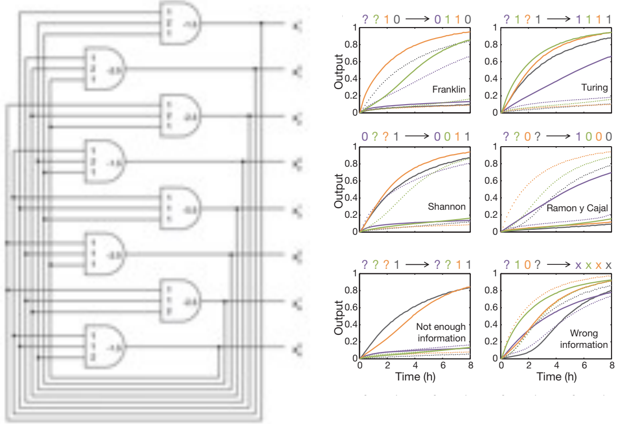
molecular logic circuits



- Largest autonomous biochemical networks built from scratch

Qian, Winfree, *Science* 2011

molecular artificial neural networks

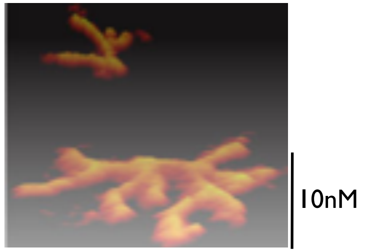


- Biochemical system doing inference

Qian, Winfree, Bruck *Nature* 2011

strand displacement cascades

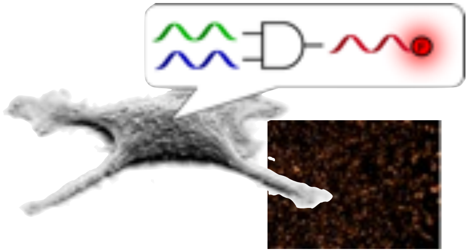
controlling assembly of nanoscale structures



- Prescribed nanoscale structures seen under atomic force microscope

Yin, Choi, Calvert, Yurke, Pierce *Nature* 2008

strand displacement in mammalian cells

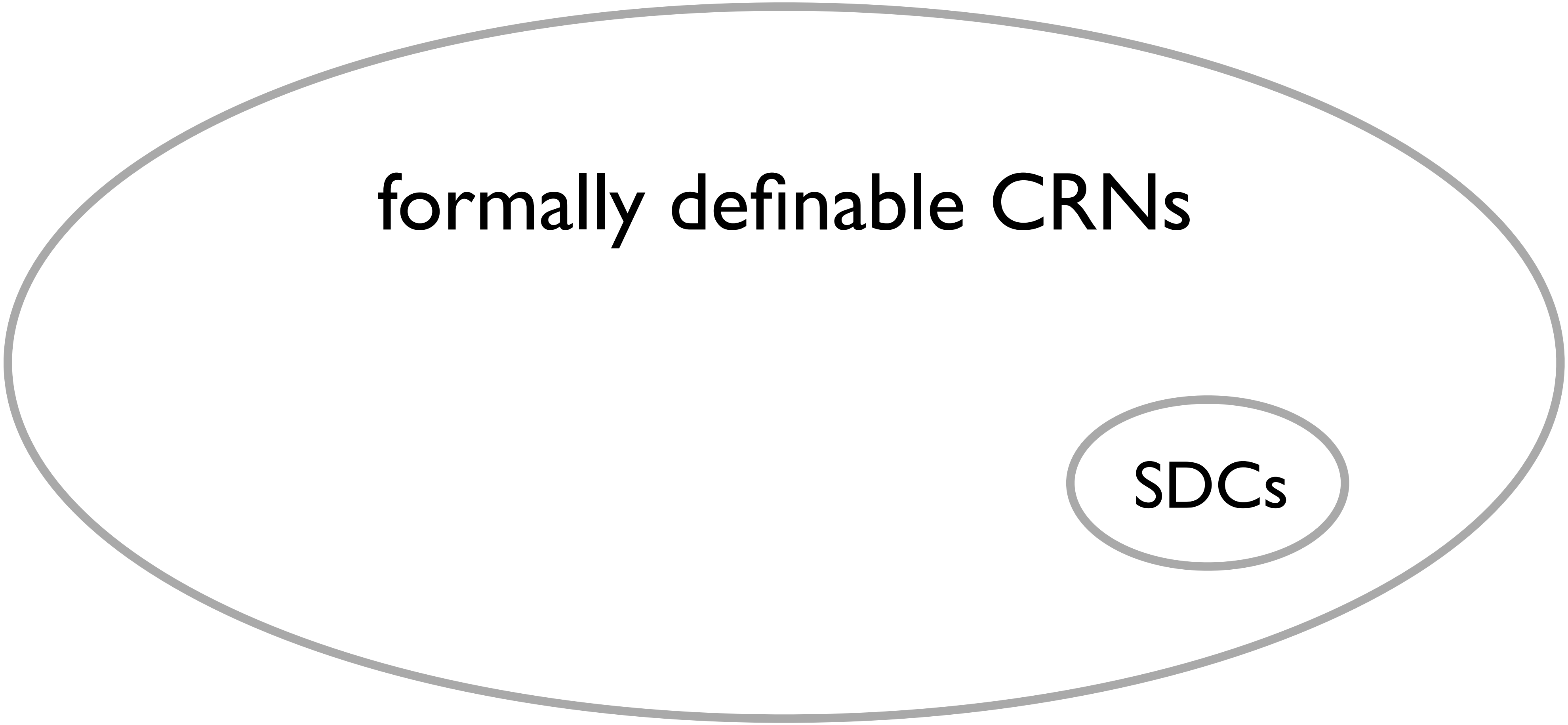


- Logic on biological signals

Hemphill, Deiters *J Am Chem Soc* 2013

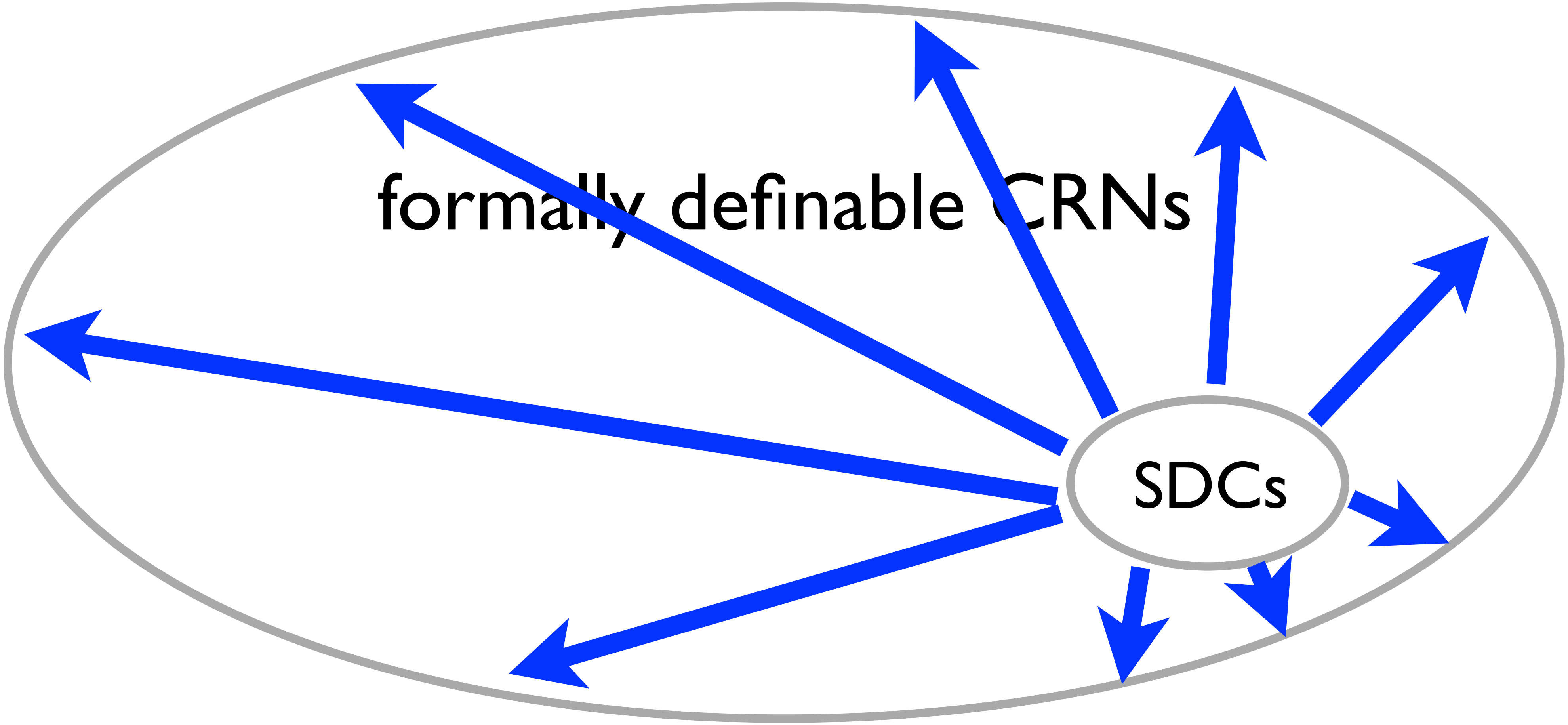
Strand displacement cascades are complete for chemical reaction networks

Soloveichik, Seelig, Winfree, “DNA as a Universal Substrate for Chemical Kinetics”, *PNAS*, 2010



Strand displacement cascades are complete for chemical reaction networks

Soloveichik, Seelig, Winfree, "DNA as a Universal Substrate for Chemical Kinetics", *PNAS*, 2010



Strand Displacement Implementation of the Approximate Majority Network

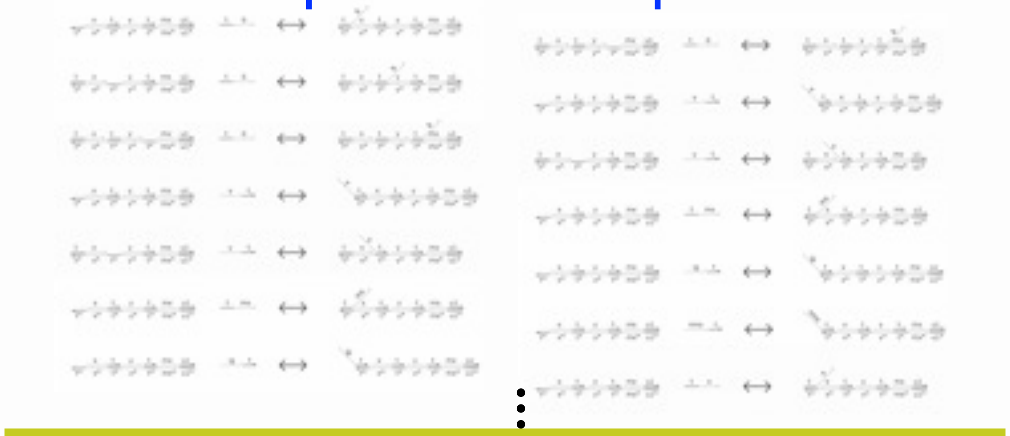
Goal: Approximate Majority

$$X + Y \rightarrow B + Y$$
$$X + Y \rightarrow X + B$$
$$B + X \rightarrow X + X$$
$$B + Y \rightarrow Y + Y$$

compile

Soloveichik, Seelig, Winfree *PNAS* 2010

Strand Displacement Implementation



"3 rules" reactions

Strand Displacement Implementation of the Approximate Majority Network

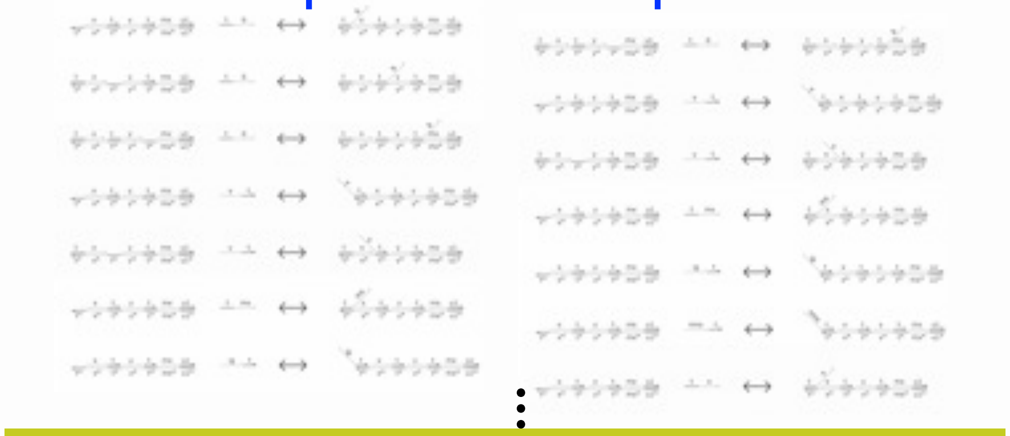
Goal: Approximate Majority

$$X + Y \rightarrow B + Y$$
$$X + Y \rightarrow X + B$$
$$B + X \rightarrow X + X$$
$$B + Y \rightarrow Y + Y$$

compile

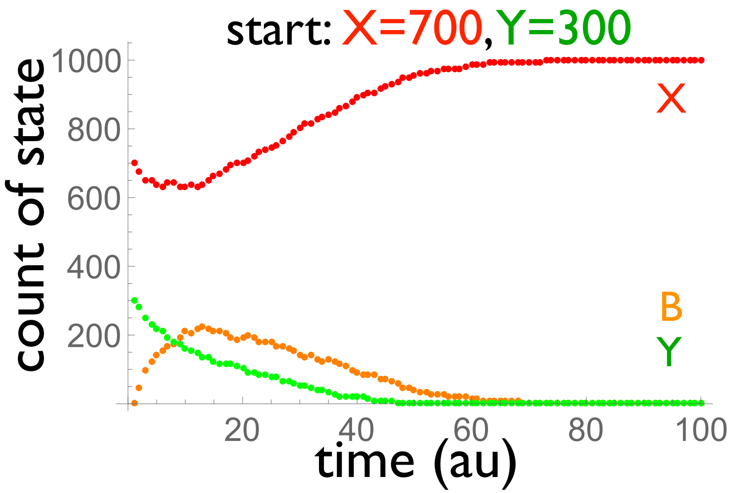
Soloveichik, Seelig, Winfree *PNAS* 2010

Strand Displacement Implementation

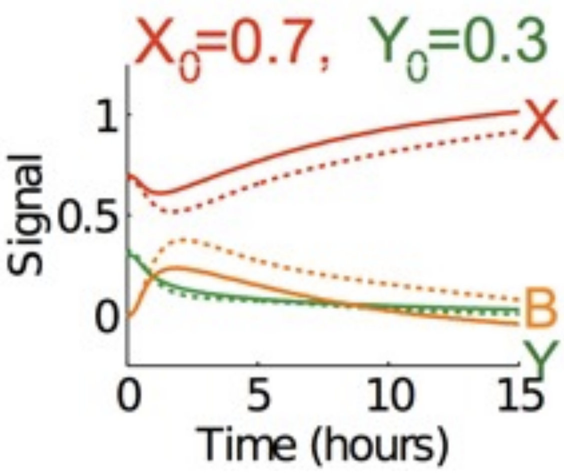


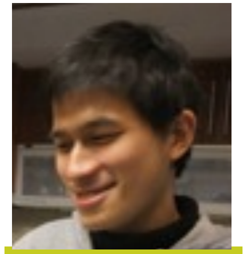
“3 rules” reactions

Ideal



Test tube





Yuan-Jyue Chen (graduate student)

Chen, Dalchau, Srinivas, Phillips, Cardelli, Soloveichik, Seelig, *Nature Nanotechnology* 2013

Strand Displacement Implementation of the Approximate Majority Network

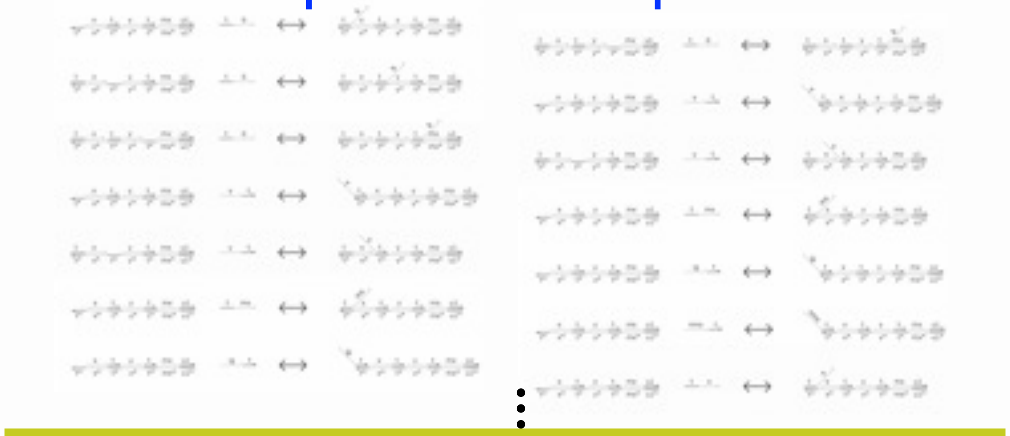
Goal: Approximate Majority

$$X + Y \rightarrow B + Y$$
$$X + Y \rightarrow X + B$$
$$B + X \rightarrow X + X$$
$$B + Y \rightarrow Y + Y$$

compile

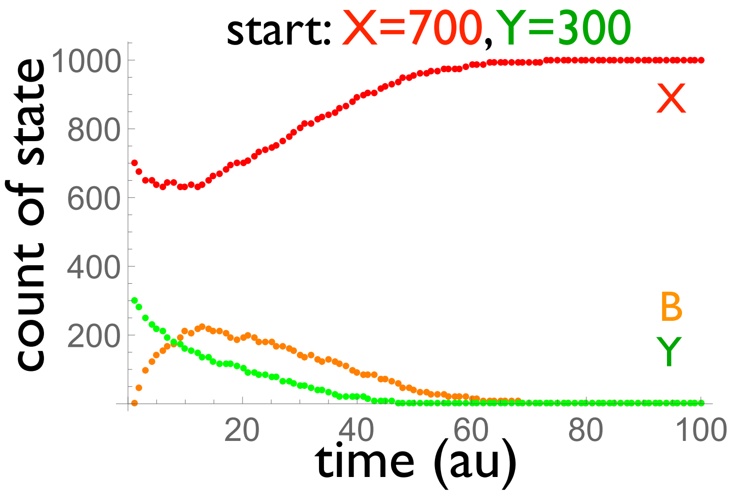
Soloveichik, Seelig, Winfree *PNAS* 2010

Strand Displacement Implementation



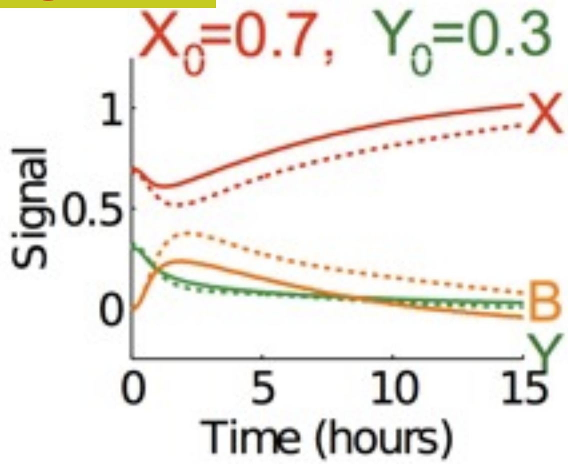
"3 rules" reactions

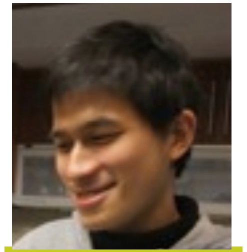
Ideal



Test tube

10^{11} agents!





Yuan-Jyue Chen (graduate student)

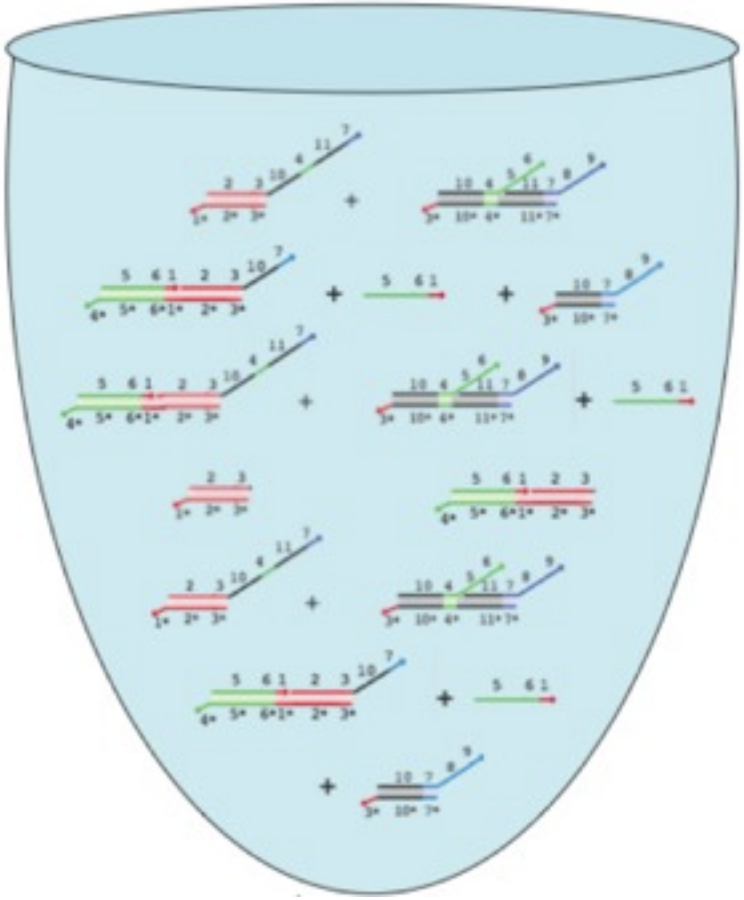
Chen, Dalchau, Srinivas, Phillips, Cardelli, Soloveichik, Seelig, *Nature Nanotechnology* 2013

Every goal reaction corresponds to a set of implementation reactions

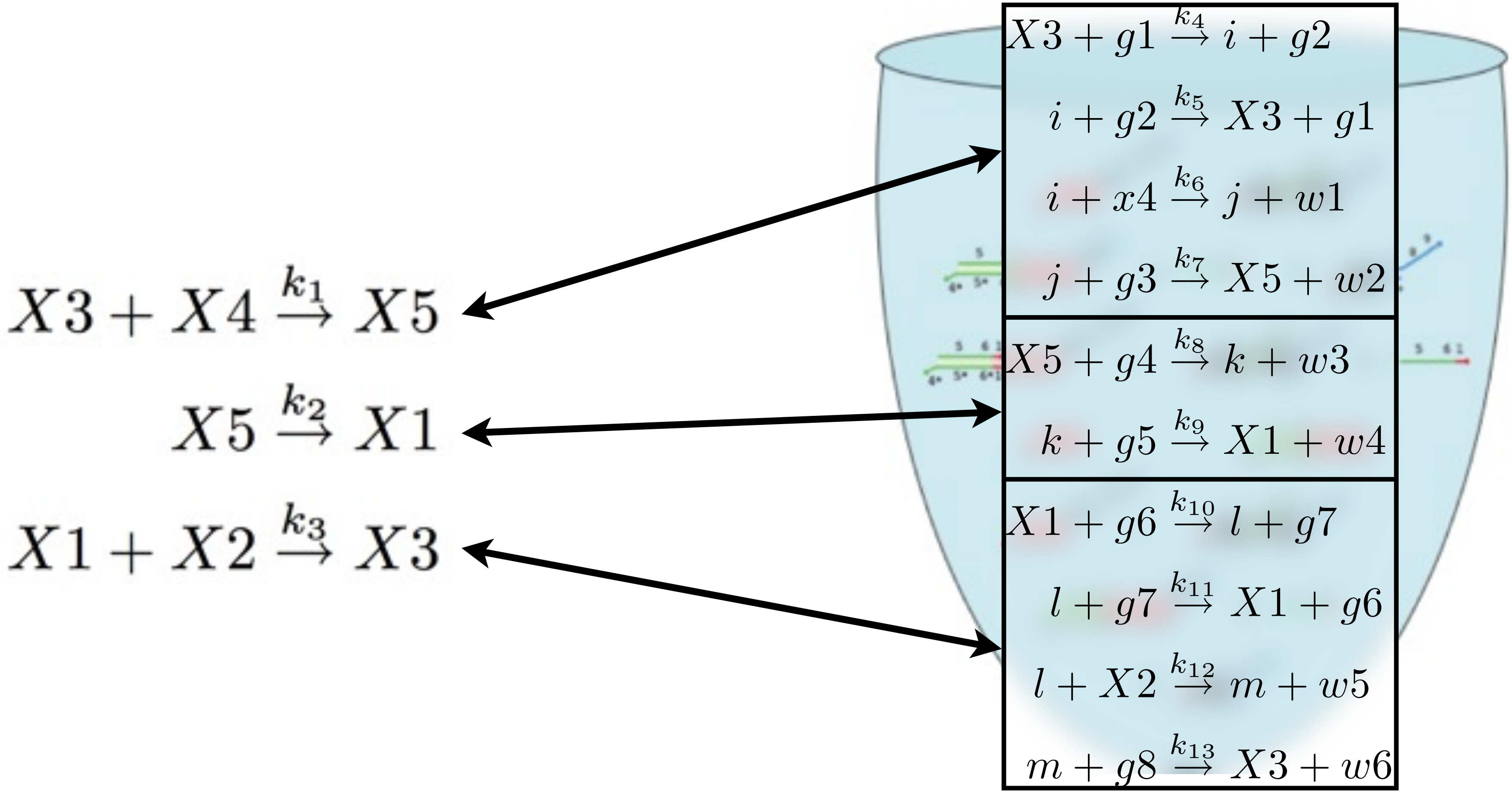
$$X3 + X4 \xrightarrow{k_1} X5$$

$$X5 \xrightarrow{k_2} X1$$

$$X1 + X2 \xrightarrow{k_3} X3$$



Every goal reaction corresponds to a set of implementation reactions



$$X3 + X4 \xrightarrow{k_1} X5$$

$$X5 \xrightarrow{k_2} X1$$

$$X1 + X2 \xrightarrow{k_3} X3$$

$$X3 + g1 \xrightarrow{k_4} i + g2$$

$$i + g2 \xrightarrow{k_5} X3 + g1$$

$$i + x4 \xrightarrow{k_6} j + w1$$

$$j + g3 \xrightarrow{k_7} X5 + w2$$

$$X5 + g4 \xrightarrow{k_8} k + w3$$

$$k + g5 \xrightarrow{k_9} X1 + w4$$

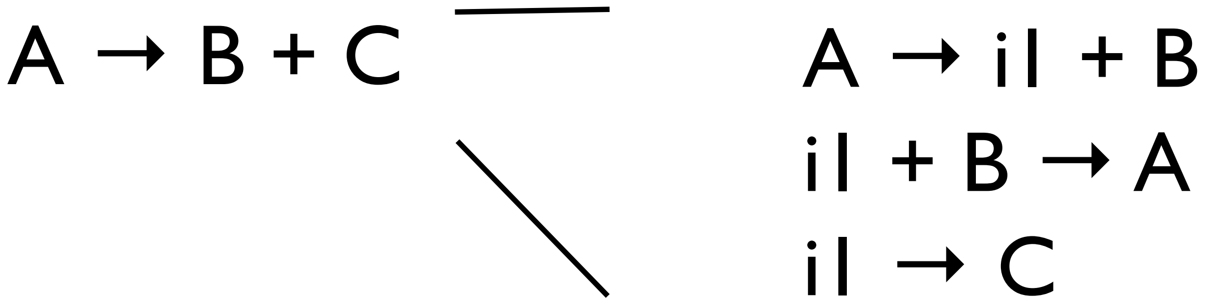
$$X1 + g6 \xrightarrow{k_{10}} l + g7$$

$$l + g7 \xrightarrow{k_{11}} X1 + g6$$

$$l + X2 \xrightarrow{k_{12}} m + w5$$

$$m + g8 \xrightarrow{k_{13}} X3 + w6$$

How can you tell that an implementation of a reaction is correct? Can be tricky!



Goal reactions

$A \rightarrow B + C$

Implementation

$A \rightarrow i1 + B$

$i1 + B \rightarrow A$

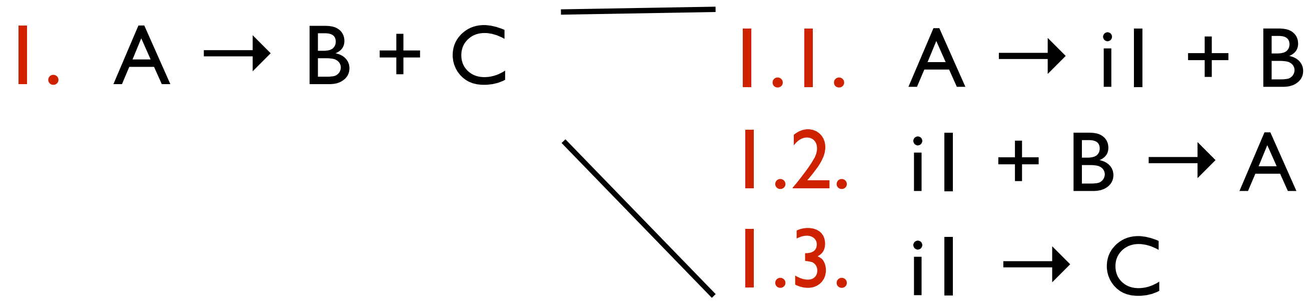
$i1 \rightarrow C$

[Shin, Thachuk, Winfree, VEMDP 2014]

How can you tell that an implementation of a reaction is correct? Can be tricky!

Goal reactions

Implementation



1. A → B + C

1.1. A → i1 + B

1.2. i1 + B → A

1.3. i1 → C

2. B + D → B + E

3. A + E → F

[Shin, Thachuk, Winfree, VEMDP 2014]

How can you tell that an implementation of a reaction is correct? Can be tricky!

Goal reactions

1. $A \rightarrow B + C$
2. $B + D \rightarrow B + E$
3. $A + E \rightarrow F$

Implementation

1.1. $A \rightarrow i1 + B$

1.2. $i1 + B \rightarrow A$

1.3. $i1 \rightarrow C$

Ex. Error

$\{1\ A, 1\ D\}$

[Shin, Thachuk, Winfree, VEMDP 2014]

How can you tell that an implementation of a reaction is correct? Can be tricky!

Goal reactions

1. $A \rightarrow B + C$

2. $B + D \rightarrow B + E$

3. $A + E \rightarrow F$

Implementation

1.1. $A \rightarrow i1 + B$

1.2. $i1 + B \rightarrow A$

1.3. $i1 \rightarrow C$

Ex. Error

$$\{1\ A, 1\ D\}$$

$$1.1 \Downarrow$$

$$\{1\ i1, 1\ B, 1\ D\}$$

$$2 \Downarrow$$

$$\{1\ i1, 1\ B, 1\ E\}$$

$$1.2 \Downarrow$$

$$\{1\ A, 1\ E\}$$

$$3 \Downarrow$$

$$\{1\ F\}$$

[Shin, Thachuk, Winfree, VEMDP 2014]

Acknowledgements

Adam Arkin
Luca Cardelli
Ho-Lin Chen
Yuan-Jyue Chen
Matthew Cook
David Doty
Manoj Gopalkrishnan

Lulu Qian
Paul W.K. Rothemund
Georg Seelig
Niranjan Srinivas
Erik Winfree
Damien Woods
David Zhang

UCSF Center for Systems & Synth Bio
Winfree group (Caltech)
Seelig group (UW)

NSF MPP grant

DISC'14

CI Fellows

UCSF
center for systems
& synthetic biology
an NIGMS national systems biology center

THE UNIVERSITY OF
TEXAS
AT AUSTIN