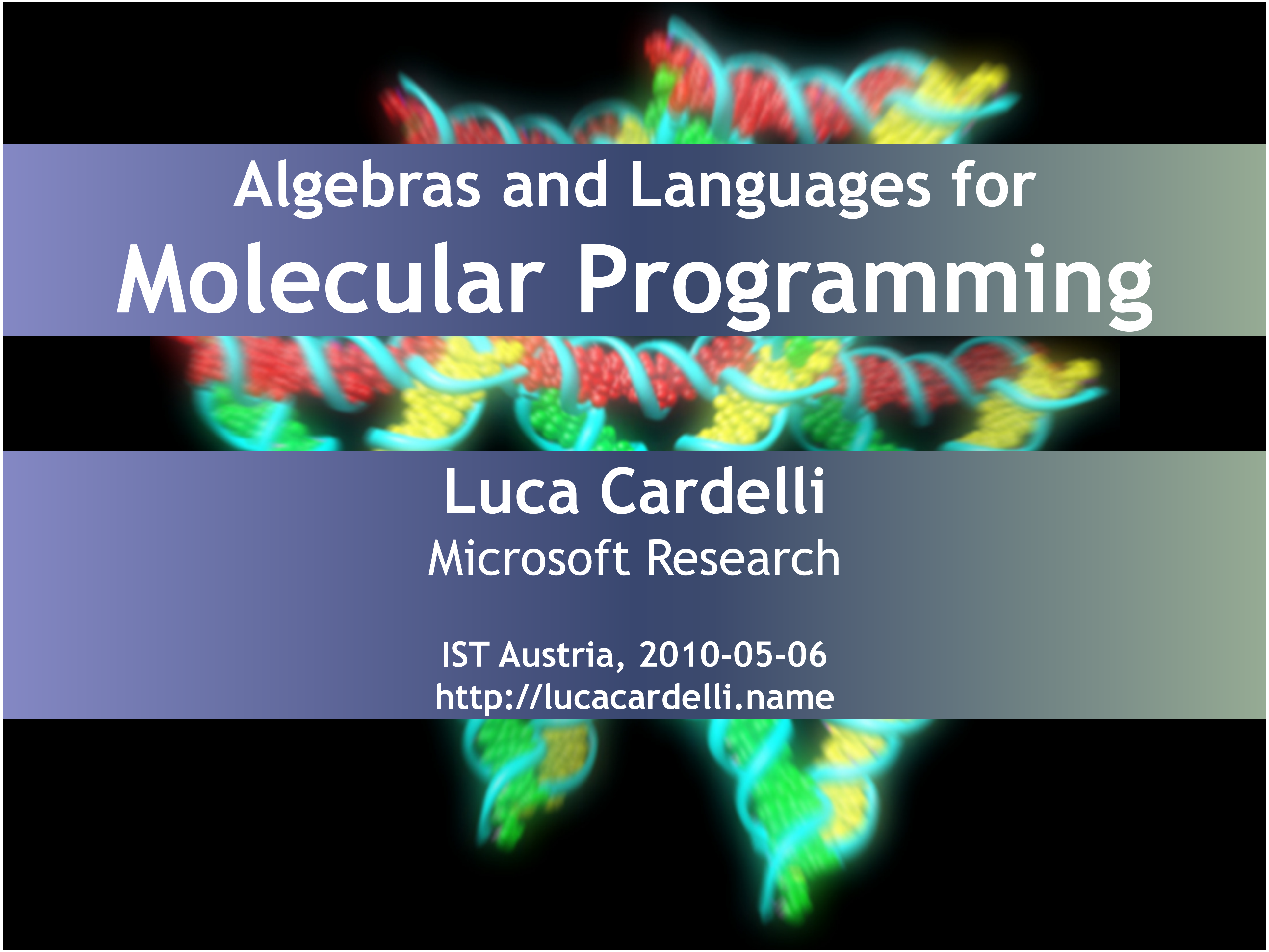


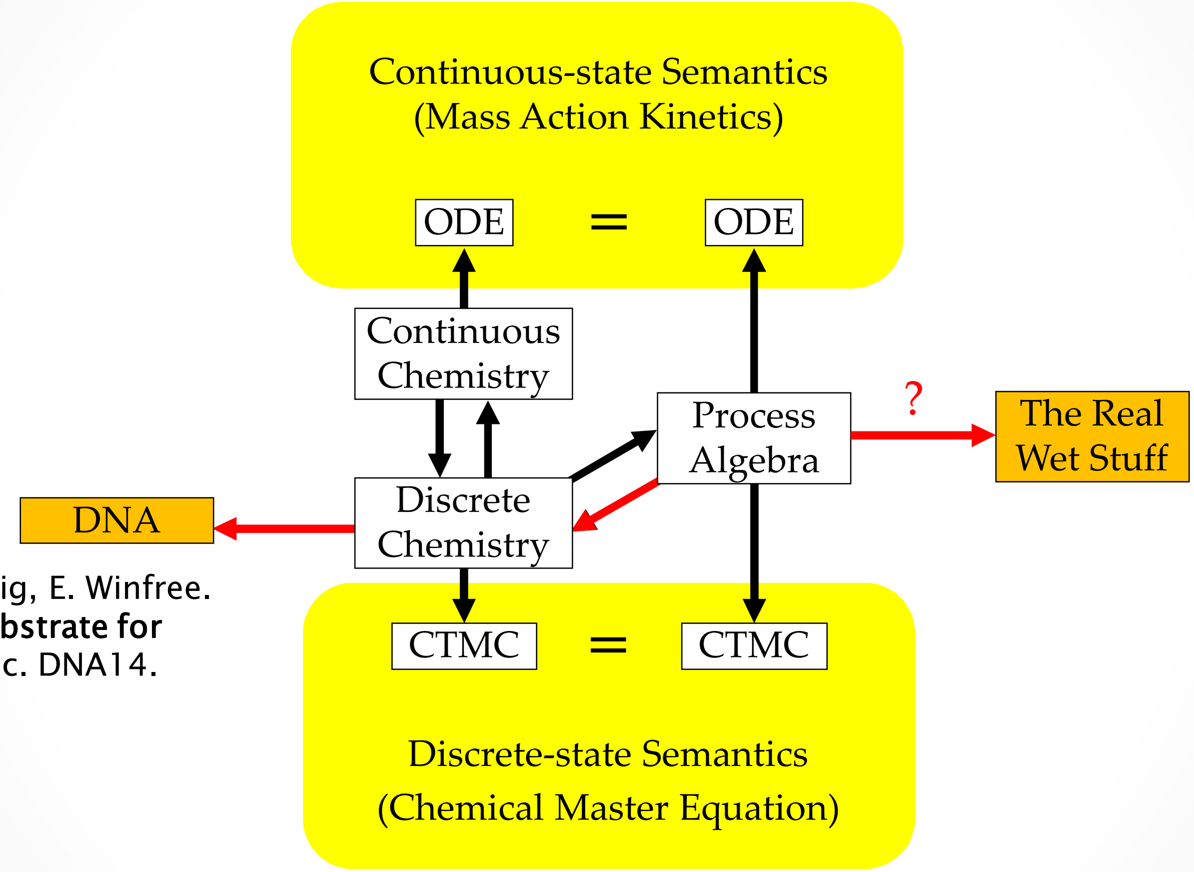
Algebras and Languages for
Molecular Programming

Luca Cardelli

Microsoft Research

IST Austria, 2010-05-06
http://lucacardelli.name

Molecules as Automata



Continuous-state Semantics
(Mass Action Kinetics)

ODE = ODE

Continuous Chemistry

Process Algebra

?

The Real Wet Stuff

DNA

Discrete Chemistry

CTMC = CTMC

Discrete-state Semantics
(Chemical Master Equation)

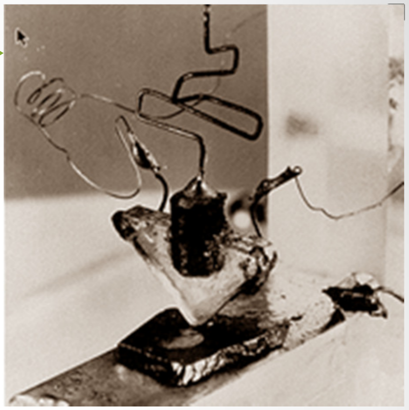
D. Soloveichik, G. Seelig, E. Winfree. **DNA as a Universal Substrate for Chemical Kinetics.** Proc. DNA14.

L. Cardelli: “On Process Rate Semantics” (TCS)

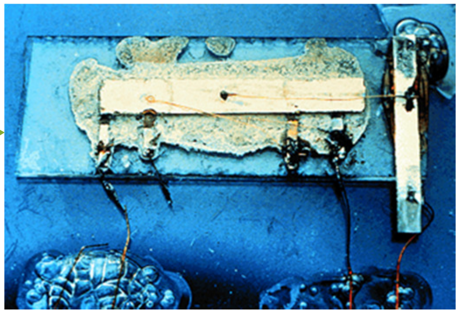
L. Cardelli: “A Process Algebra Master Equation” (QEST’07)

Smaller and Smaller

Dec. 23, 1947. John Bardeen and Walter Brattain show the first working transistor.

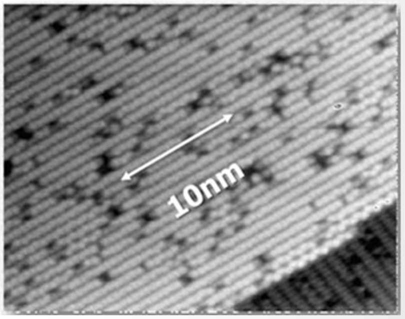


Sep. 1958. Jack Kilby builds the first integrated circuit.



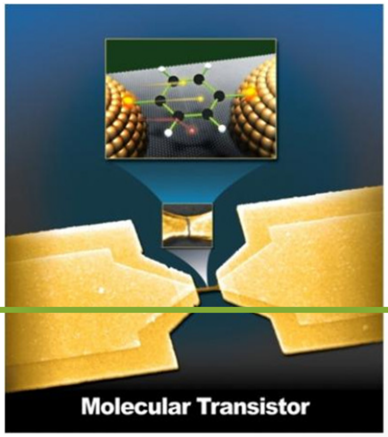
50 years later

Jan. 2010. Intel and Micron announce 25nm NAND flash.



Scanning tunneling microscope image of a silicon surface showing 10nm is ~20 atoms across

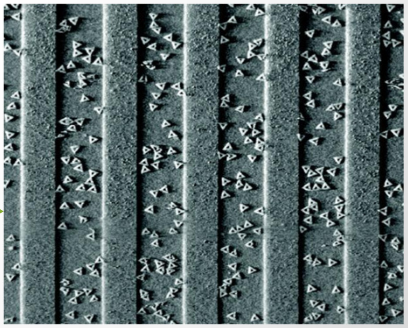
Dec. 24, 2009. Working transistor made of a single molecule.



Observation of molecular orbital gating. *Nature*, 2009; 462 (7276): 1039

<10 iterations of Moore's Law left!

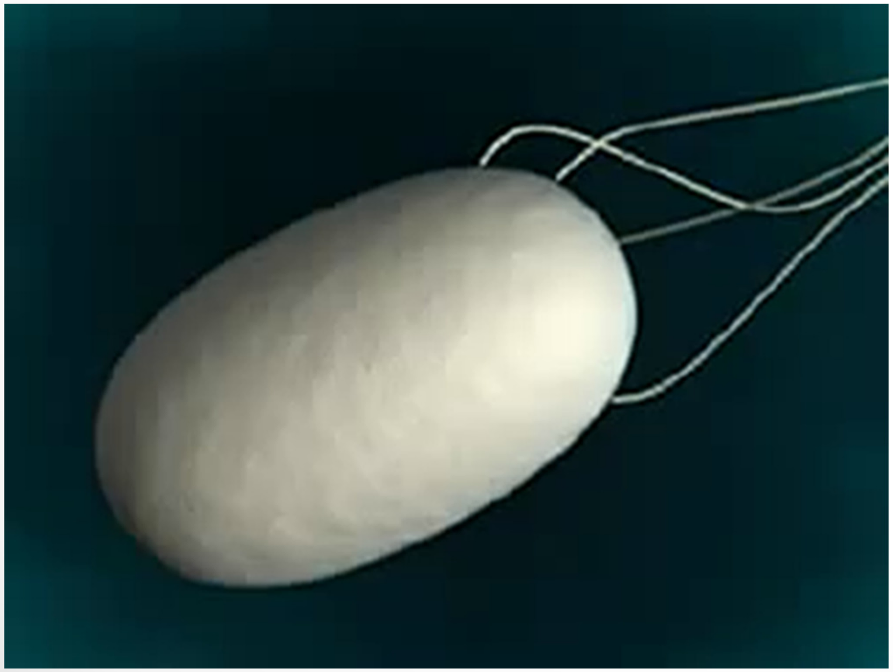
The race is on for *molecular scale integrated circuits*.



Placement and orientation of individual DNA shapes on lithographically patterned surfaces. Nature Nanotechnology 4, 557 – 561 (2009).

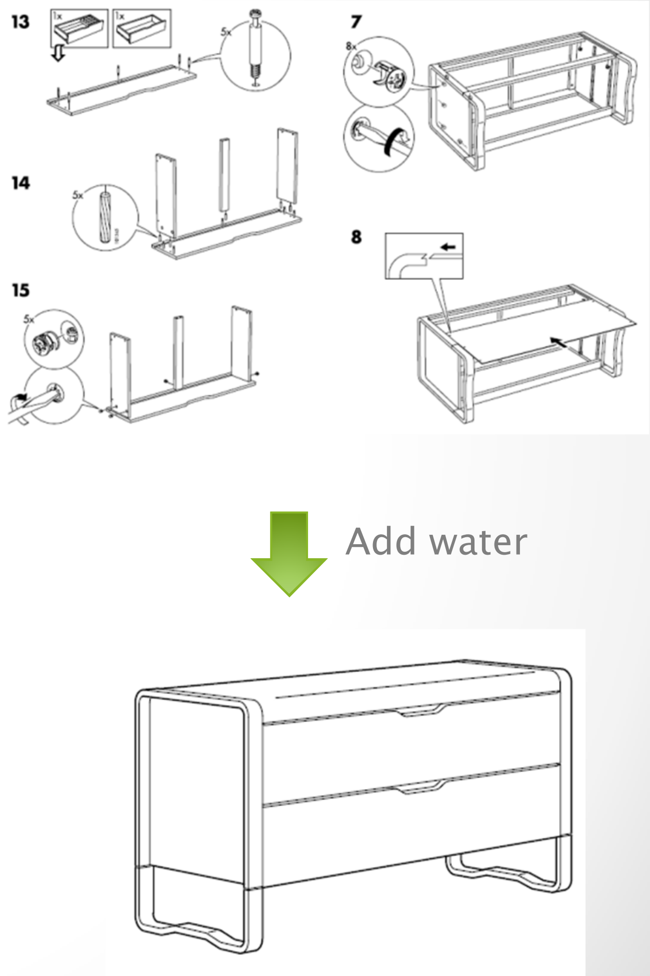
Building The Small*est* Things

- How do we build structures that are by definition smaller than your tools?
- Basic answer: you can't. Structures (and tools) should build themselves!
- By *programmed self-assembly*.



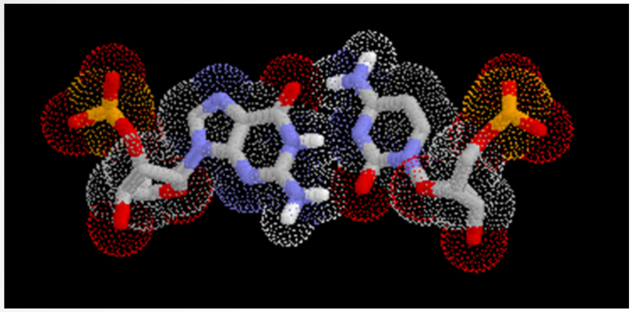
Molecular IKEA

- Nature can self-assemble.
  Can we?
- *“Dear IKEA, please send me a chest of drawers that assembles itself.”*
- We need a magical material where the pieces are pre-programmed to fit into to each other.
- At the molecular scale many such materials exist; let’s pick one…



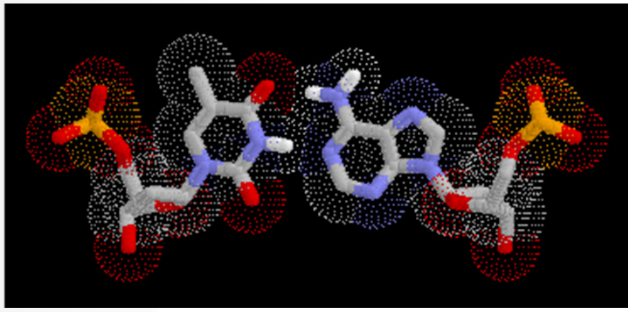
Add water

DNA



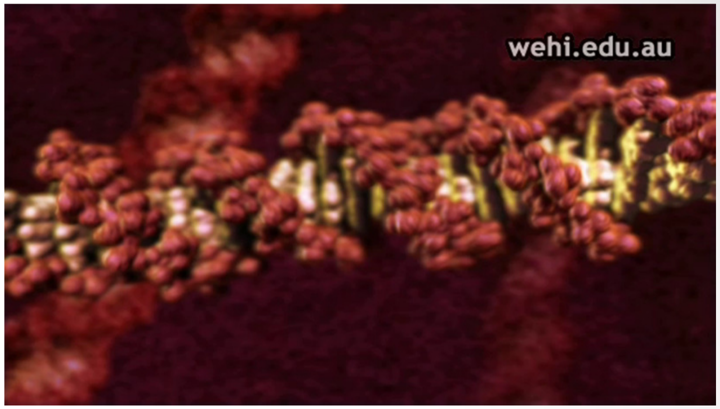
GC Base Pair

Guanine–Cytosine



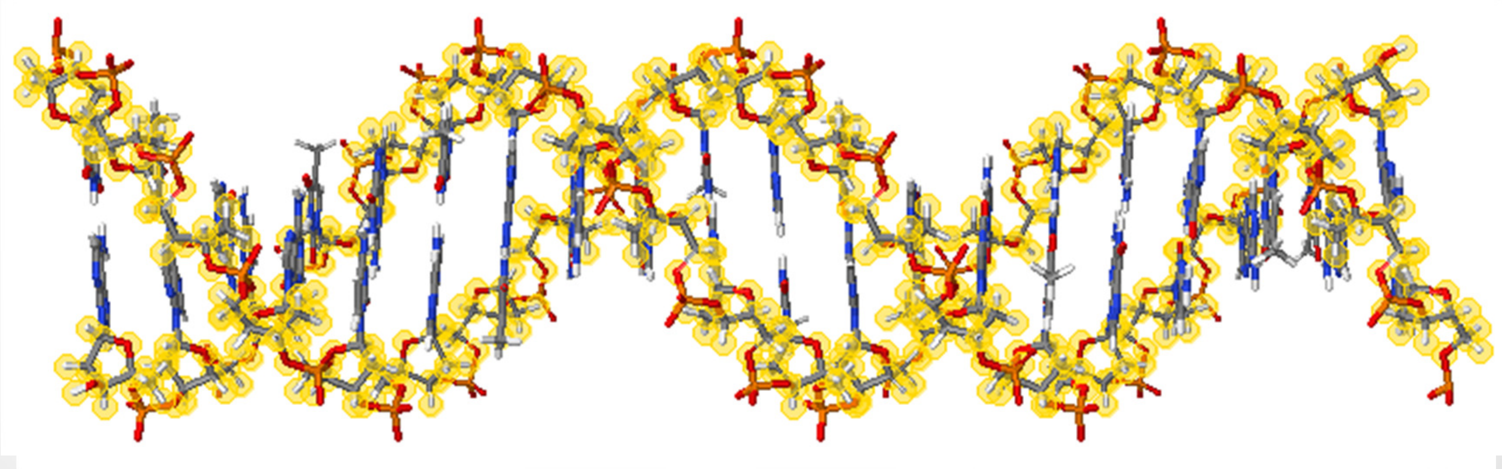
TA Base Pair

Thymine–Adenine



[Interactive DNA Tutorial](http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)

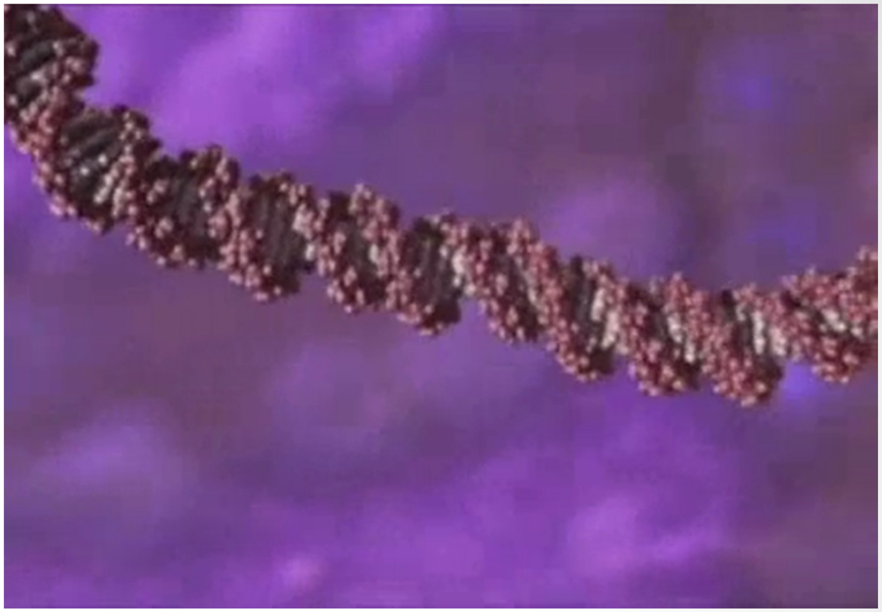
(http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)



Sequence of Base Pairs (GACT alphabet)

Robust, and *Long*

- DNA in each human cell:
  - 3 billion base pairs
  - 2 meters long, 2nm thick
  - folded into a 6μm ball
  - 750 MegaBytes
- A huge amount for a cell
  - Every time a cell replicates it has to copy *2 meters of DNA* reliably.
  - To get a feeling for the scale disparity, compute:
- DNA in human body
  - 10 trillion cells
  - 133 Astronomical Units long
  - 7.5 OctaBytes
- DNA in human population
  - 20 million light years long

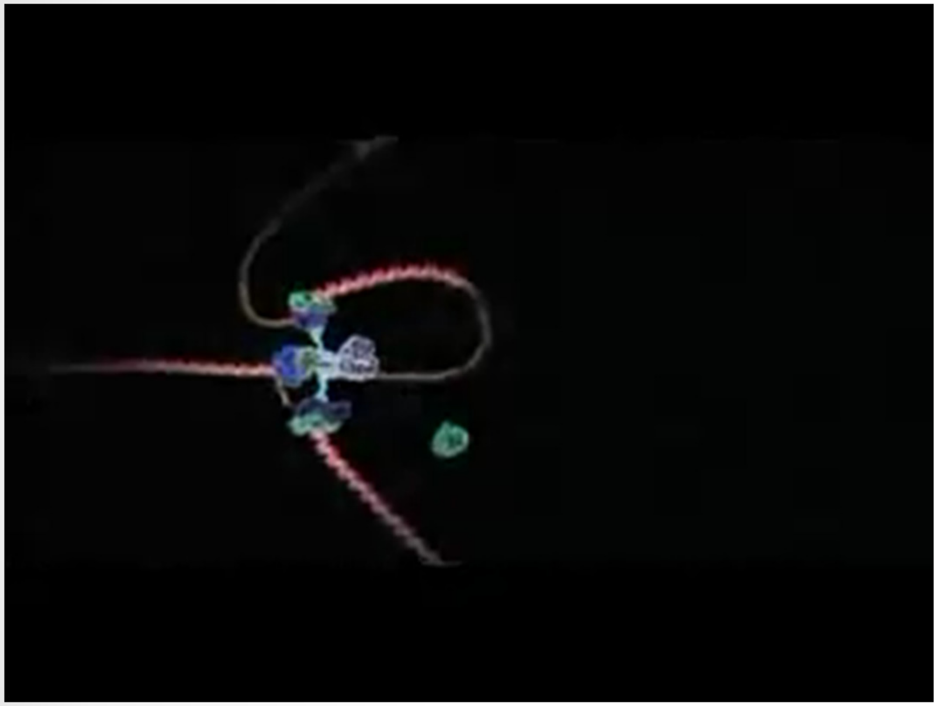


DNA wrapping into chromosomes

Andromeda Galaxy
2.5 million light years

Zipping Along

- DNA can support structural and computational complexity.



DNA replication in *real time*

In Humans: 50 nucleotides/second
Whole genome in a few hours (with parallel processing)

In Bacteria: 1000 nucleotides/second
(higher error rate)

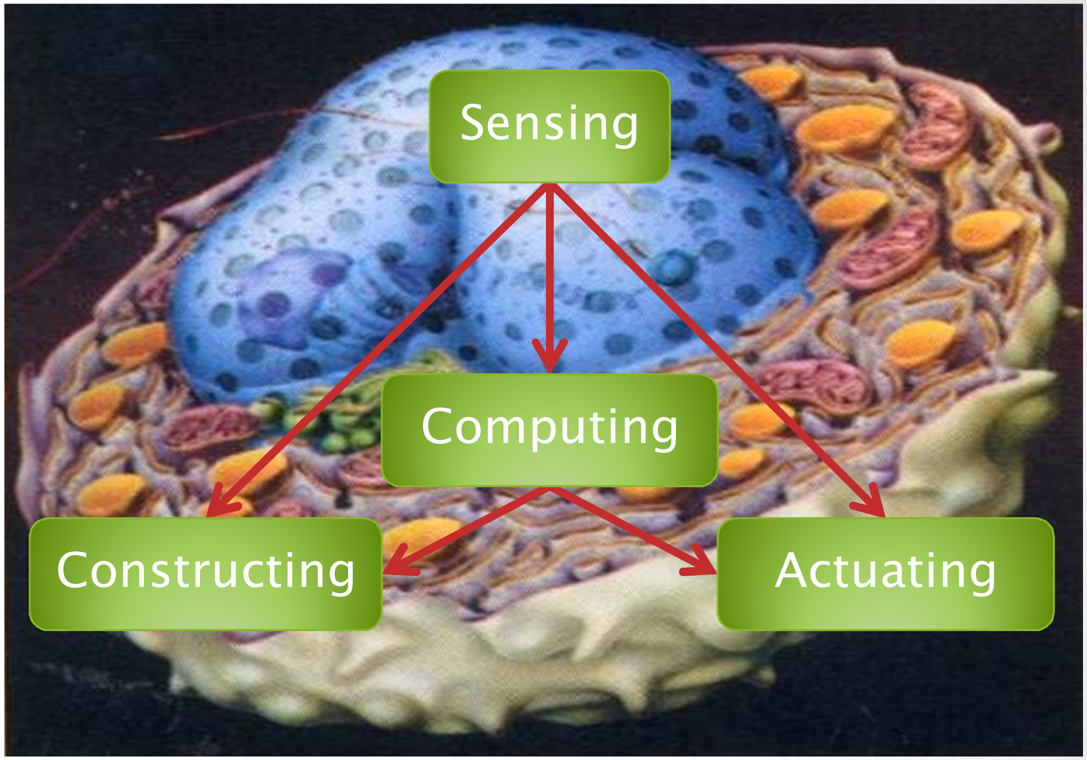
DNA transcription in *real time*

RNA polymerase II: 15-30 base/second

Drew Berry
http://www.wehi.edu.au/wehi-tv

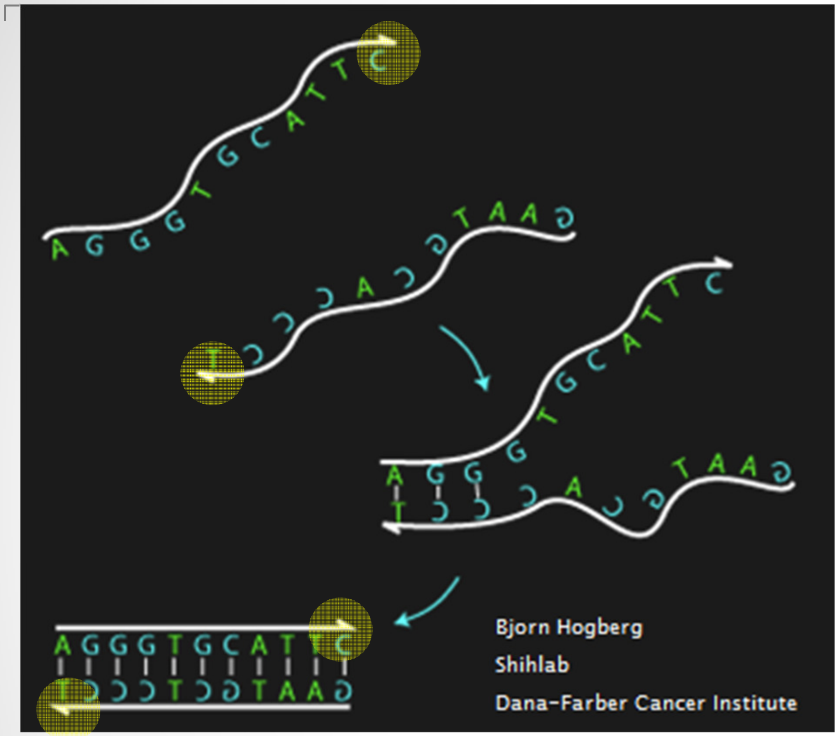
Nanoscale Engineering

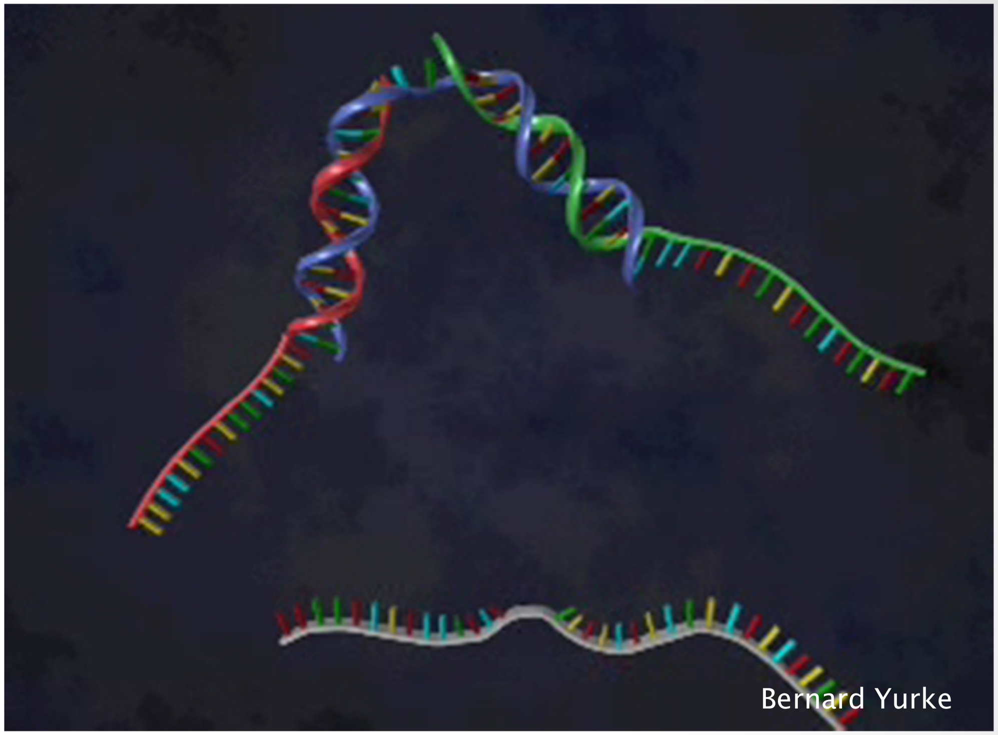
- Sensing
  - Reacting to forces
  - Binding to molecules
- Actuating
  - Releasing molecules
  - Producing forces
- Constructing
  - Chassis
  - Growth
- Computing
  - Signal Processing
  - Decision Making



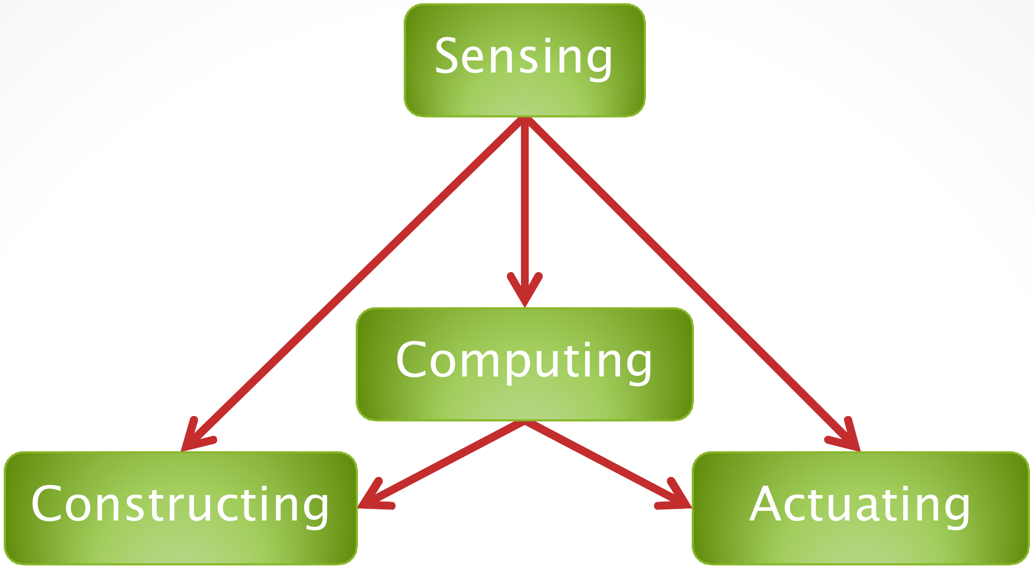
Nucleic Acids can do all this.
And interface to biology.

Hybridization





- Strands with **opposite orientation** and **complementary base pairs** stick to each other (Watson-Crick duality).
- This is all we are going to use
  - We are not going to exploit DNA replication, transcription, translation, **restriction and ligation enzymes**, etc., which enable other classes of tricks.



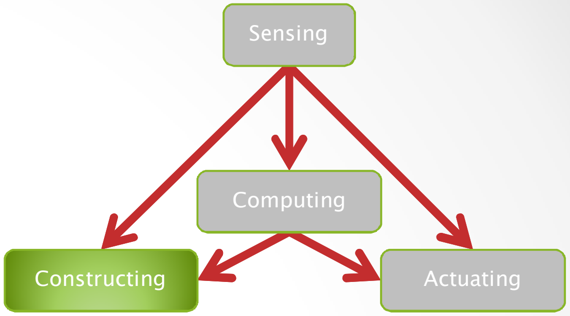
Sensing

Computing

Constructing

Actuating

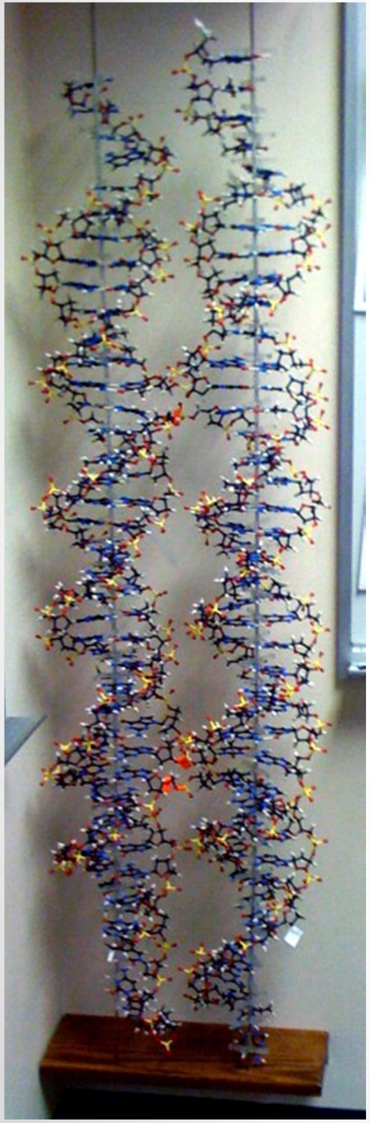
Hybridization Tricks

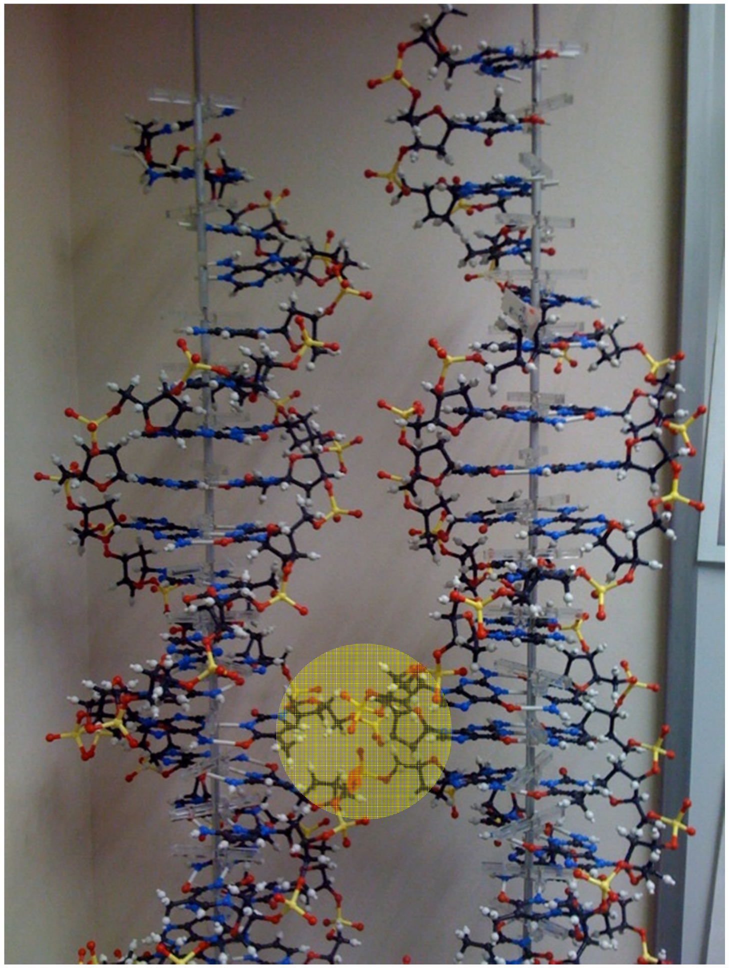


Constructing

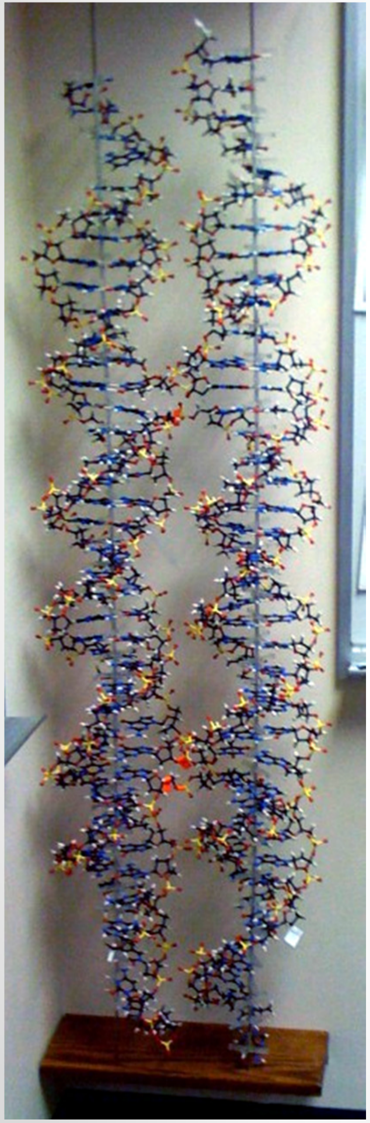
• • •

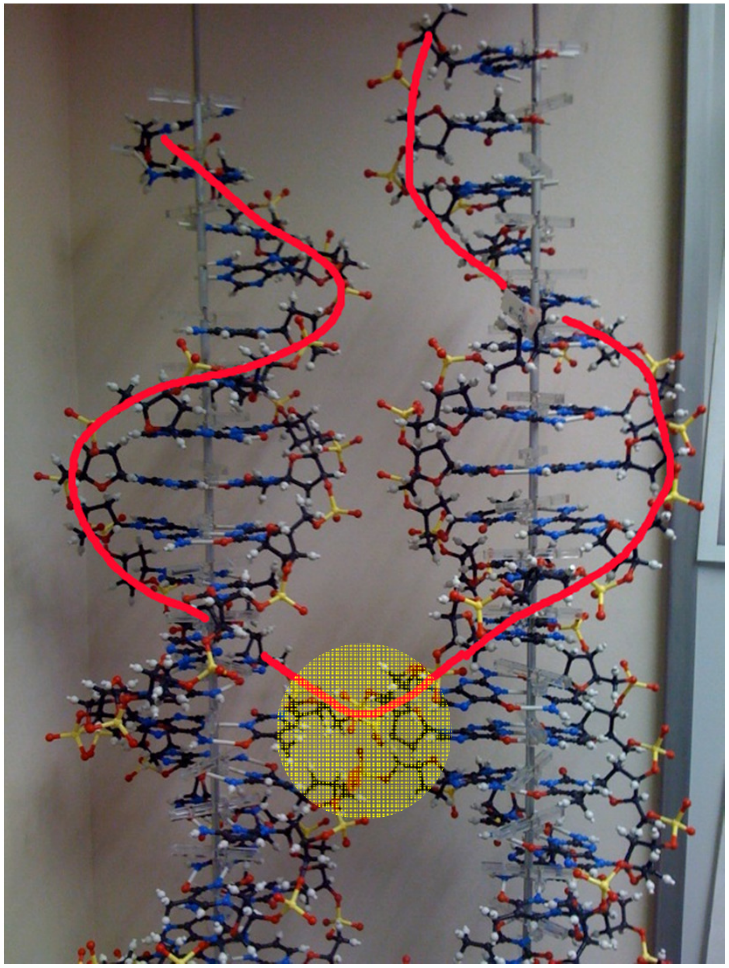
Crosslinking



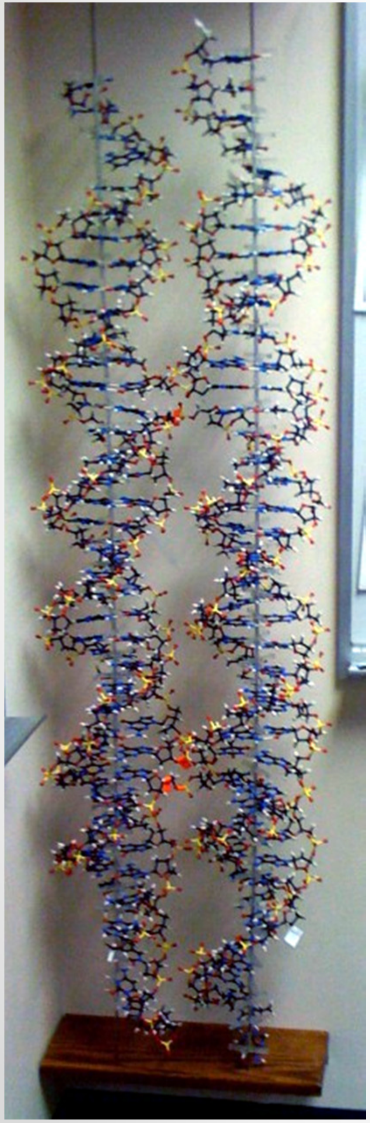


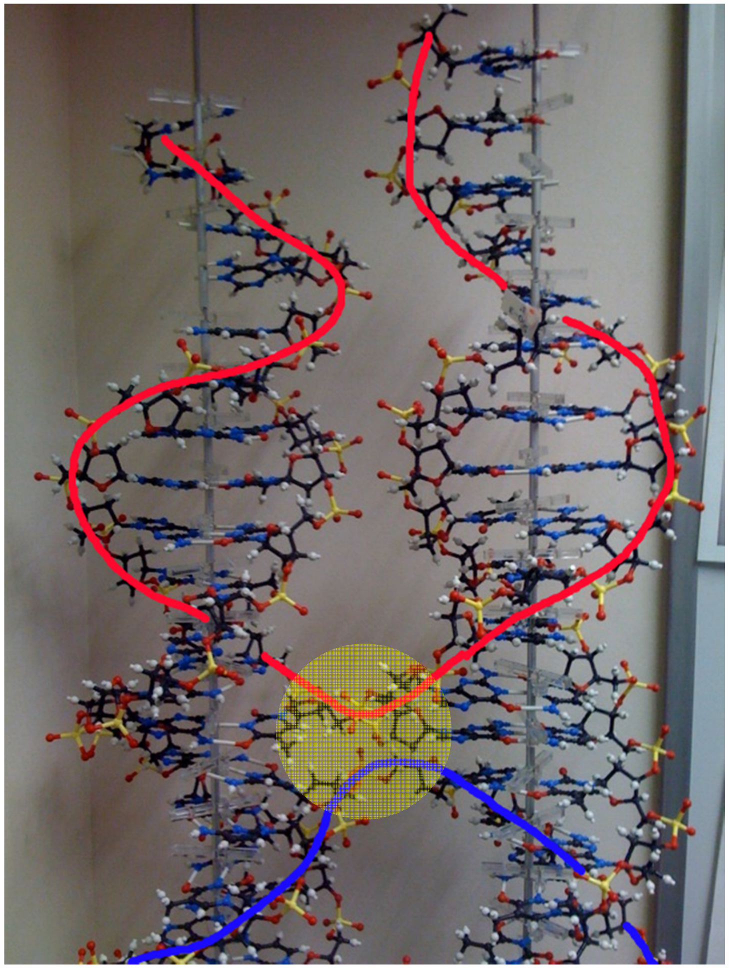
Crosslinking



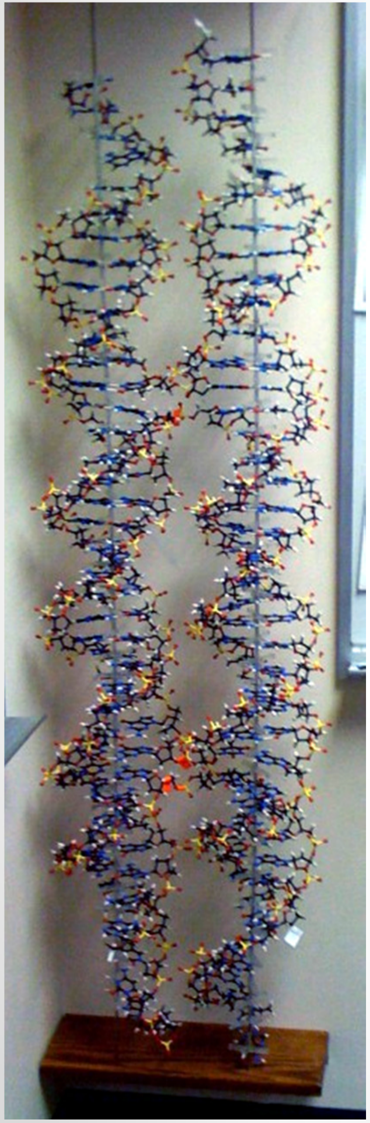


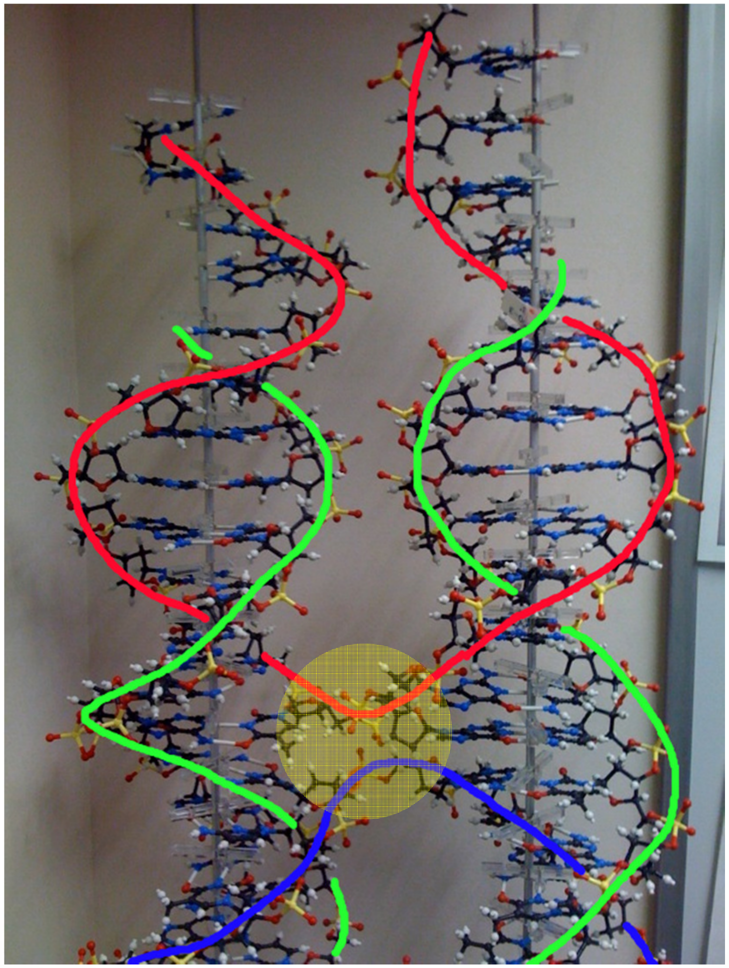
Crosslinking



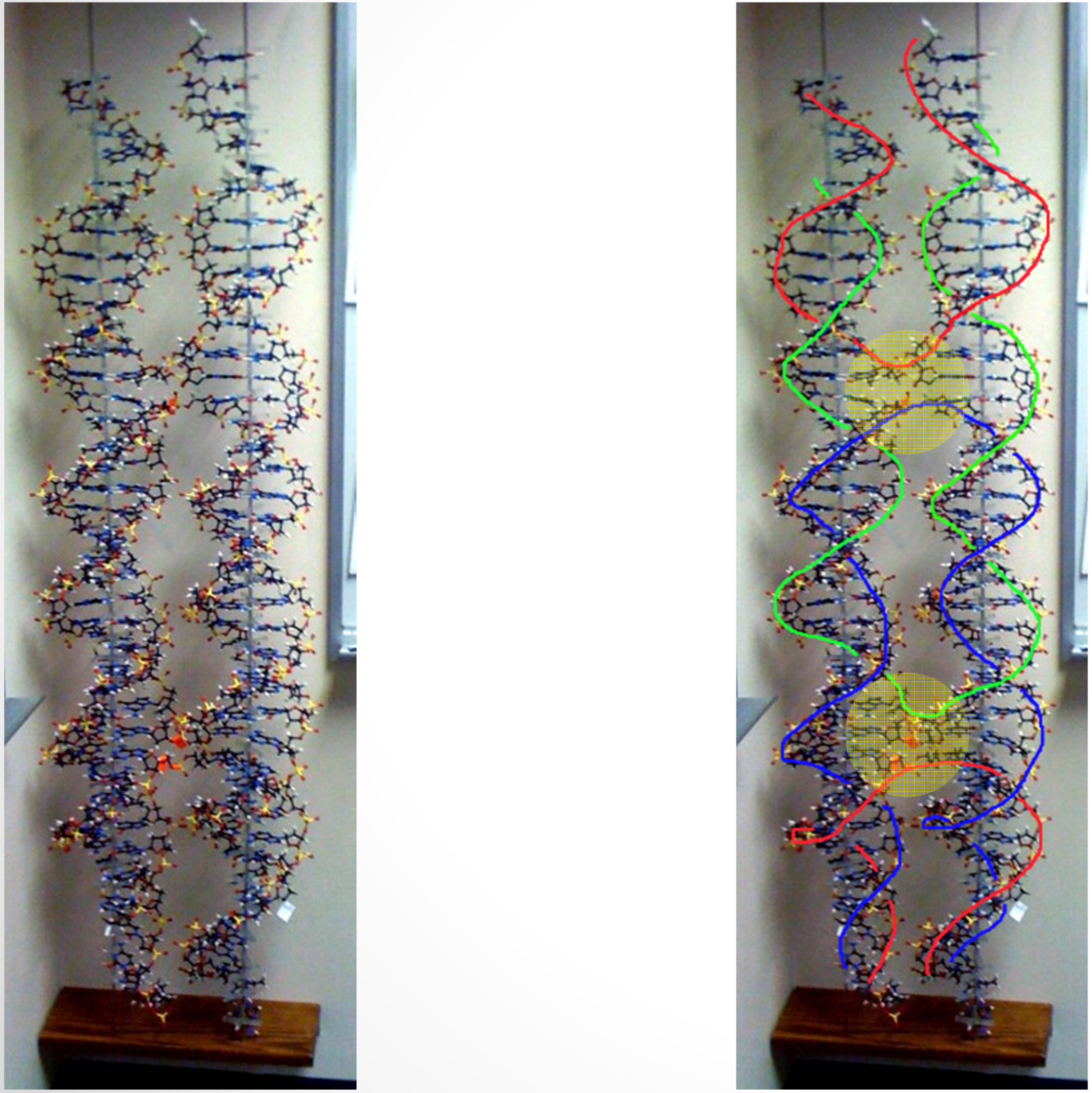


Crosslinking

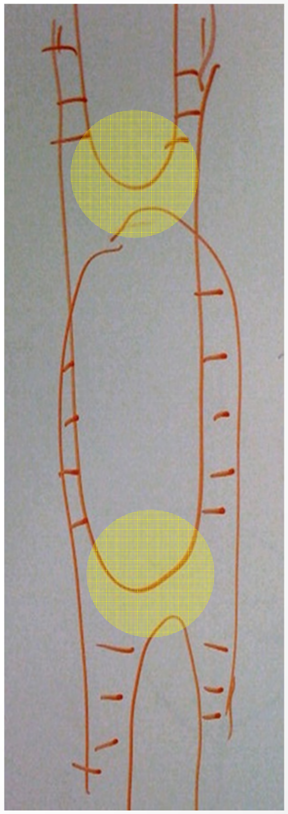




Crosslinking

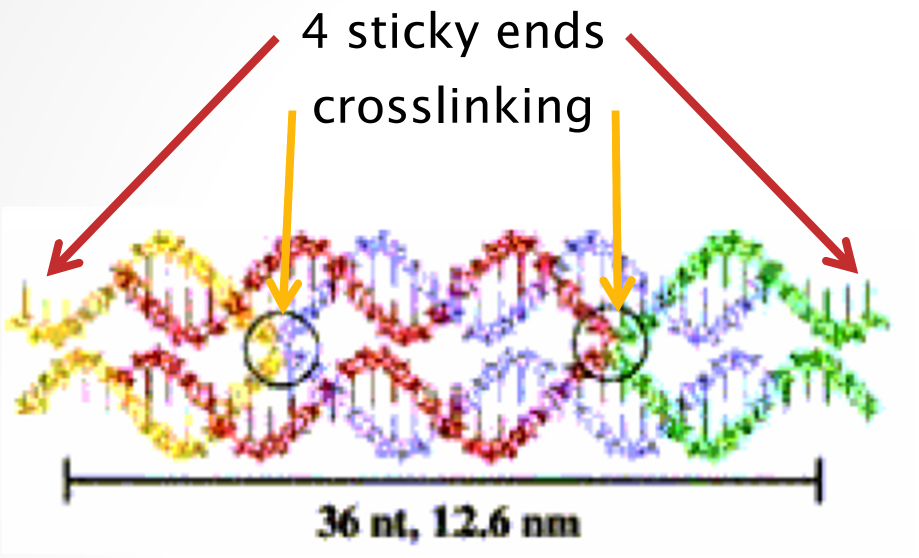


In nature, crosslinking is deadly (blocks DNA replication).

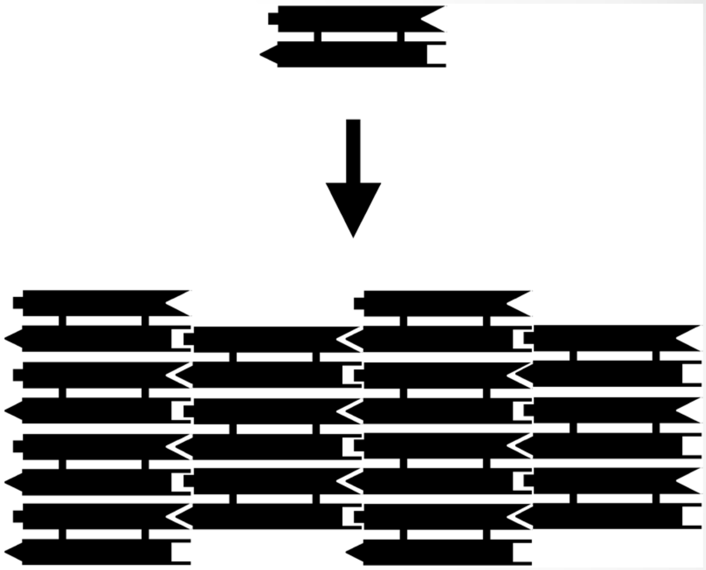


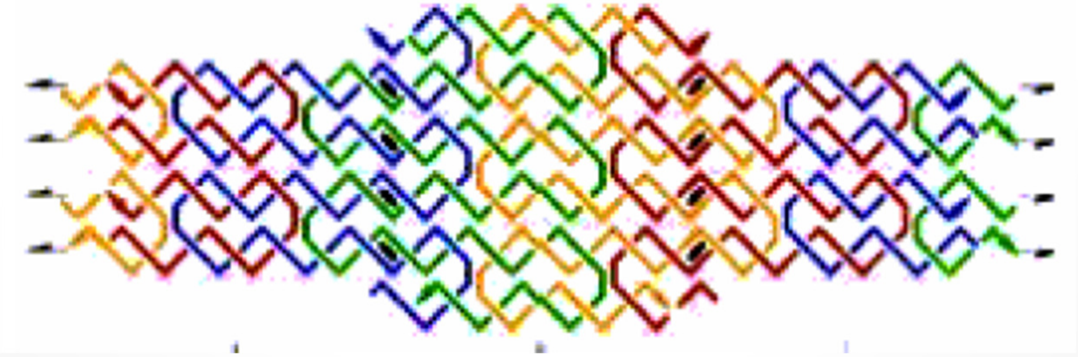
In engineering, crosslinking is the key to using DNA as a construction material.

DNA Tiling


4 sticky ends

crosslinking

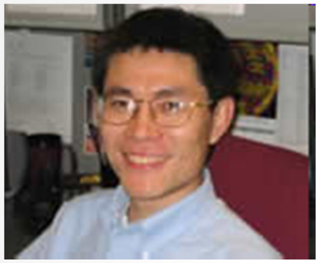




Construction and manipulation of DNA tiles in free space

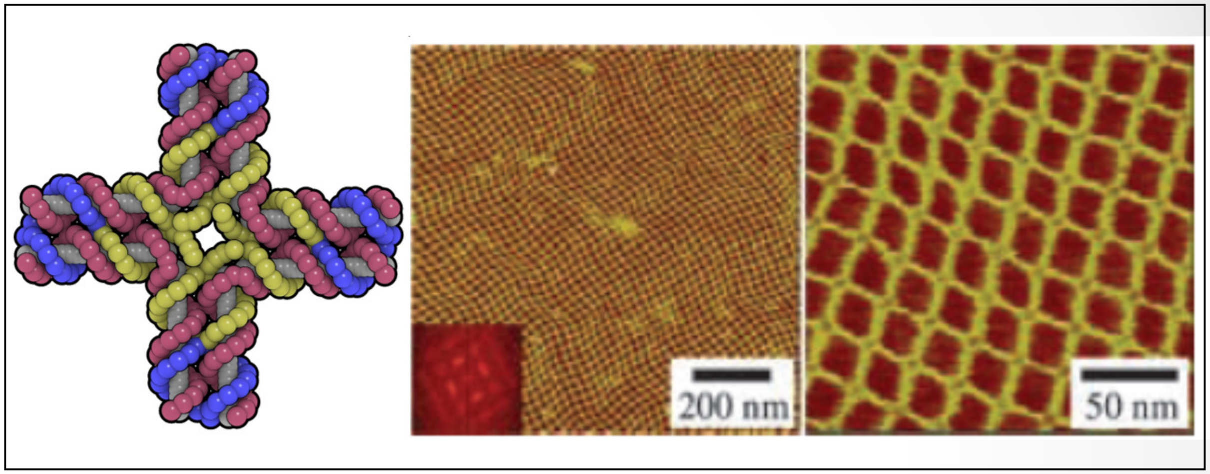
Pankhudi

2D DNA Lattices

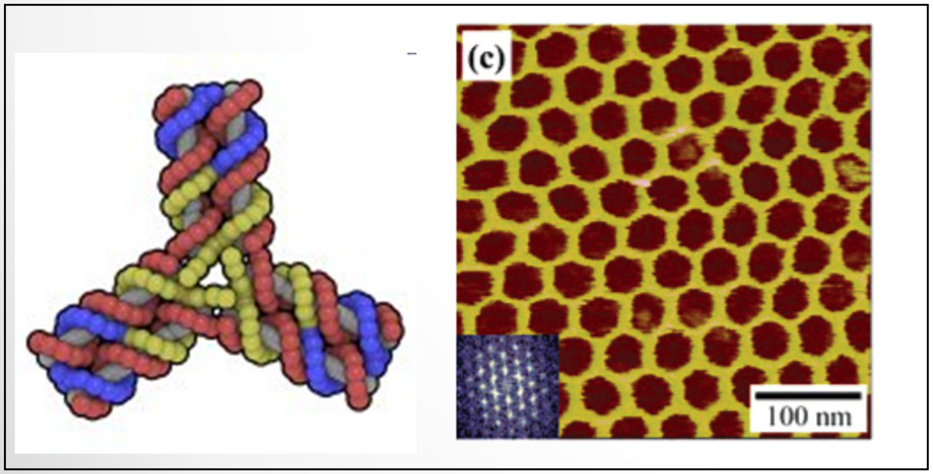


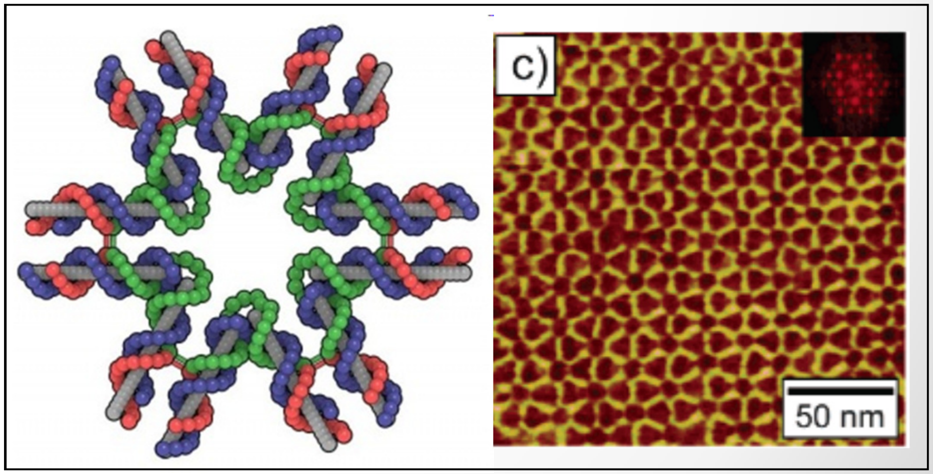
Chengde Mao

Purdue University, USA



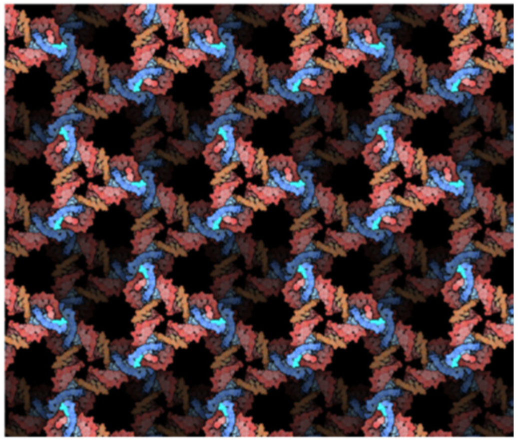
N-point Stars





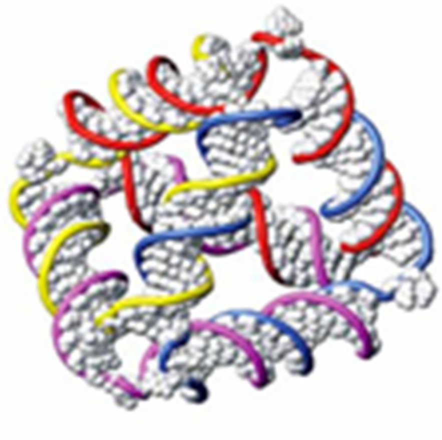
3D DNA Structures

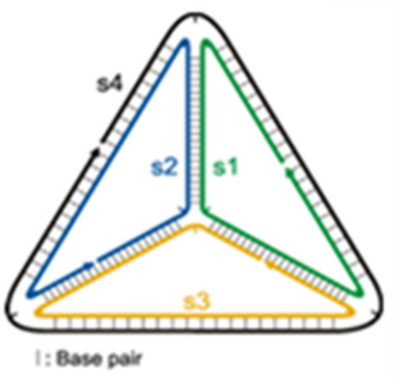
Ned Seeman
NYU



3D Cyrstal

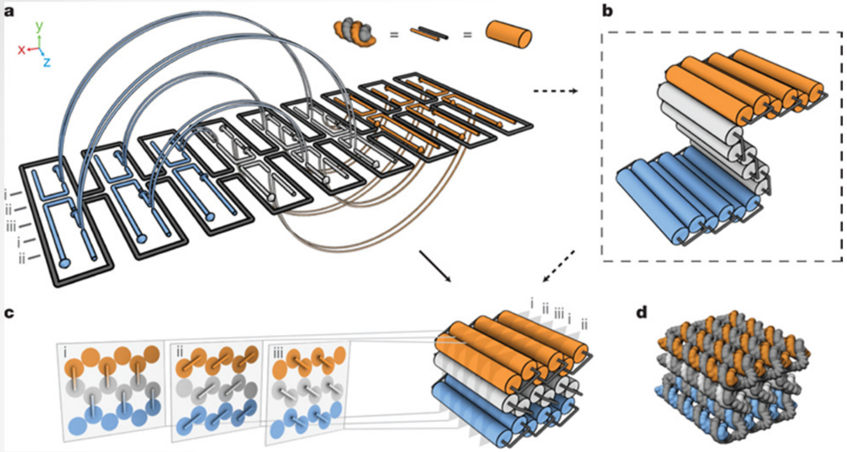
AndrewTuberfield
Oxford

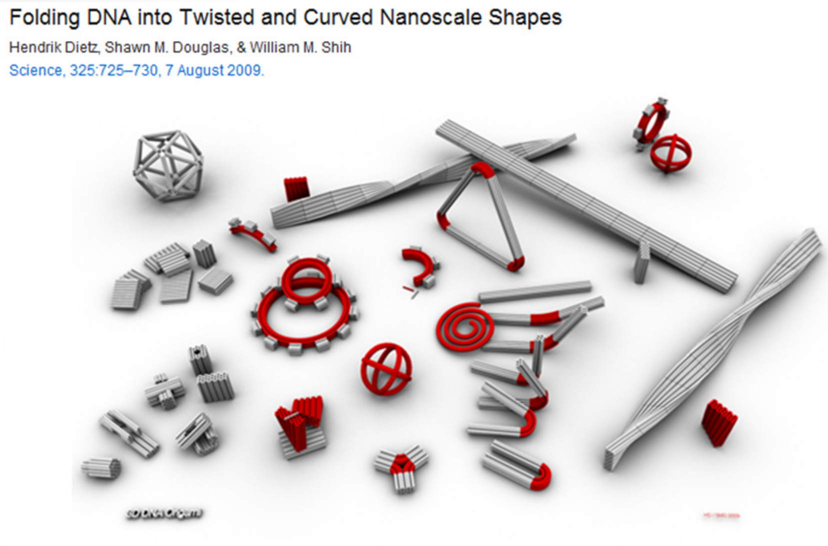




Tetrahedron

CADnano

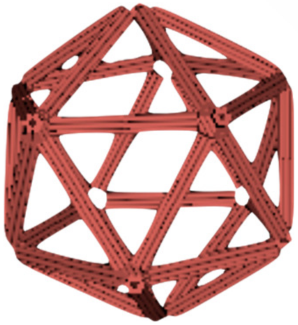


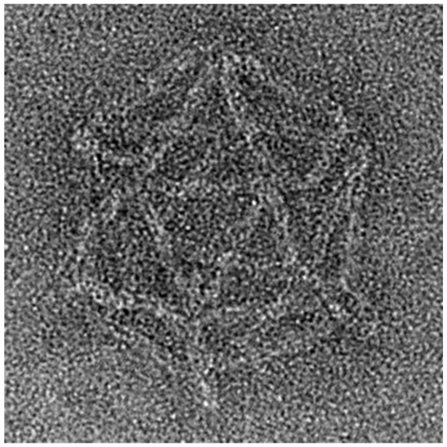


Folding DNA into Twisted and Curved Nanoscale Shapes

Hendrik Dietz, Shawn M. Douglas, & William M. Shih

Science, 325:725–730, 7 August 2009.



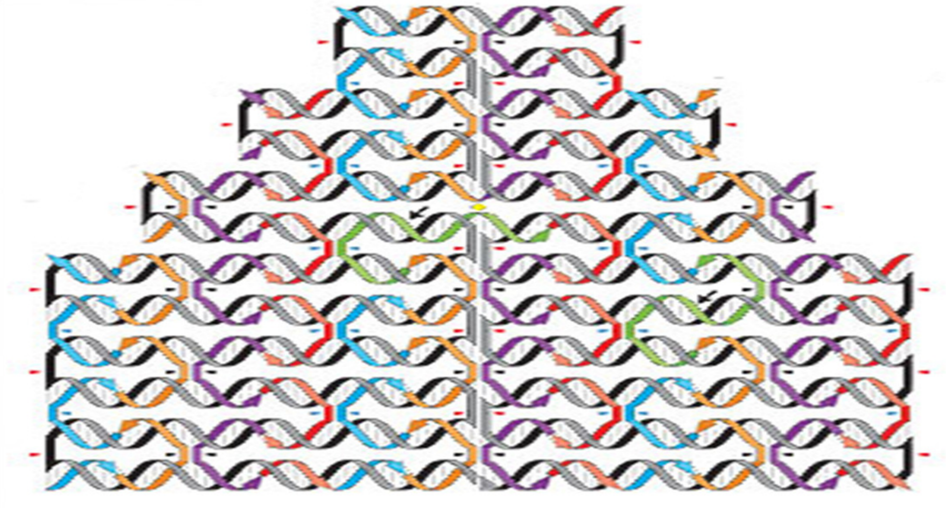


William Shih
Harvard

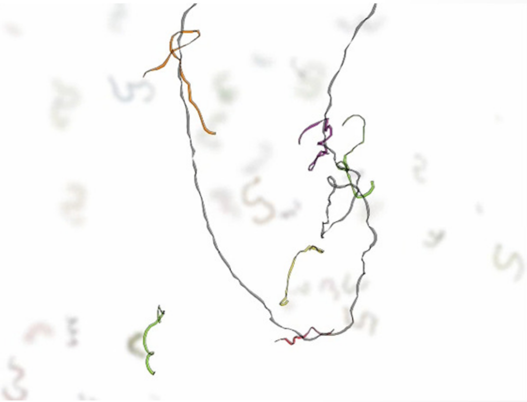
S.M. Douglas, H. Dietz, T. Liedl, B. Högberg, F. Graf and W. M. Shih
Self-assembly of DNA into nanoscale three-dimensional shapes, Nature (2009)

DNA Origami

- *Folding* long (7000bp) naturally occurring (viral) ssDNA
- By lots of short ‘staple’ strands that constrain it

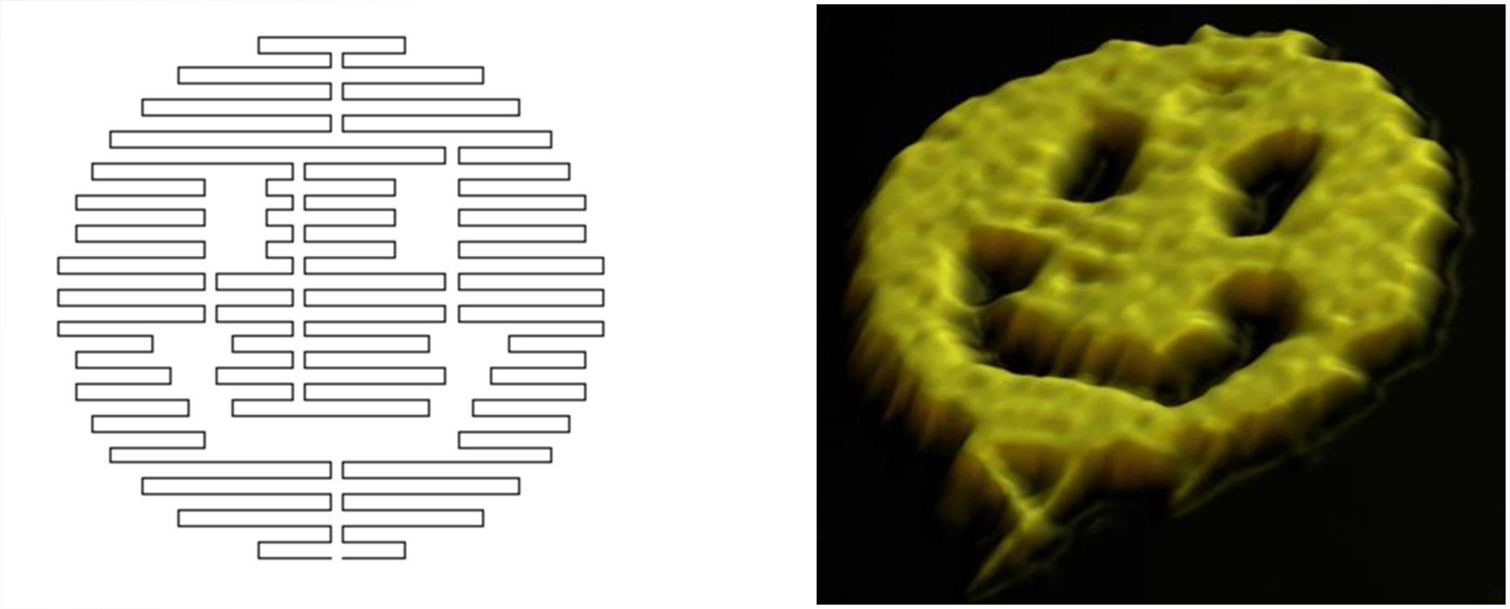


PWK Rothemund, *Nature* 440, 297 (2006)



Black: long viral strand
Color: short staple strands

DNA Origami



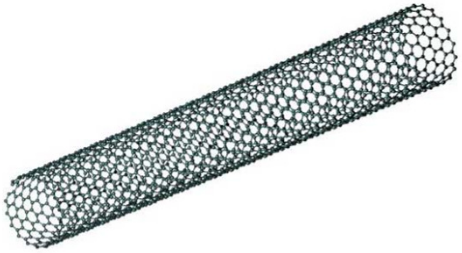
Paul Rothemund's "Disc with three holes" (2006)

Paul W K Rothemund
California Institute of Technology

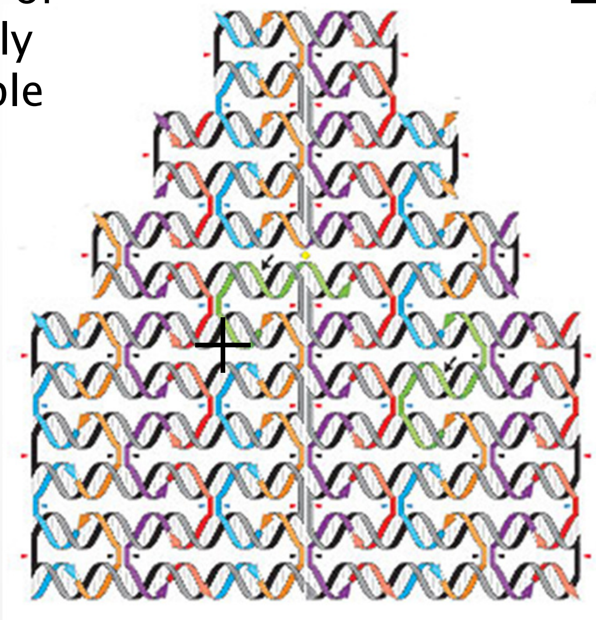
This means we can already self-assemble meso-scale structures.

DNA Circuit Boards

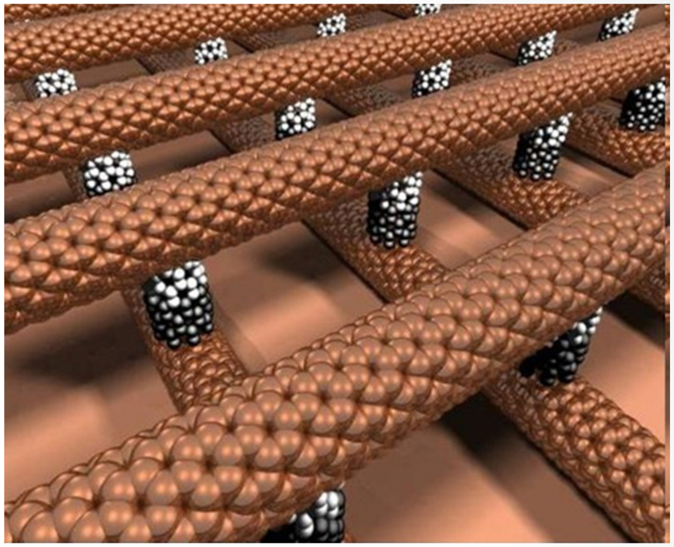
DNA-wrapped nanotubes



6 nm grid of individually addressable pixels

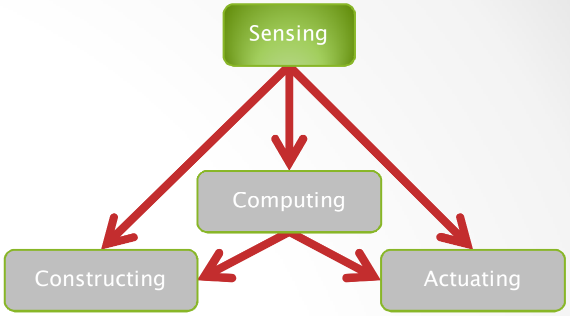


PWK Rothemund, *Nature* 440, 297 (2006)



European Nanoelectronics Initiative Advisory Council

"What we are really making are tiny DNA circuit boards that will be used to assemble other components."
Greg Wallraff, IBM

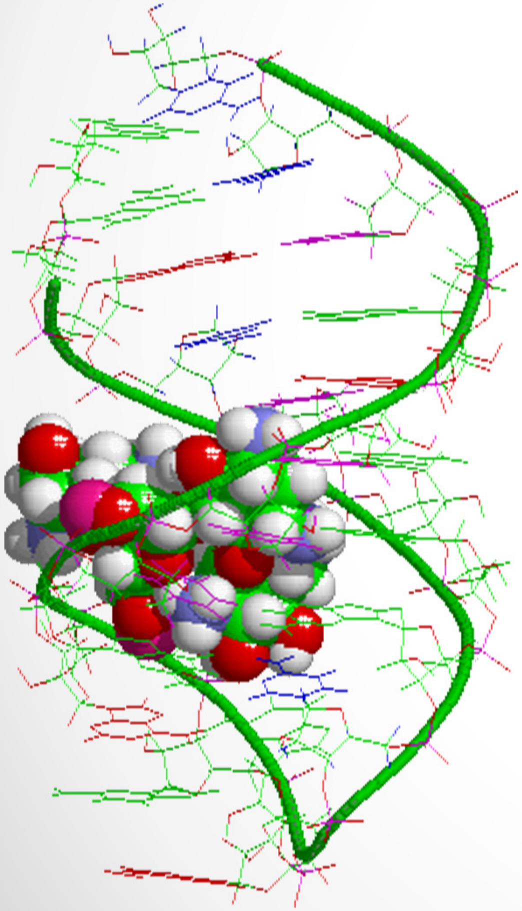


Sensing

• • •

Aptamers

- Artificially eveloved DNA molecules that stick to anything you like (highly selectively).



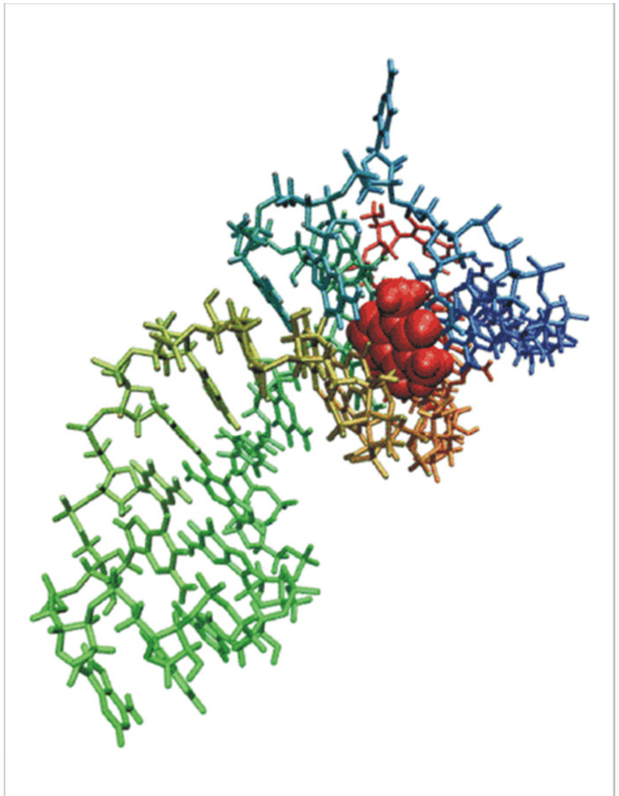
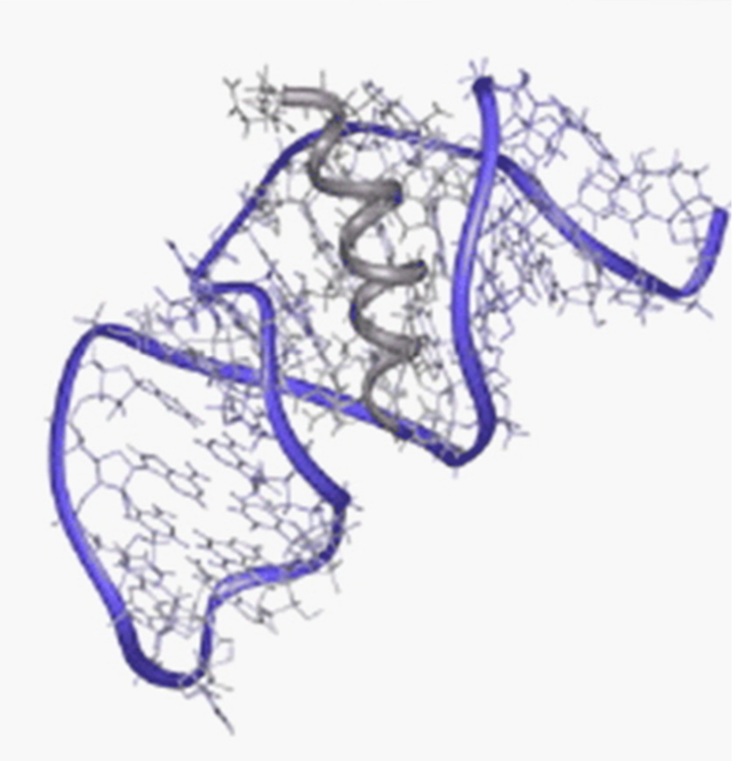


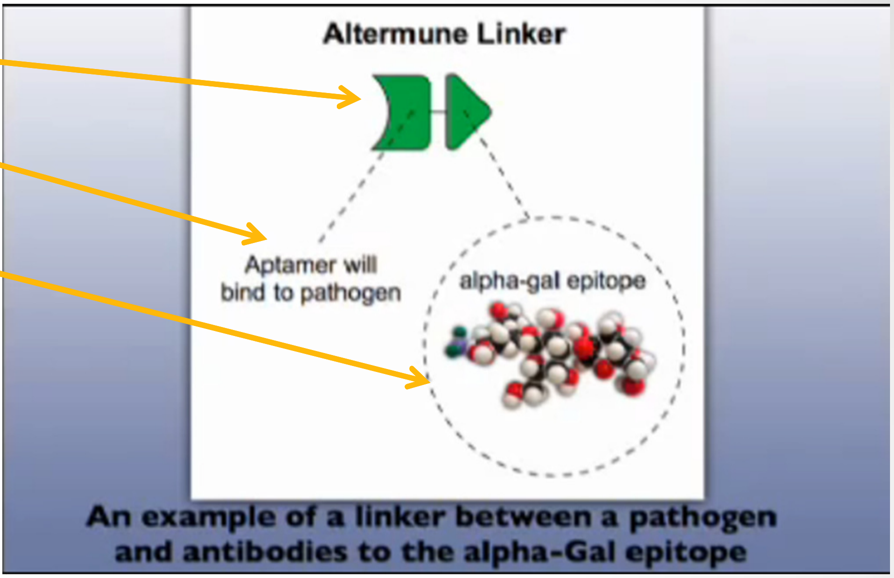
Fig.: RNA-aptamer - purine complex

PDB-ID: 1O15 generated by VMD 1.8.2



Pathogen Spotlights

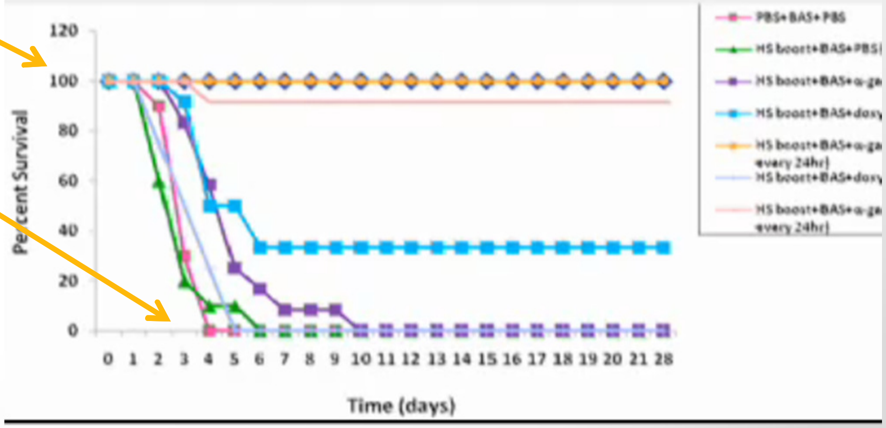
- DNA aptamer binds to:
  - A) a pathogen
  - B) a molecule our immune system already hates and immediately removes (eats) along with anything attached to it



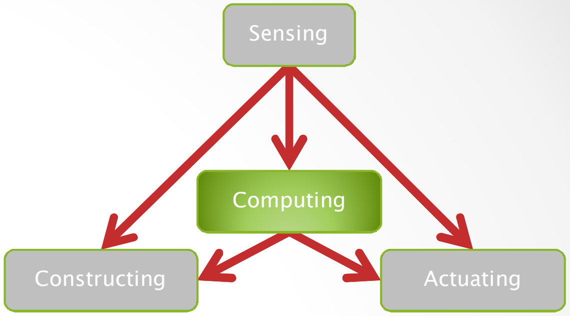
An example of a linker between a pathogen and antibodies to the alpha-Gal epitope

- Result: instant immunity
  - Mice poisoned with Anthrax plus aptamer (100% survival)
  - Mice poinsoned with Anthrax (not so good)

Survival Curve of A/J Mice Immunized with Human Serum, Challenged with BAS and Treated with α-gal PAA-12 Aptamer and Doxycycline



Kary Mullis (incidentally, also Nobel prize for inventing the Polymerase Chain Reaction)



Computing

• • •

Basic Steps

Compositionality

- Sensors and Actuators at the 'edge' of the system
    - They can use disparate kinds of inputs (sensors) and outputs (actuators)
- The 'kernel' of the system computes
    - Must use uniform inputs and outputs
- Compositionality in the kernel
    - Supporting 'arbitrary' computing complexity
    - The output of each computing components must be the same kind of 'signal' as the input
- sdf
    - If the inputs are voltages, the outputs must be voltages
    - If the inputs are DNA, the outputs must be DNA
- Central design question
    - What should our signals (not components!) be?
    - Then design components that manipulate those signals.

Rules of the Game

- Short complementary segments hybridize **reversibly**

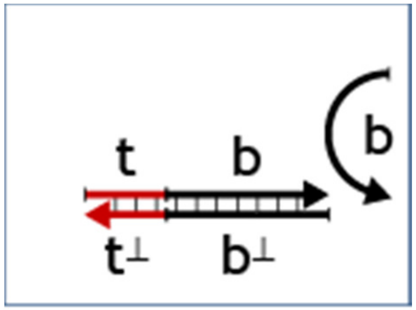
- Long complementary segments hybridize **irreversibly**

DNA Strand Displacement

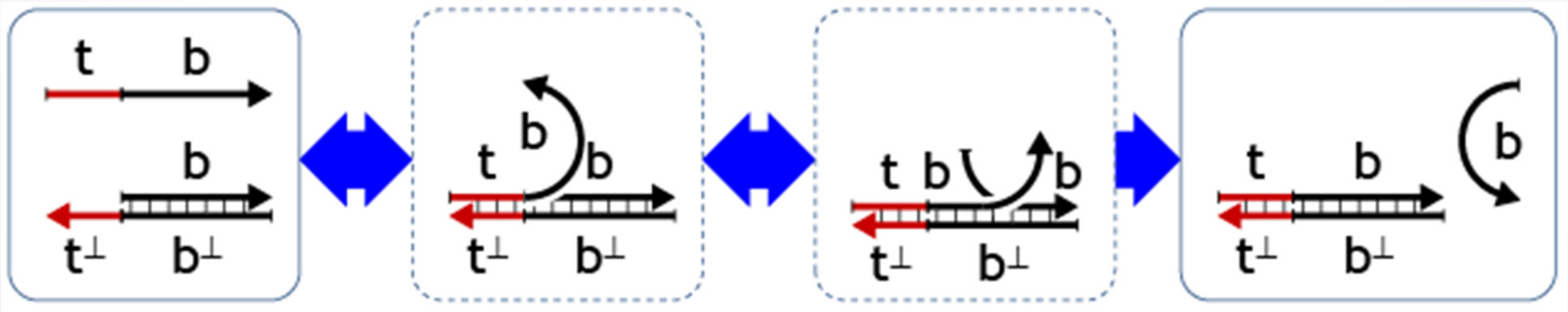
- Short strand (**toehold**): reversible binding
- Long strand (**body**): irreversible binding

Input

Gate



Output



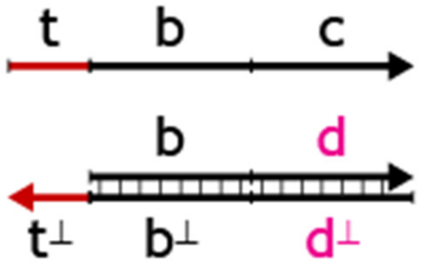
Random collision/ breakup

Random walk

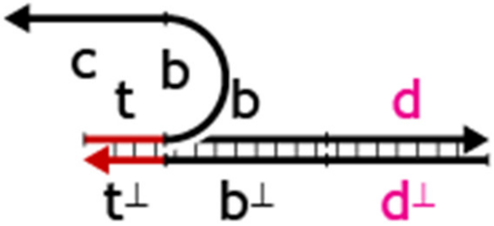
Entropy gain

Failed Strand Displacement

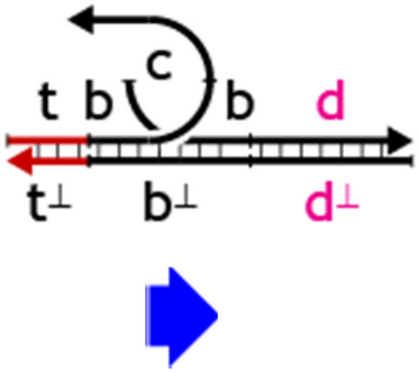
- What if the input does not match the gate?



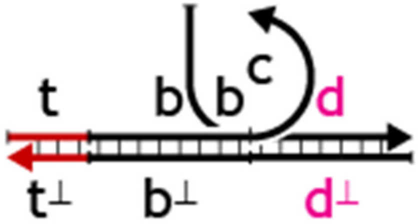
Failed Strand Displacement



Failed Strand Displacement

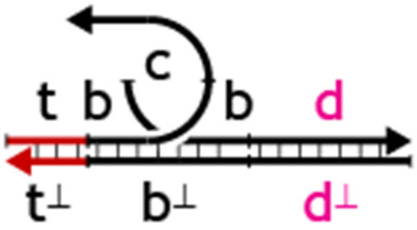


Failed Strand Displacement

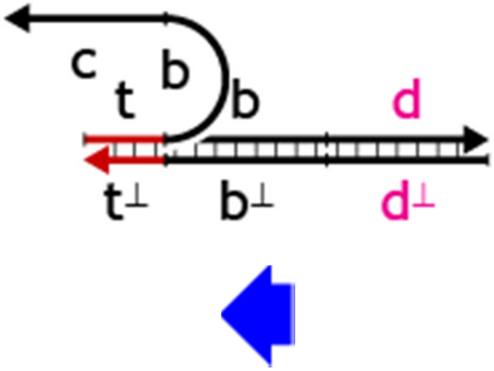


?

Failed Strand Displacement

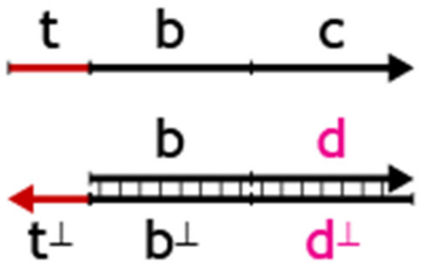


Failed Strand Displacement

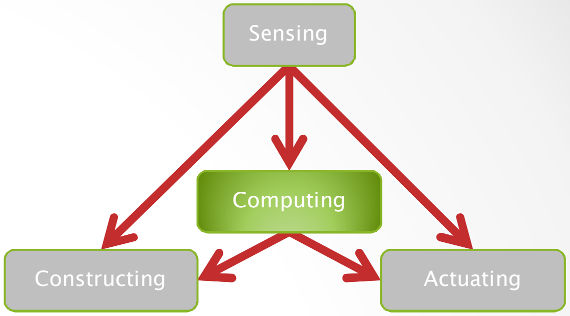


Failed Strand Displacement

- Hence an incorrect binding will undo
  - That’s why toeholds must bind reversibly



- Matching depends on the long segment only
  - Strand displacement succeeds iff the whole long segment matches
  - The address space is determined by the size of the long segment, which is unbounded (not by the size of the toehold)
  - The toehold is just a ‘cache’ of the address



Sensing

Computing

Constructing

Actuating

Computing

• • •

Implementing “Arbitrary” Computing Functions

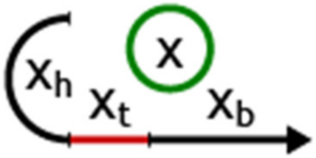
What does DNA Compute?

- Electronics has *electrons*
  - All electrons are the same: you can only count them
  - *Few* electrons = **False**; *lots* of electrons = **True**
  - But **Boolean Logic** is only a necessary evil to build symbolic computation
- DNA computing has *symbols* (DNA words)
  - DNA words are not all the same
  - **Symbolic computation** on **abstract signals** can be done *directly*
  - Signals are presented **concurrently** (in a soup)
  - No requirement to do Boolean Logic
- Then, what are our '**gates**' (if not Boolean?)
  - Theory of Concurrency
  - Process Algebra as the "Boolean Algebra" of DNA Computing

Signals

- A signal is the representation of an abstract event
  - E.g. generated by a sensor
  - E.g. accepted by an effector
  - We are not limited to true/false

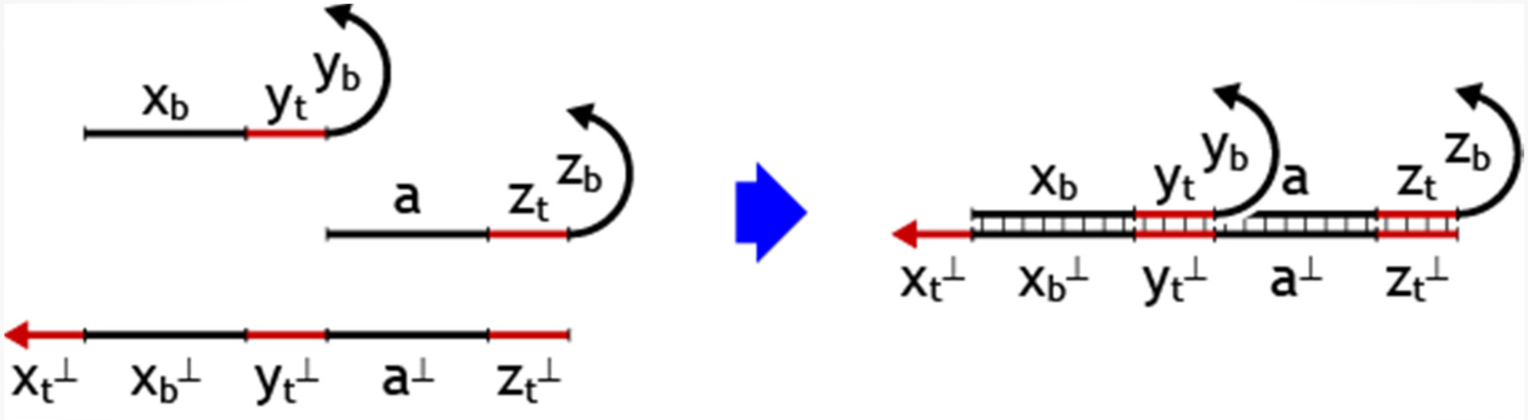
- 3-domain signals
  - x_h: hystory (ignore)
  - x_t: toehold (binding)
  - x_b: body (recognition)



- Signals (single stranded DNA) are prepared by (artificial) DNA synthesis

Gates

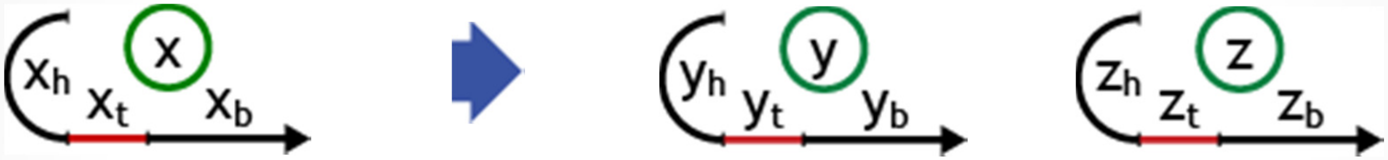
- Double-stranded structures with free toeholds



- Gates are prepared by **self-assembly** from single-stranded DNA that is synthesized

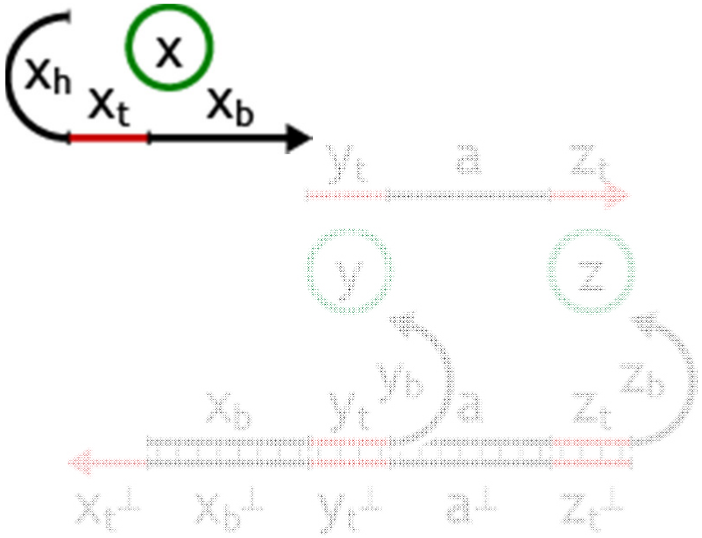
Fork Gate

- $x \rightarrow y + z$



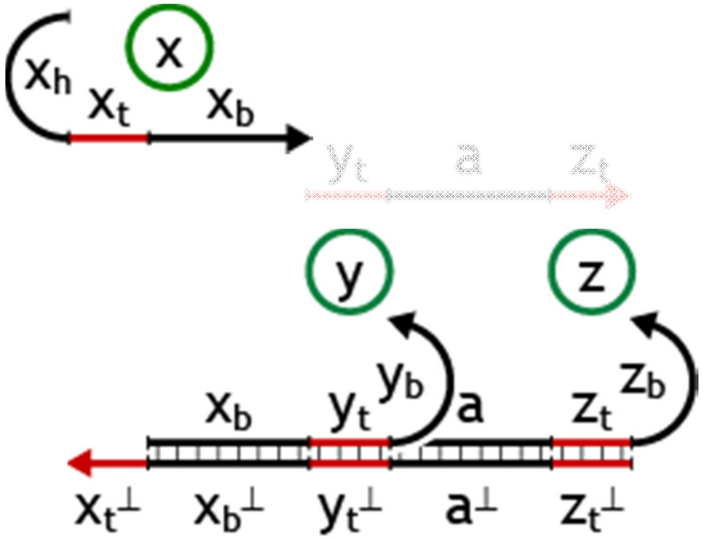
- $x \rightarrow y + 0$ transform x to y (transducer)
- $x \rightarrow x + y$ linear production of y (catalyst)
- $x \rightarrow x + x$ exponential production of x (amplifier)

Fork Gate

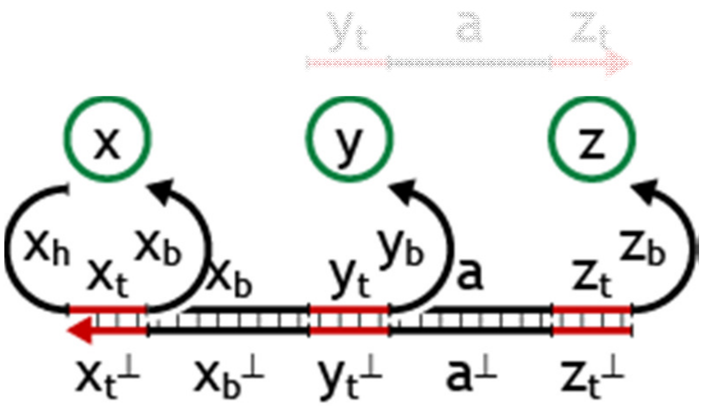


This is the Fork Gate structure

Fork Gate

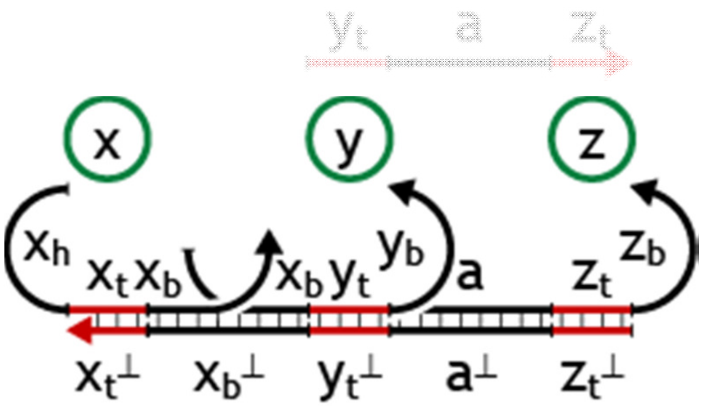

x_h
x
x_t
x_b
y_t
a
z_t
y
z
y_b
z_b
x_b
y_t
a
z_t
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

Fork Gate



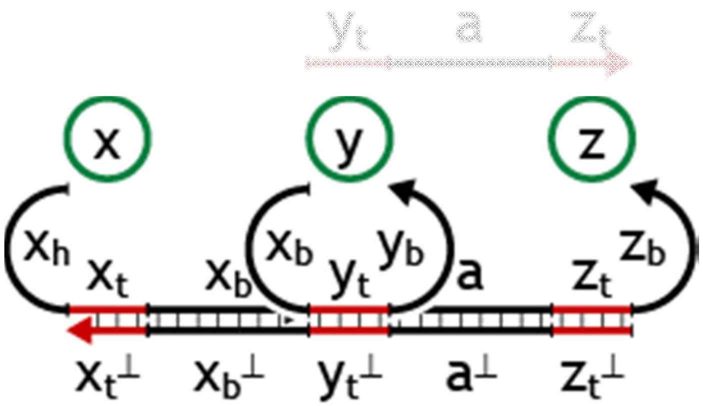
y_t
a
z_t
x
y
z
x_h
x_t
x_b
x_b
y_t
y_b
a
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

Fork Gate



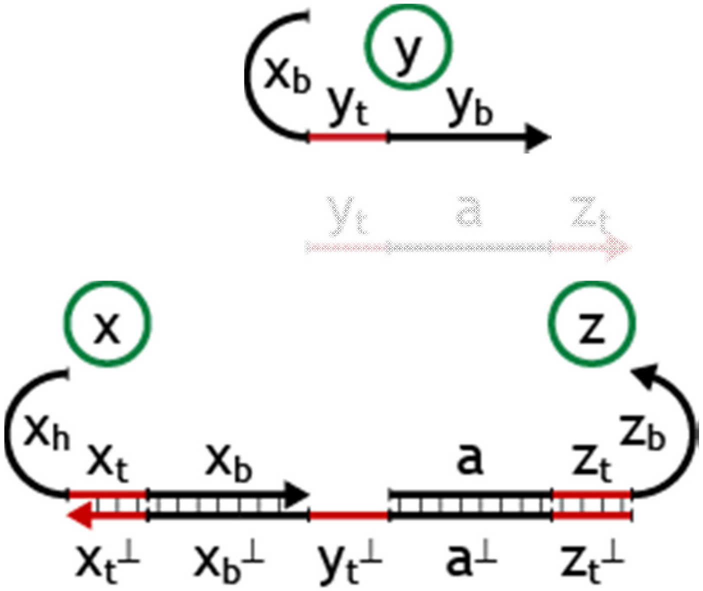
y_t
a
z_t
x
y
z
x_h
x_t
x_b
x_b
y_t
y_b
a
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

Fork Gate

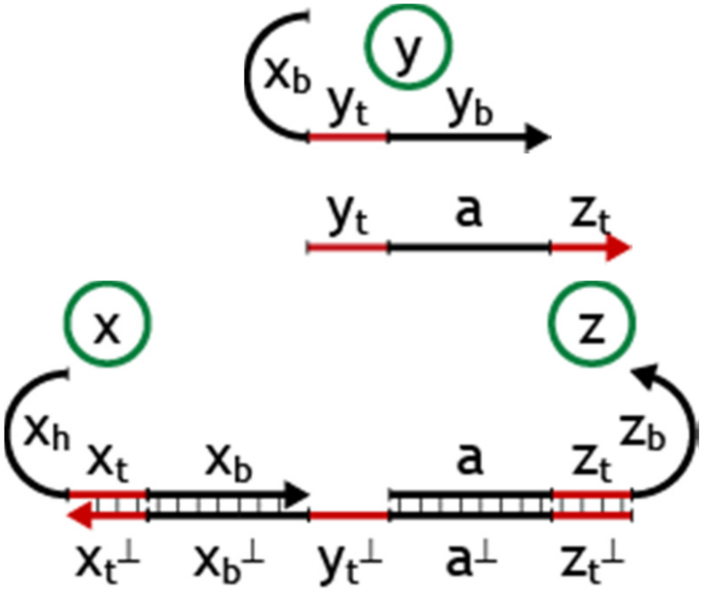


y_t
a
z_t
x
y
z
x_h
x_t
x_b
x_b
y_t
y_b
a
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

Fork Gate

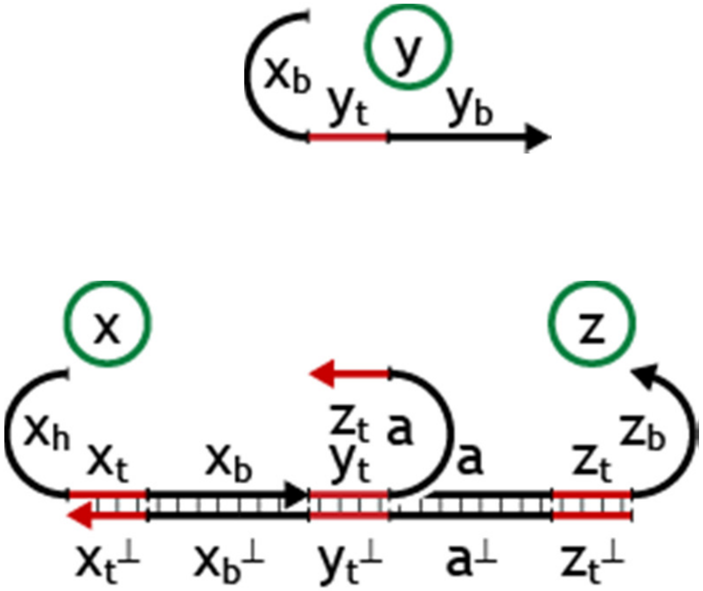

y
x
z
x_b
y_t
y_b
y_t
a
z_t
x_h
x_t
x_b
a
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

Fork Gate

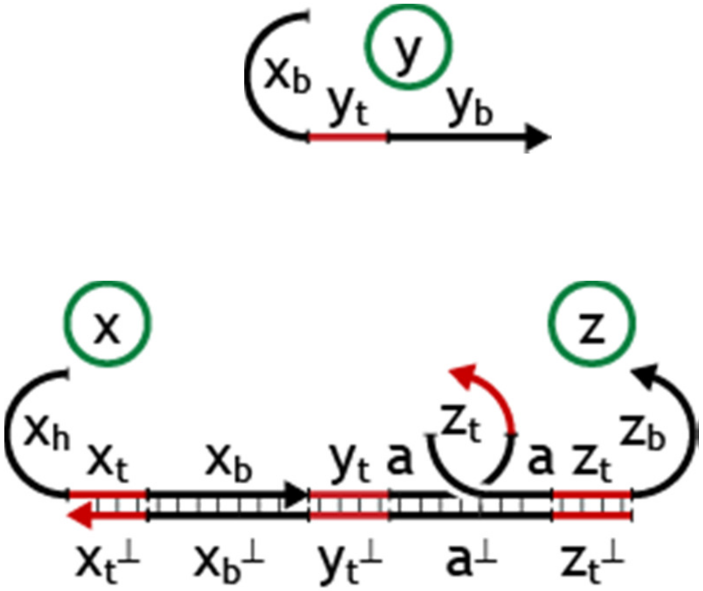


y
x_b
y_t
y_b
y_t
a
z_t
x
z
x_h
x_t
x_b
a
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

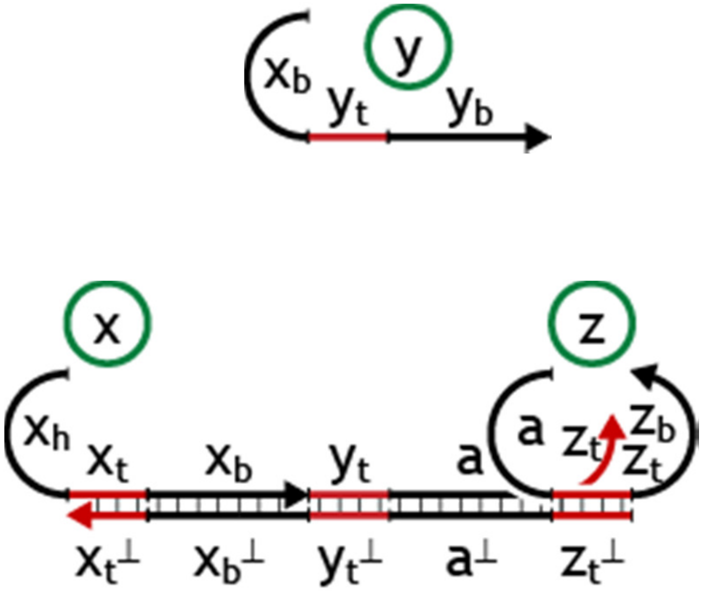
Fork Gate


y
x_b
y_t
y_b
x
z
x_h
x_t
x_b
z_t
a
y_t
a
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

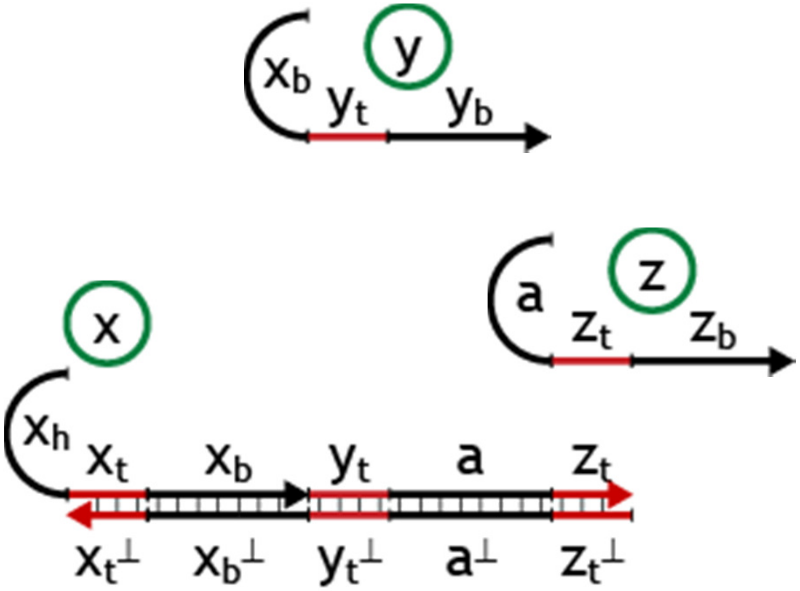
Fork Gate


y
x_b
y_t
y_b
x
z
z_t
x_h
x_t
x_b
y_t
a
a
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$a^\perp$
$z_t^\perp$

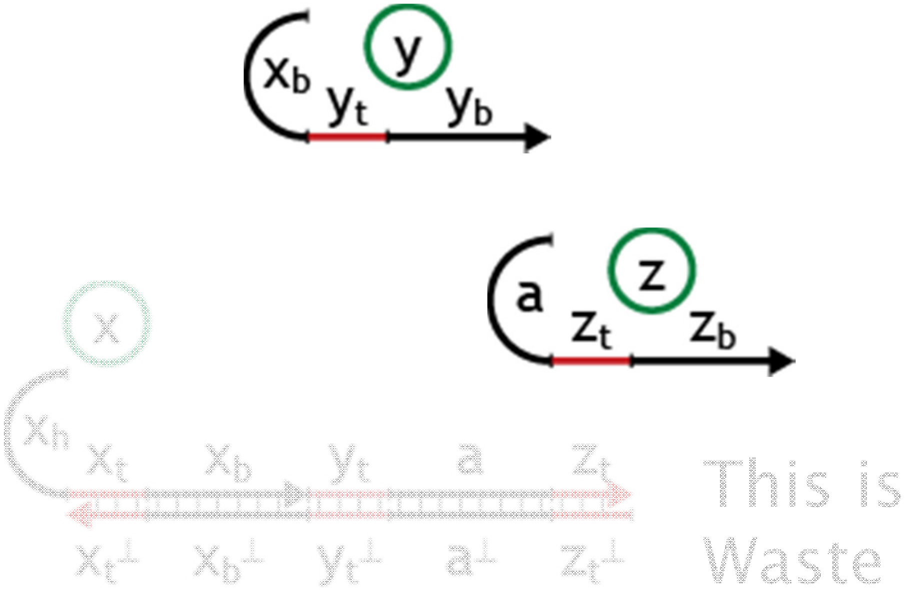
Fork Gate



Fork Gate



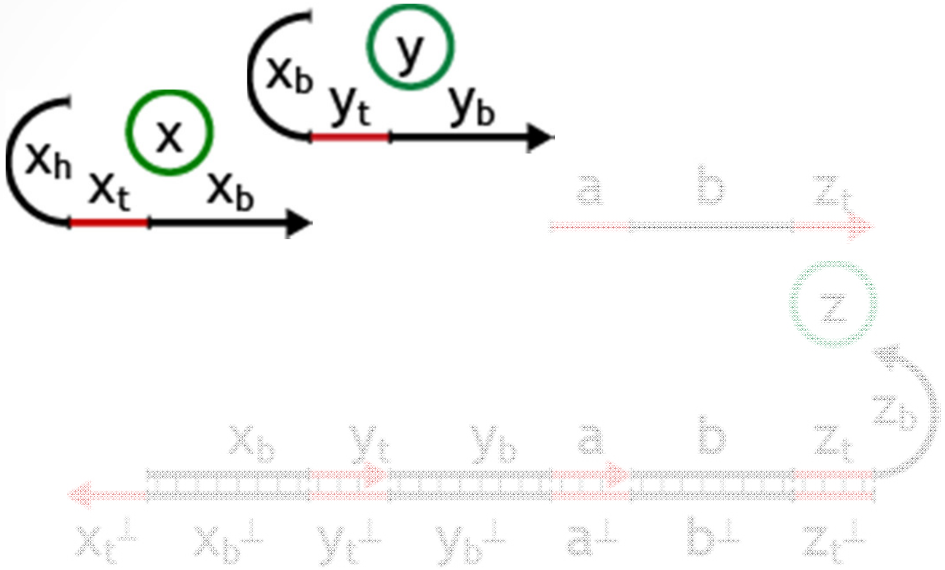
Fork Gate



Join Gate

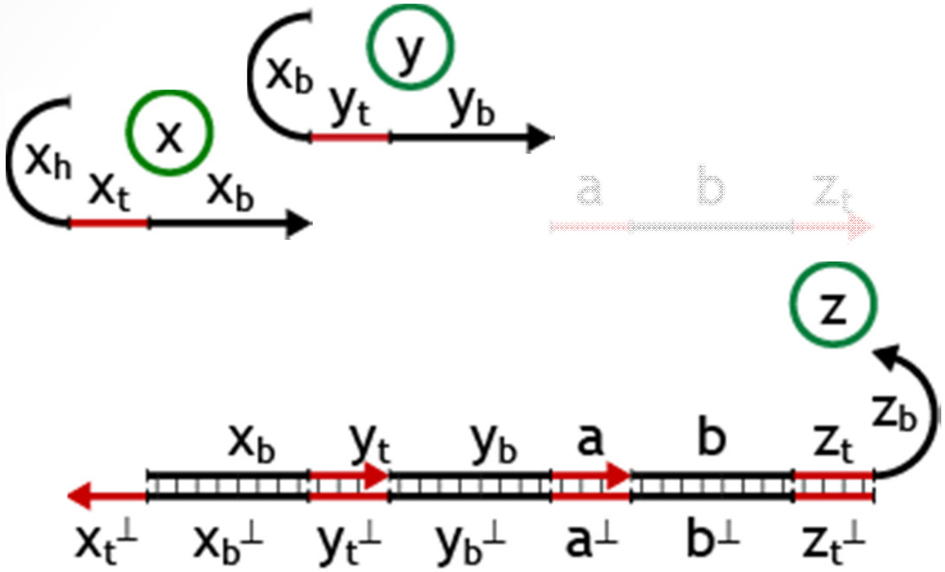
- $x + y \rightarrow z$

Join Gate

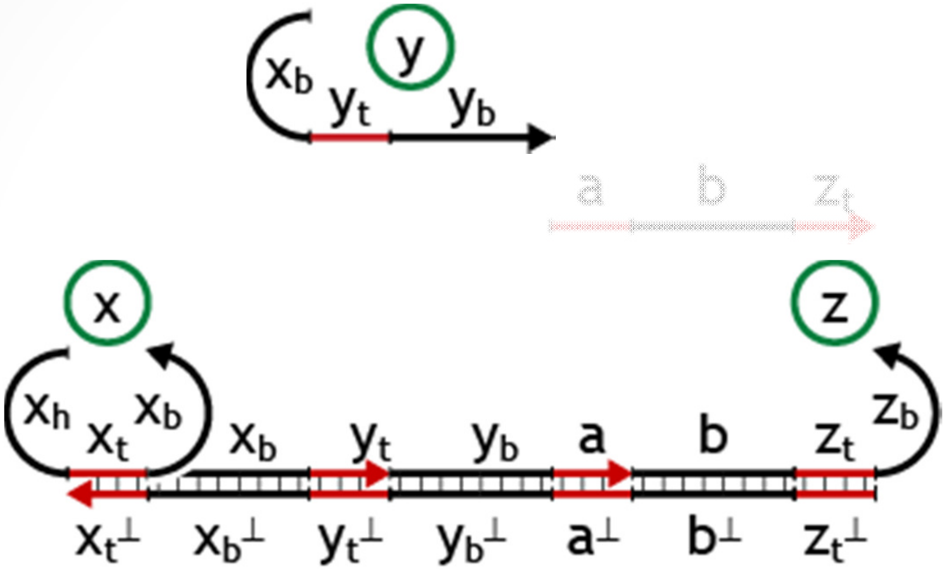


This is the Join Gate structure

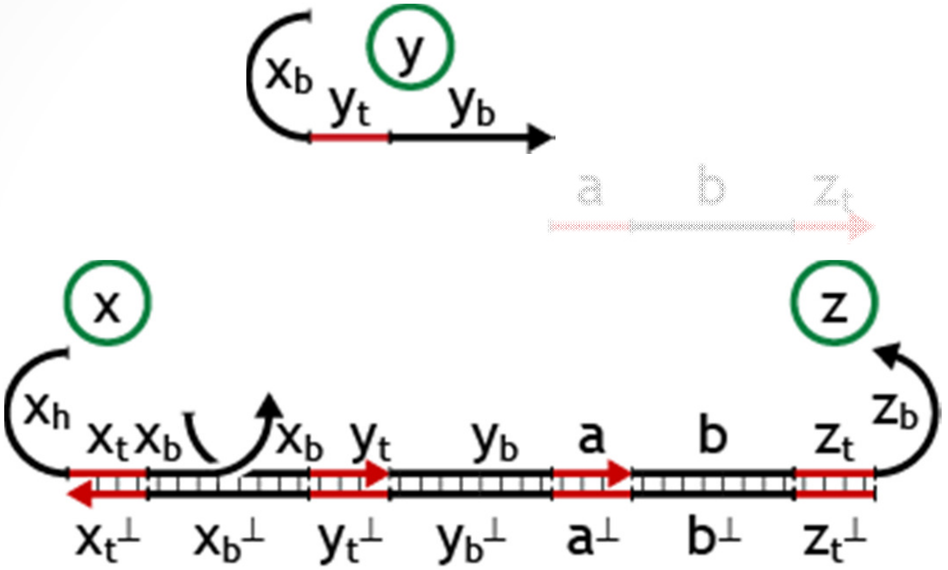
Join Gate


x
x_h
x_t
x_b
y
x_b
y_t
y_b
a
b
z_t
z
z_b
x_b
y_t
y_b
a
b
z_t
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$

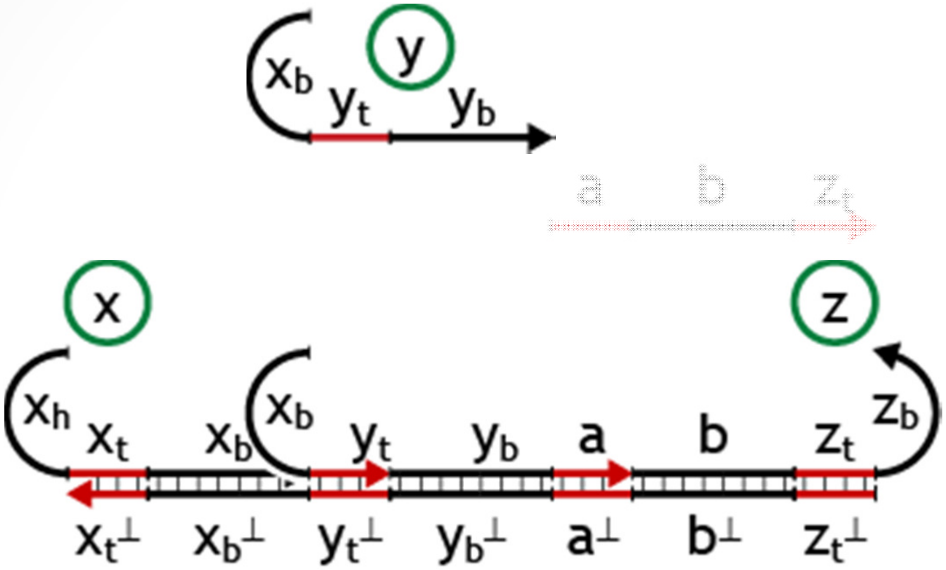
Join Gate



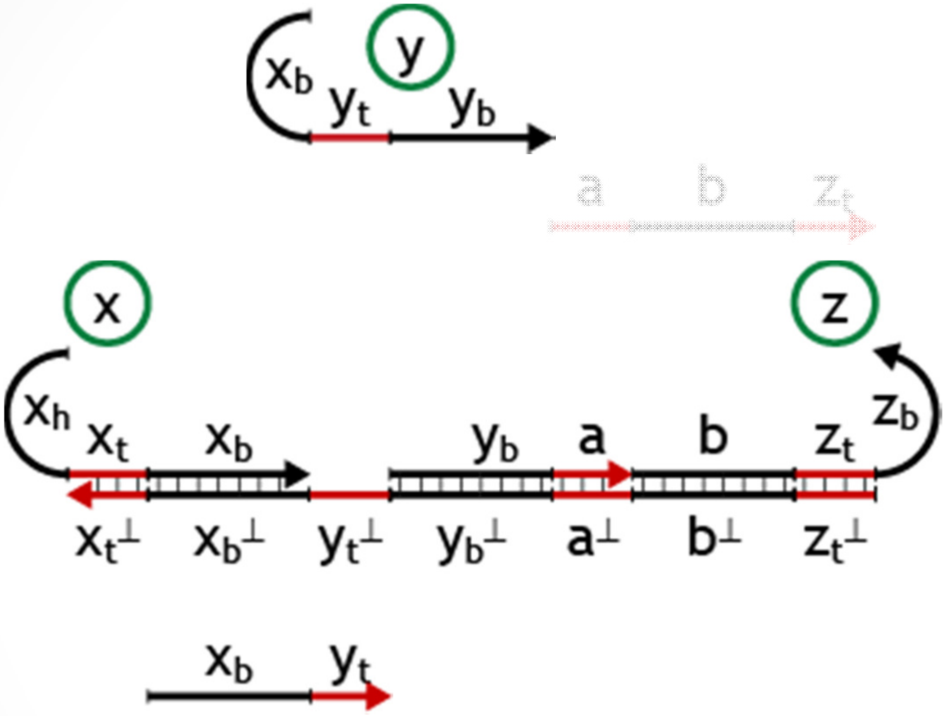
Join Gate


y
x_b
y_t
y_b
a
b
z_t
x
z
x_h
x_t
x_b
x_b
y_t
y_b
a
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$

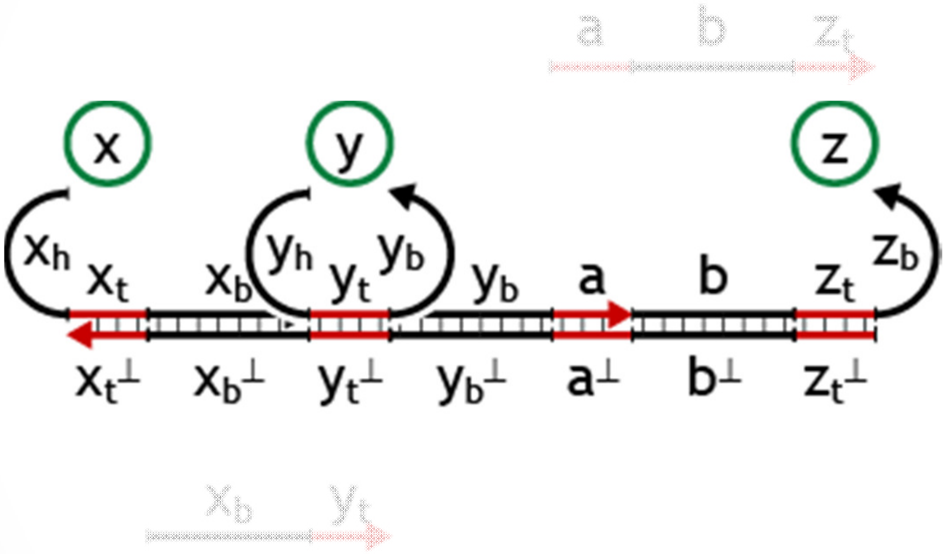
Join Gate



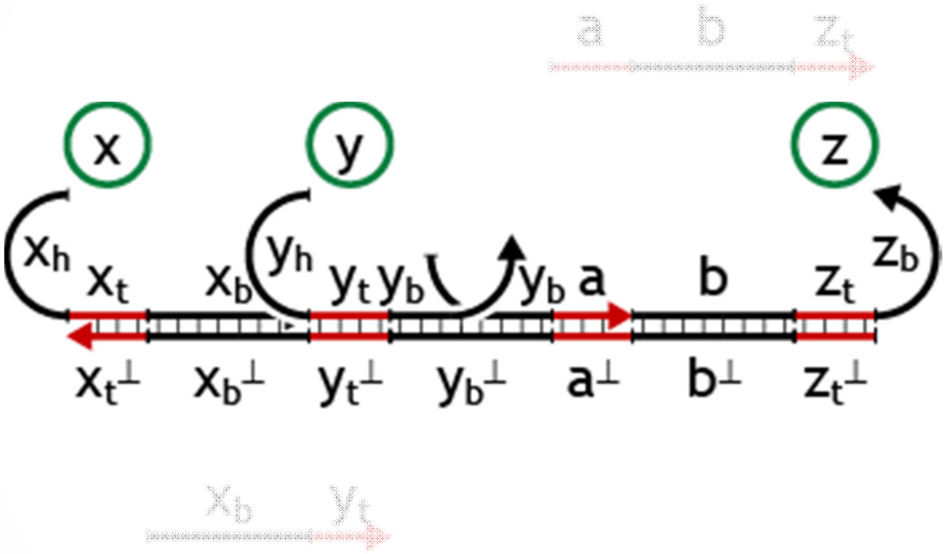
Join Gate


y
x_b
y_t
y_b
a
b
z_t
x
z
x_h
x_t
x_b
y_b
a
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t

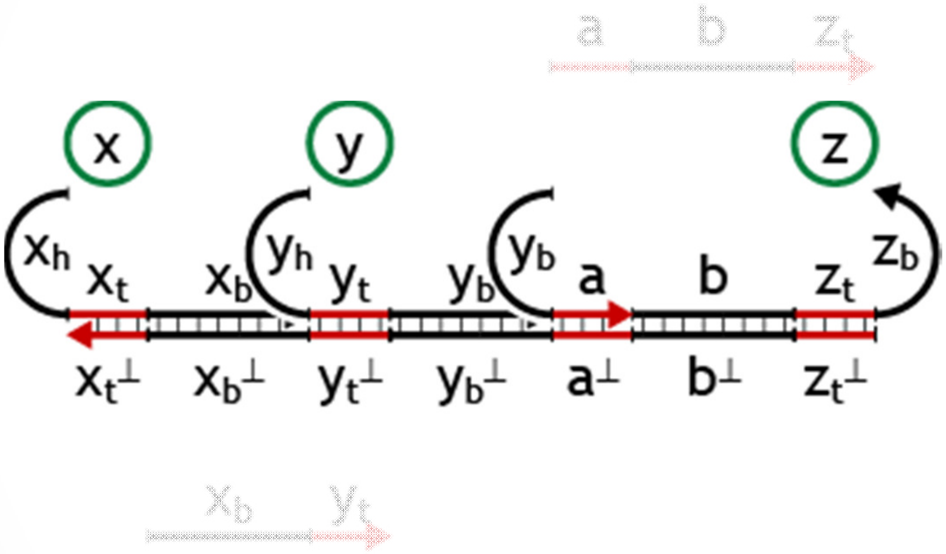
Join Gate


a
b
z_t
x
y
z
x_h
x_t
x_b
y_h
y_t
y_b
y_b
a
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t

Join Gate

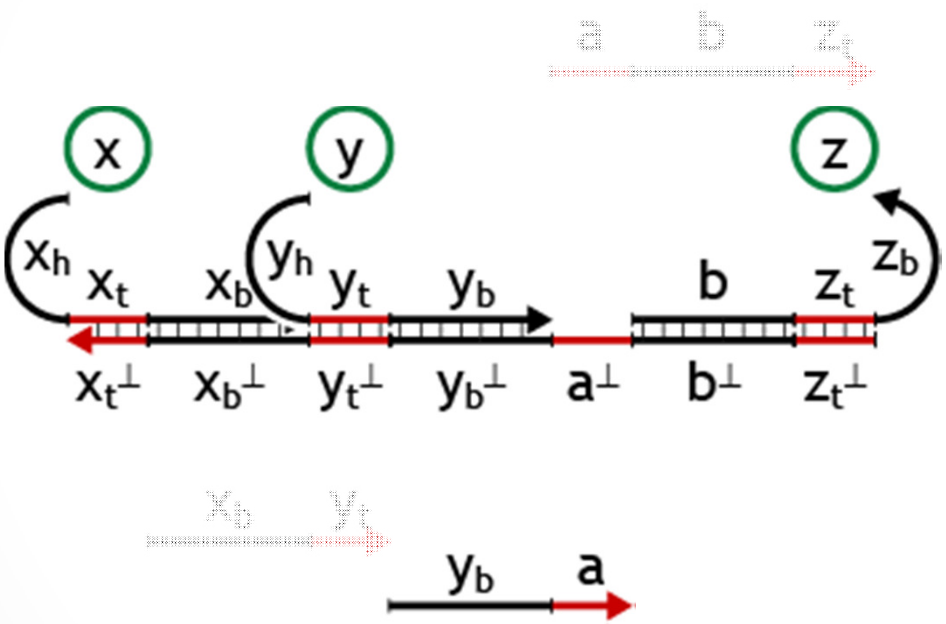


Join Gate

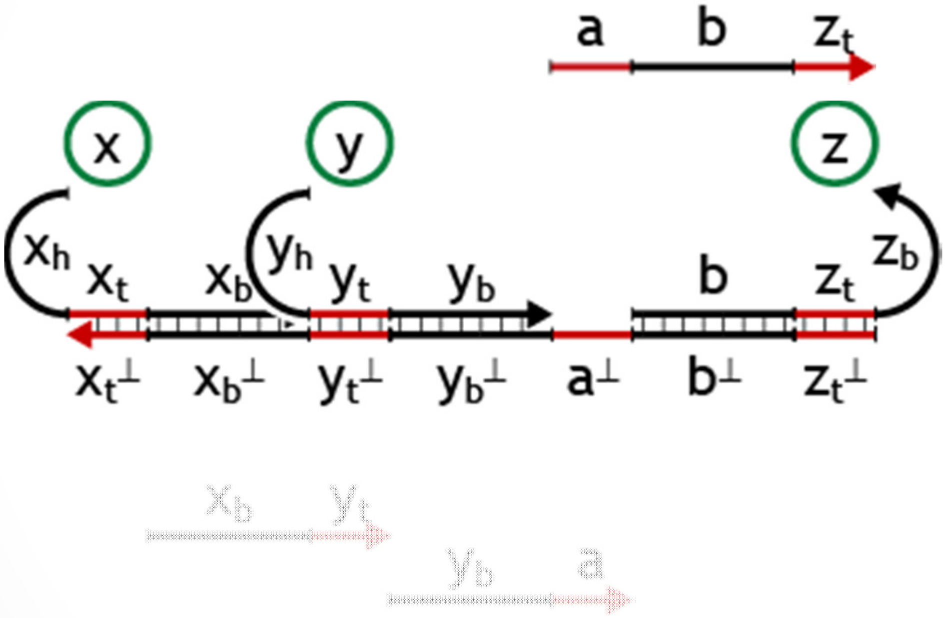


a
b
z_t
x
y
z
x_h
x_t
x_b
y_h
y_t
y_b
y_b
a
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t

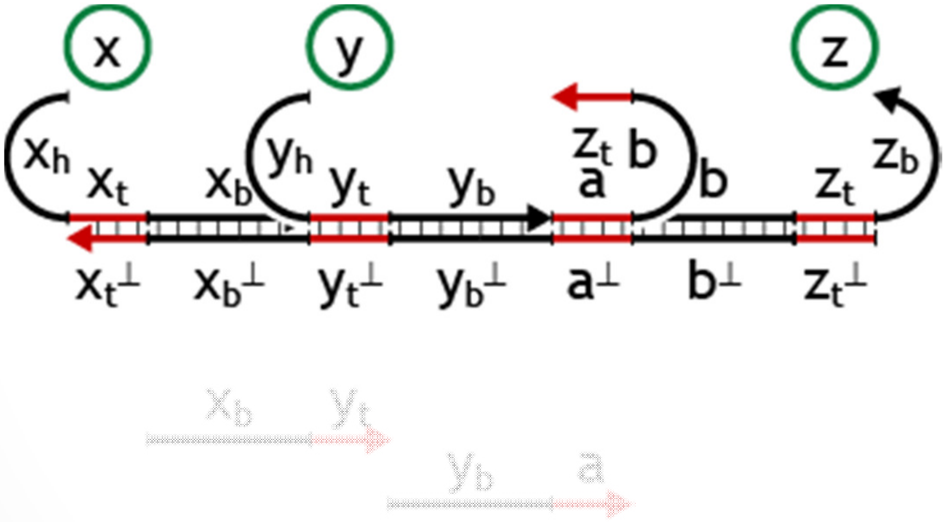
Join Gate


a
b
z_t
x
y
z
x_h
x_t
x_b
y_h
y_t
y_b
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t
y_b
a

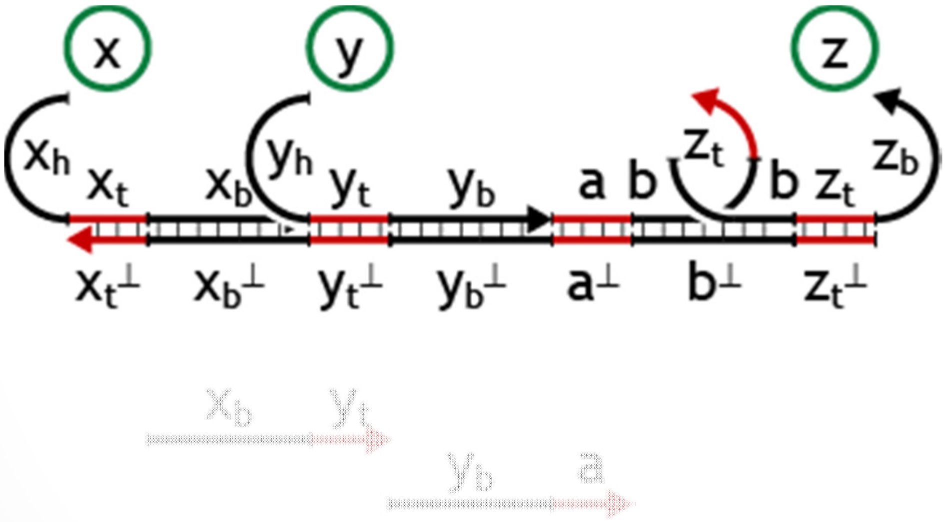
Join Gate


a
b
z_t
x
y
z
x_h
x_t
x_b
y_h
y_t
y_b
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t
y_b
a

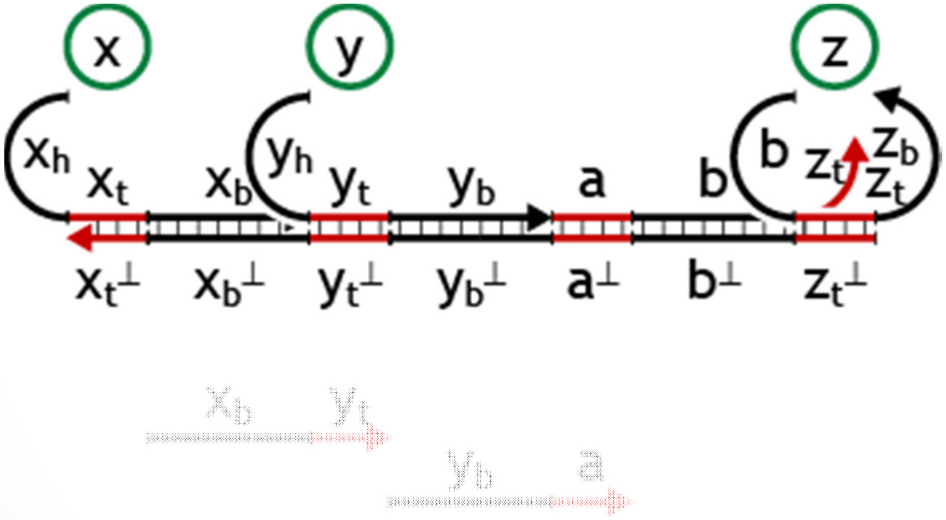
Join Gate


x
y
z
x_h
x_t
x_b
y_h
y_t
y_b
z_t
a
b
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t
y_b
a

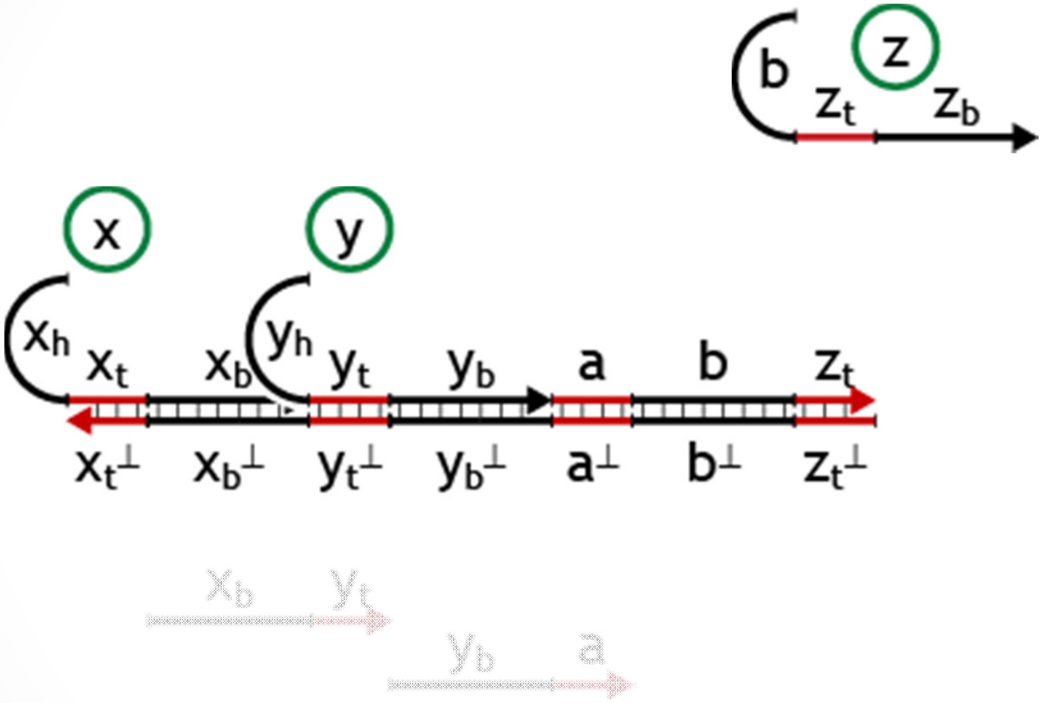
Join Gate


x
y
z
x_h
x_t
x_b
y_h
y_t
y_b
a
b
z_t
b
z_t
z_b
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t
y_b
a

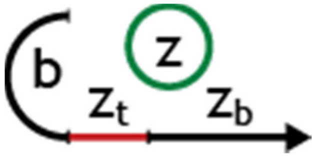
Join Gate

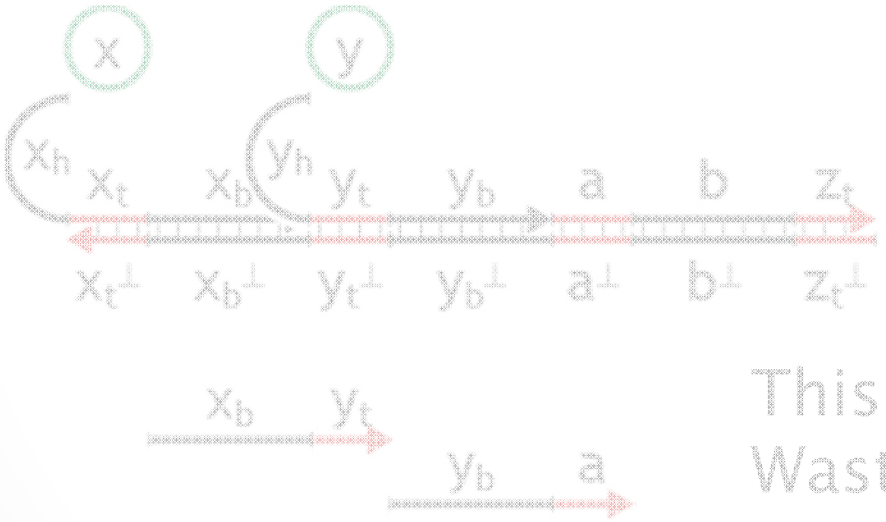


Join Gate


b
z
z_t
z_b
x
y
x_h
x_t
x_b
y_h
y_t
y_b
a
b
z_t
$x_t^\perp$
$x_b^\perp$
$y_t^\perp$
$y_b^\perp$
$a^\perp$
$b^\perp$
$z_t^\perp$
x_b
y_t
y_b
a

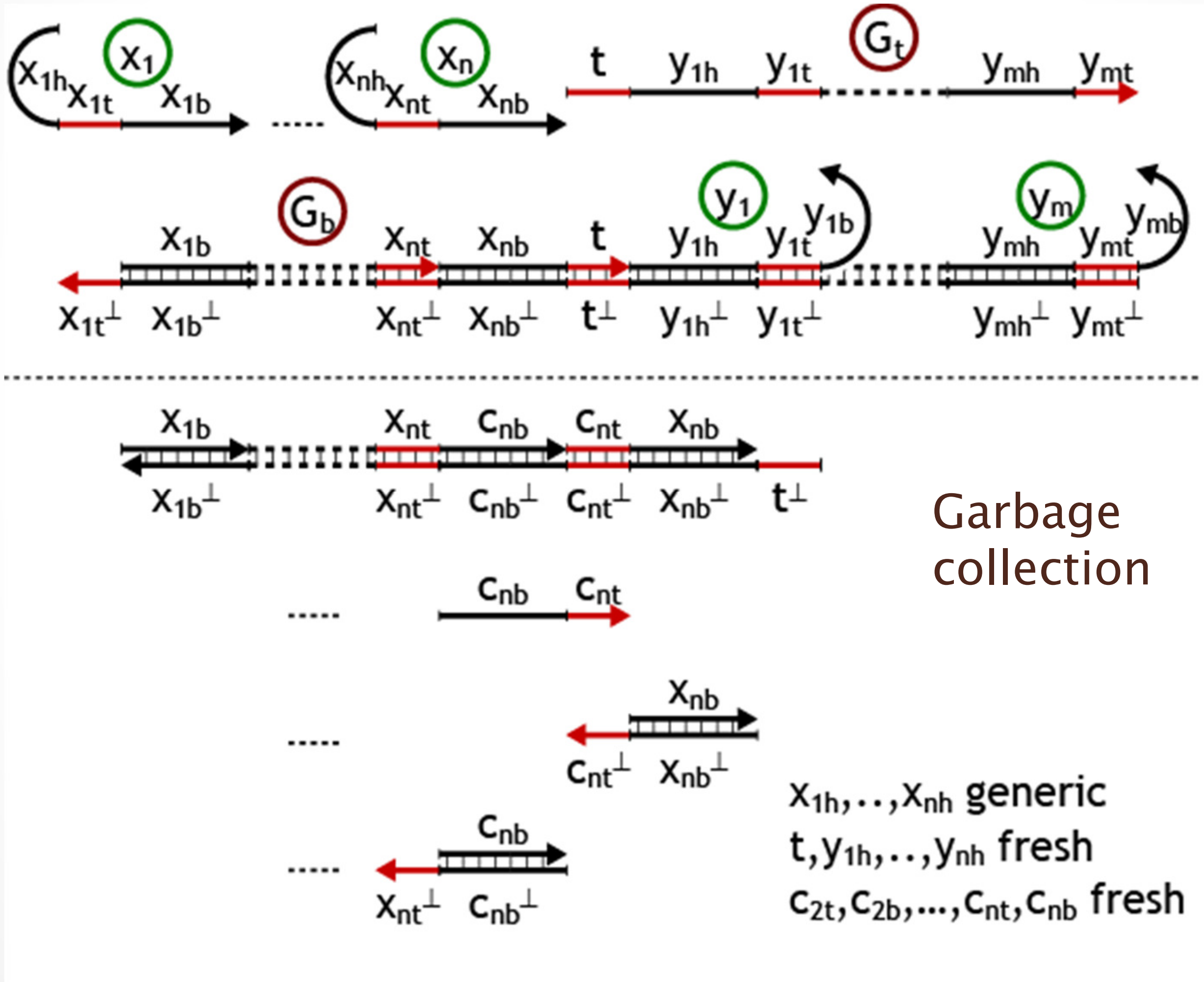
Join Gate





This is Waste

General n-Join/m-Fork Gate



Garbage collection

$x_{1h},..,x_{nh}$ generic
$t,y_{1h},..,y_{nh}$ fresh
$c_{2t},c_{2b},...,c_{nt},c_{nb}$ fresh

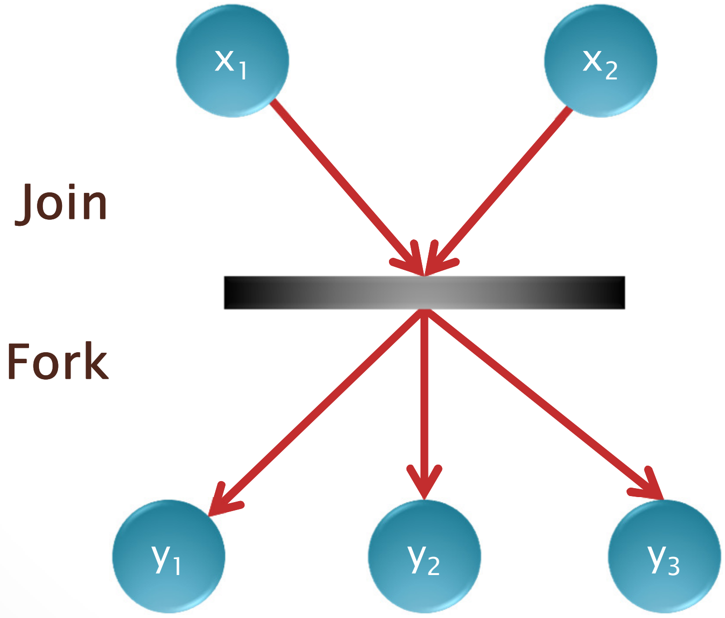
Gate Design Verification

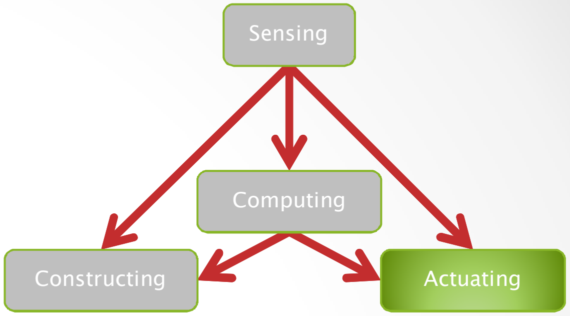
- Active garbage
  - The active join residuals slow down the performance of following joins.
  - → Add a garbage collector to remove the active residuals.
- Interference between gates
  - The join garbage collector interferes with the fork gate.
  - → Modify the fork gate to remove the interference.
- What else could go wrong?
  - Endless possibilities.
  - → Prove that the fork/join gate structures correctly implement fork/join in all larger circuits.

Strand Algebra

$$x_1 \mid .. \mid x_n \mid [x_1,..,x_n].[y_1,..,y_m] \rightarrow y_1 \mid .. \mid y_m$$

- Join + Fork + Populations = (Stochastic) Petri Nets




Sensing

Computing

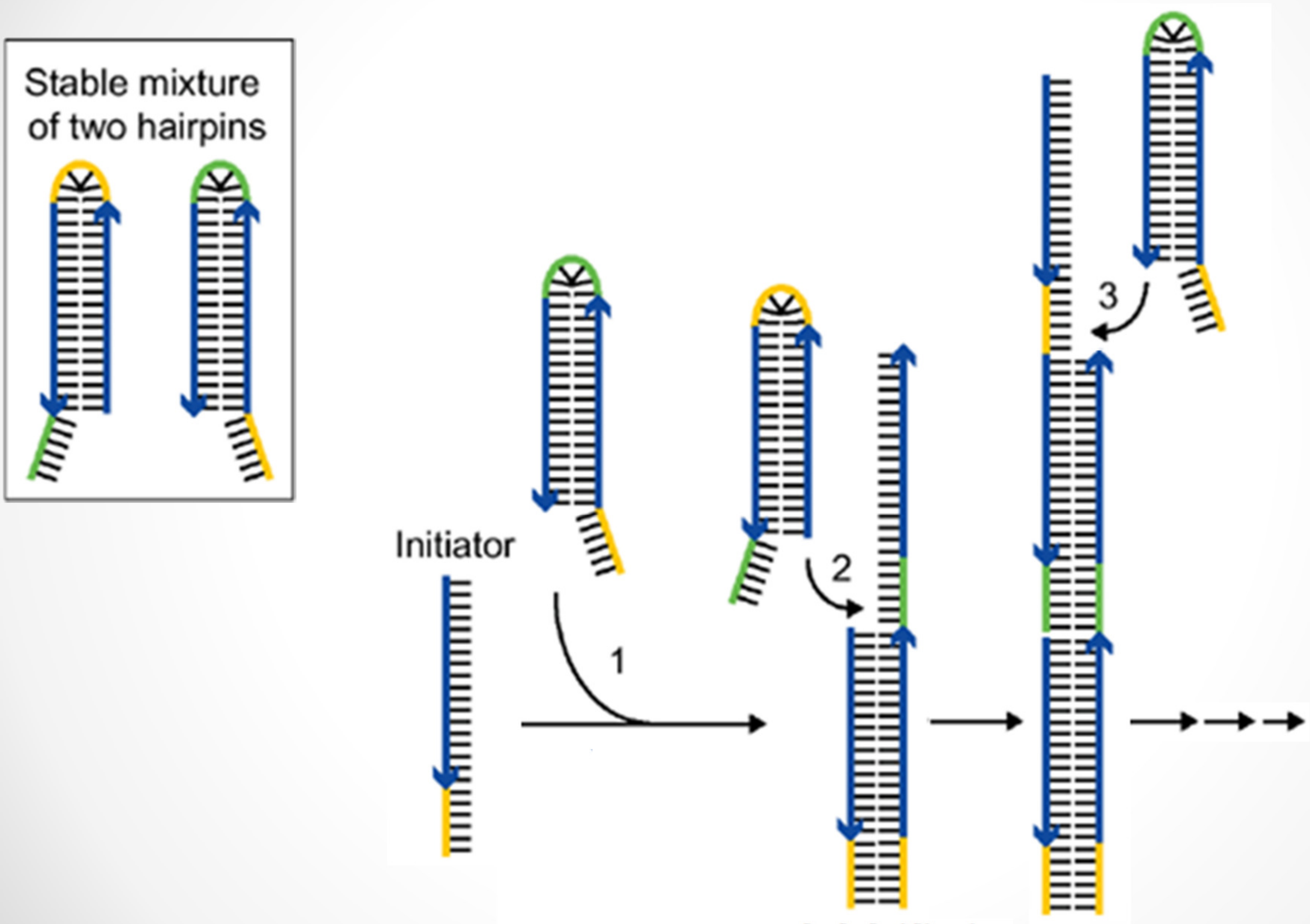
Constructing

Actuating

Actuating

• • •

Hybridization Chain Reaction

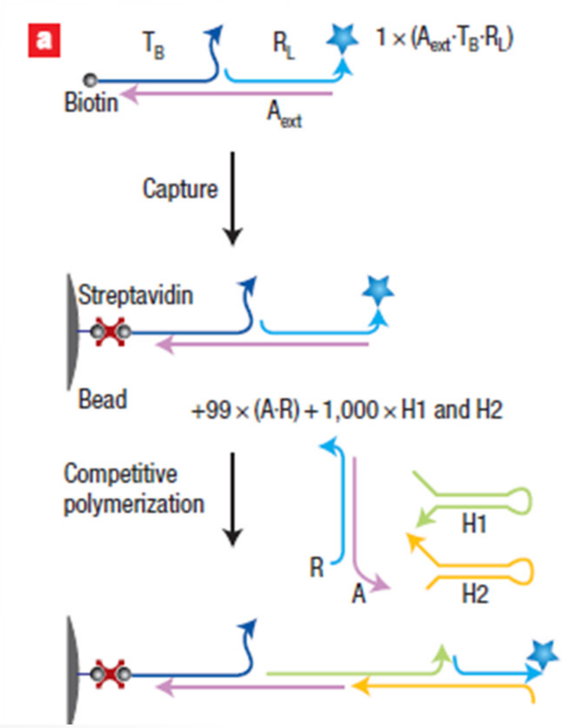


n by hybridization

chain reaction

Robert M. Dirks† and Niles A. Pierce‡,§

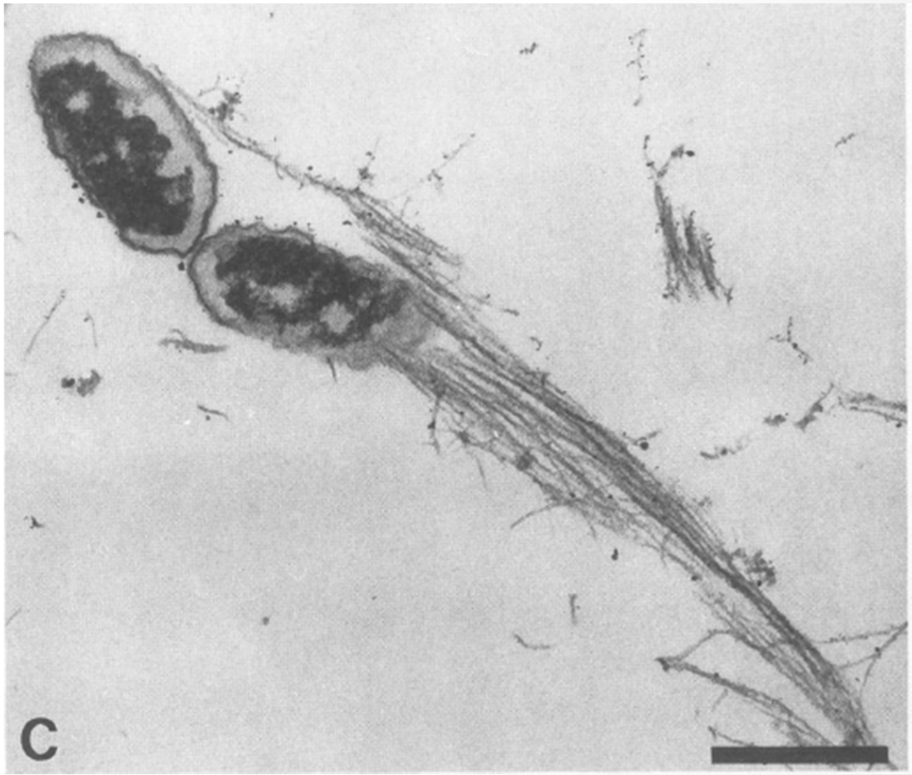
Polymerization Motor



An autonomous polymerization motor powered by DNA hybridization

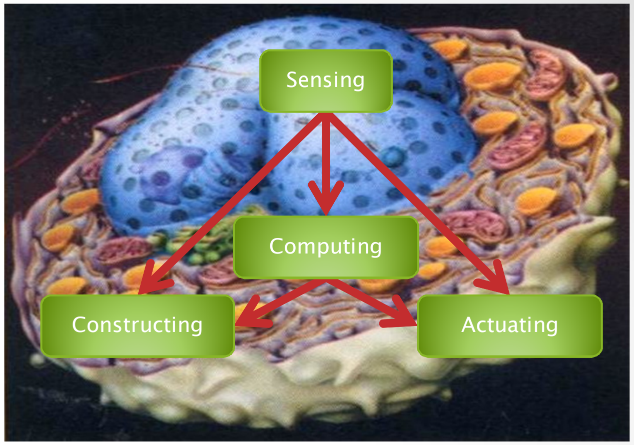
SUVIR VENKATARAMAN[1], ROBERT M. DIRKS[1], PAUL W. K. ROTHEMUND[2,3], ERIK WINFREE[2,3] AND NILES A. PIERCE[1,4]*

Rickettsia (spotted fever)



Directional Actin Polymerization Associated with Spotted Fever Group Rickettsia Infection of Vero Cells

ROBERT A. HEINZEN, STANLEY F. HAYES, MARIUS G. PEACOCK, AND TED HACKSTADT*



Curing

…

A Doctor in Each Cell

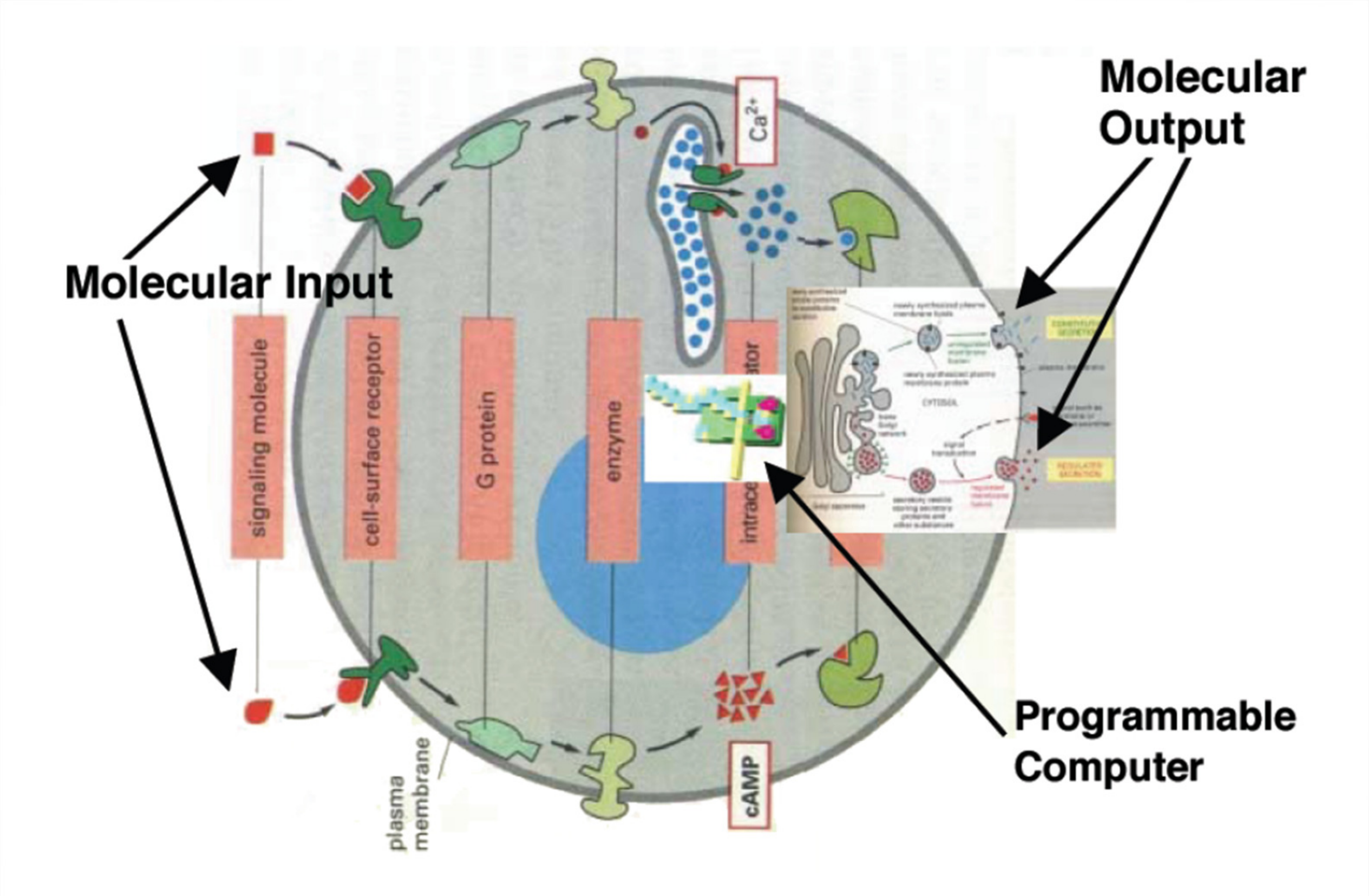


Fig. 1 Medicine in 2050: "Doctor in a Cell"

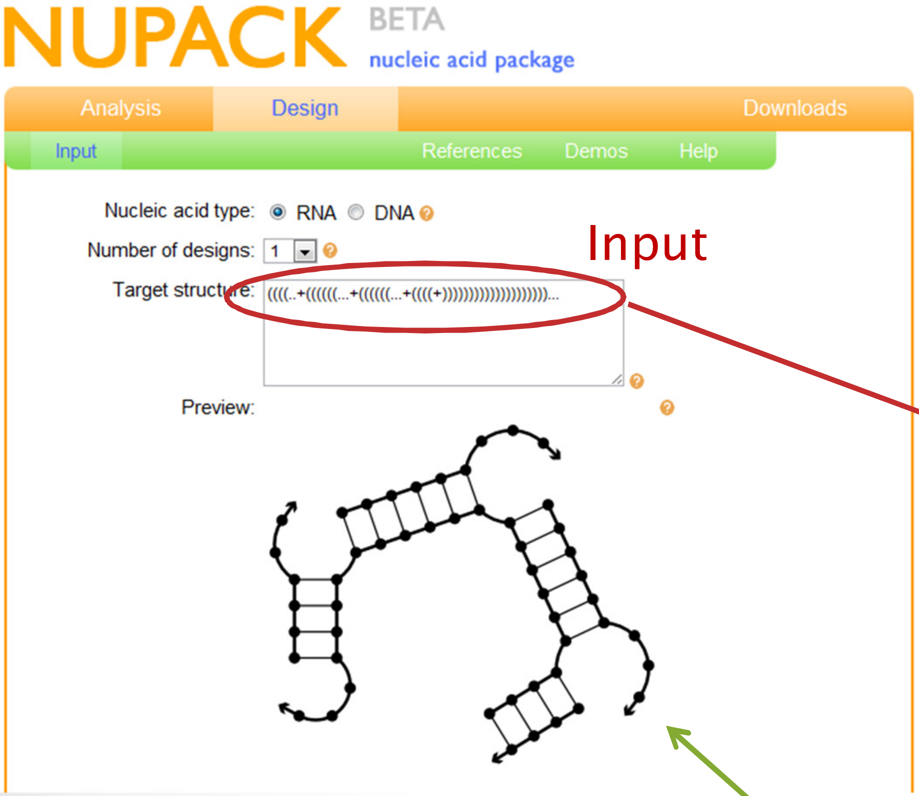
Ehud Shapiro

Rivka Adar
Kobi Benenson
Gregory Linshitz
Aviv Regev
William Silverman

Molecules and computation

Tools

Sequence Design



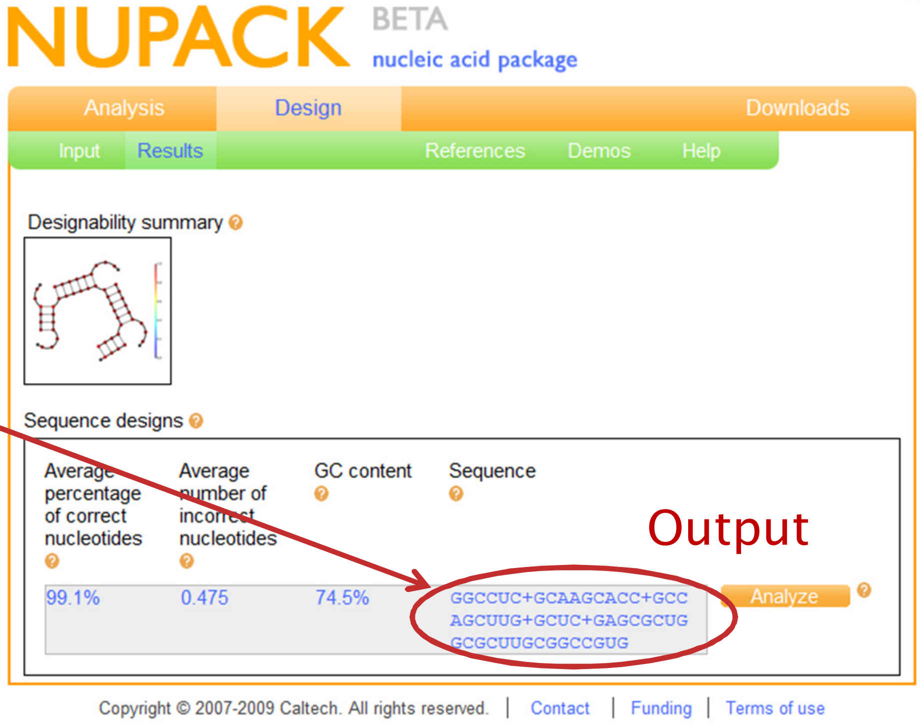
Nucleic acid type: ● RNA ○ DNA

Number of designs: 1

Target structure: ((((..+((((((...+((((((...+((((+))))))))))))))))))))...

Preview:

Input



Average percentage of correct nucleotides	Average number of incorrect nucleotides	GC content	Sequence
99.1%	0.475	74.5%	GGCCUC+GCAAGCACC+GCCAGCUUG+GCUC+GAGCGCUGGCGCUUGCGGCCGUG

Output

| Contact | Funding | Terms of use

So we can in principle work at this level.

Visual DSD

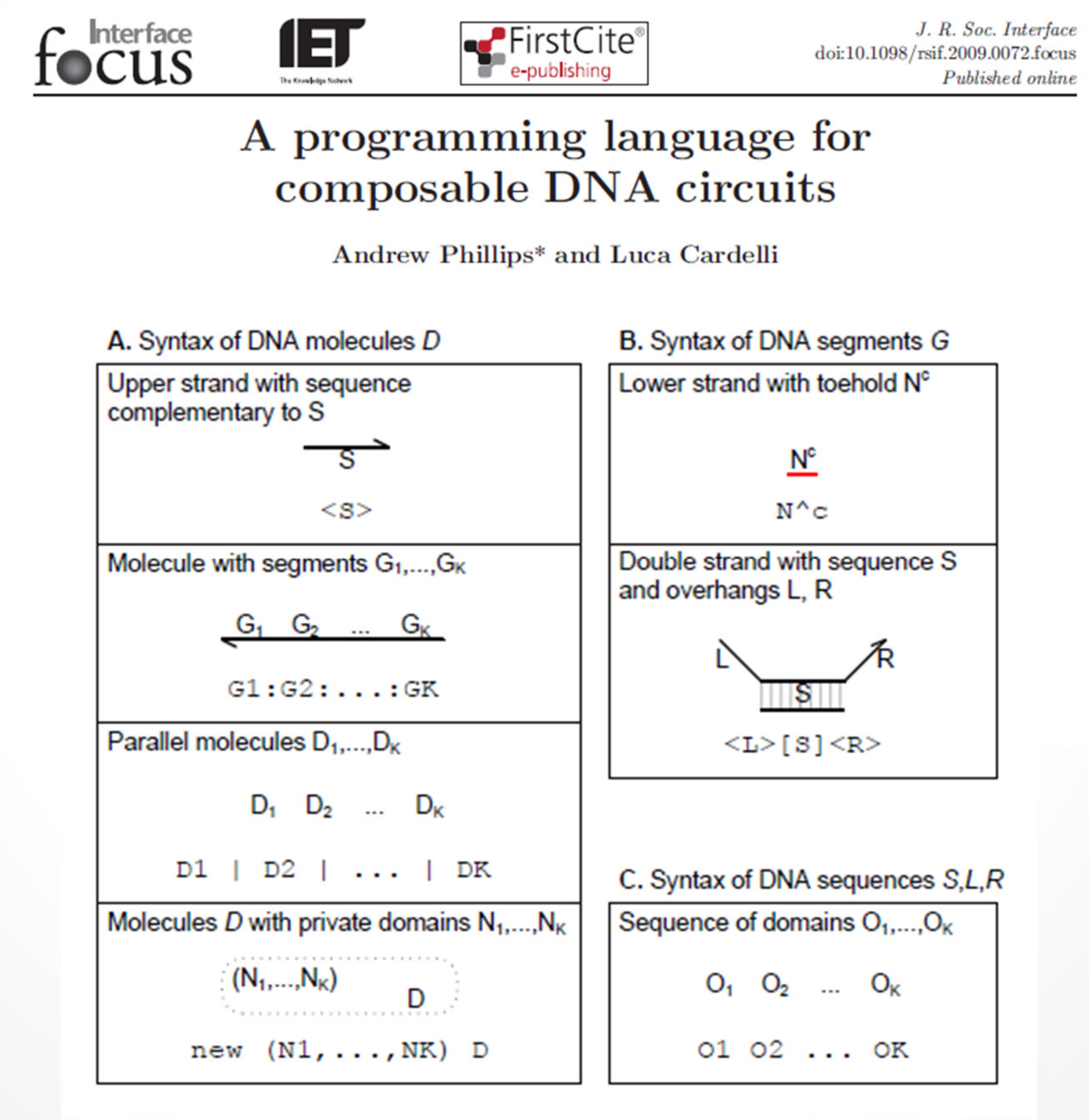
A Strand Displacement Simulator

• • •

Matthew Lakin, Simon Youssef, Andrew Phillips

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

Syntax



Interface focus IET FirstCite e-publishing

J. R. Soc. Interface
doi:10.1098/rsif.2009.0072.focus
Published online

A programming language for composable DNA circuits

Andrew Phillips* and Luca Cardelli

A. Syntax of DNA molecules *D*

Upper strand with sequence complementary to S

S

`<S>`

Molecule with segments $G_1,\ldots,G_K$

G_1 G_2 … G_K

`G1:G2:...:GK`

Parallel molecules $D_1,\ldots,D_K$

D_1 D_2 … D_K

`D1 | D2 | ... | DK`

Molecules *D* with private domains $N_1,\ldots,N_K$

$(N_1,\ldots,N_K)$ D

`new (N1,...,NK) D`

B. Syntax of DNA segments *G*

Lower strand with toehold N^c

N^c

`N^c`

Double strand with sequence S and overhangs L, R

L S R

`<L>[S]<R>`

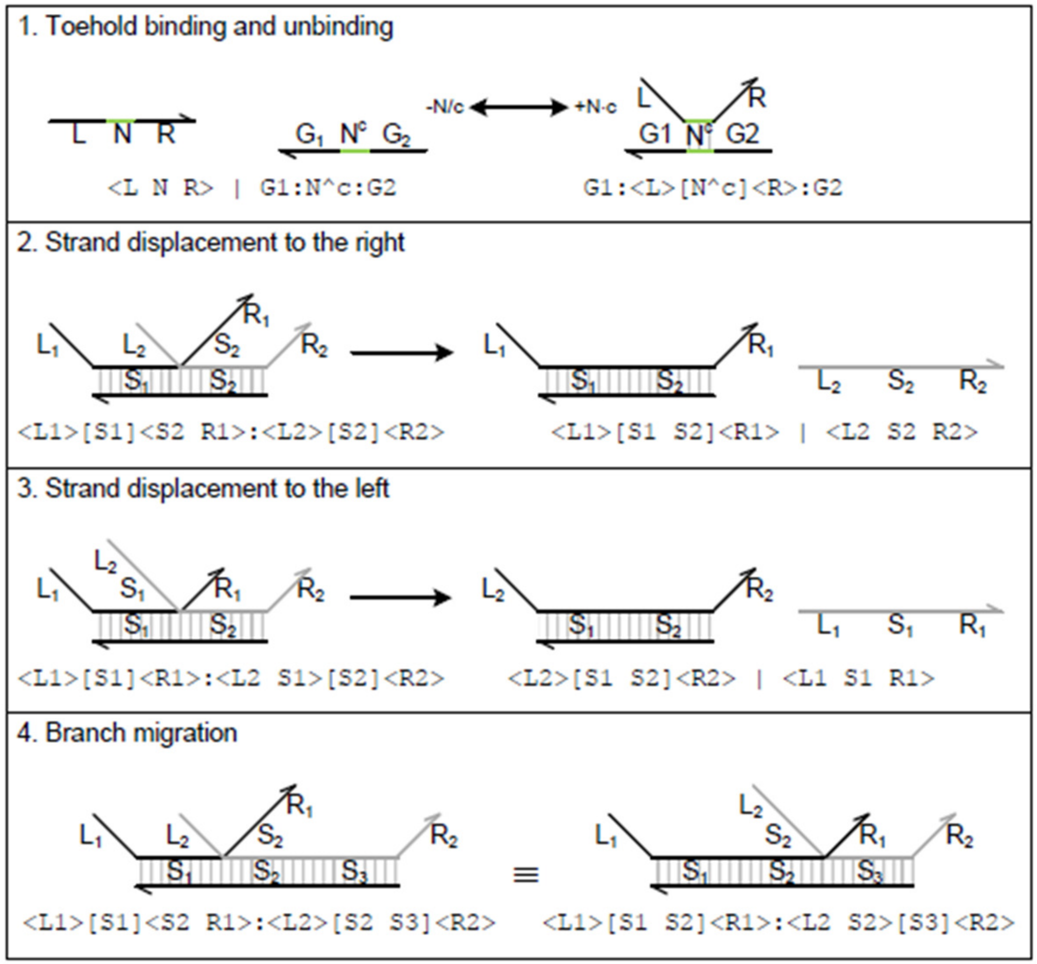
C. Syntax of DNA sequences *S*,*L*,*R*

Sequence of domains $O_1,\ldots,O_K$

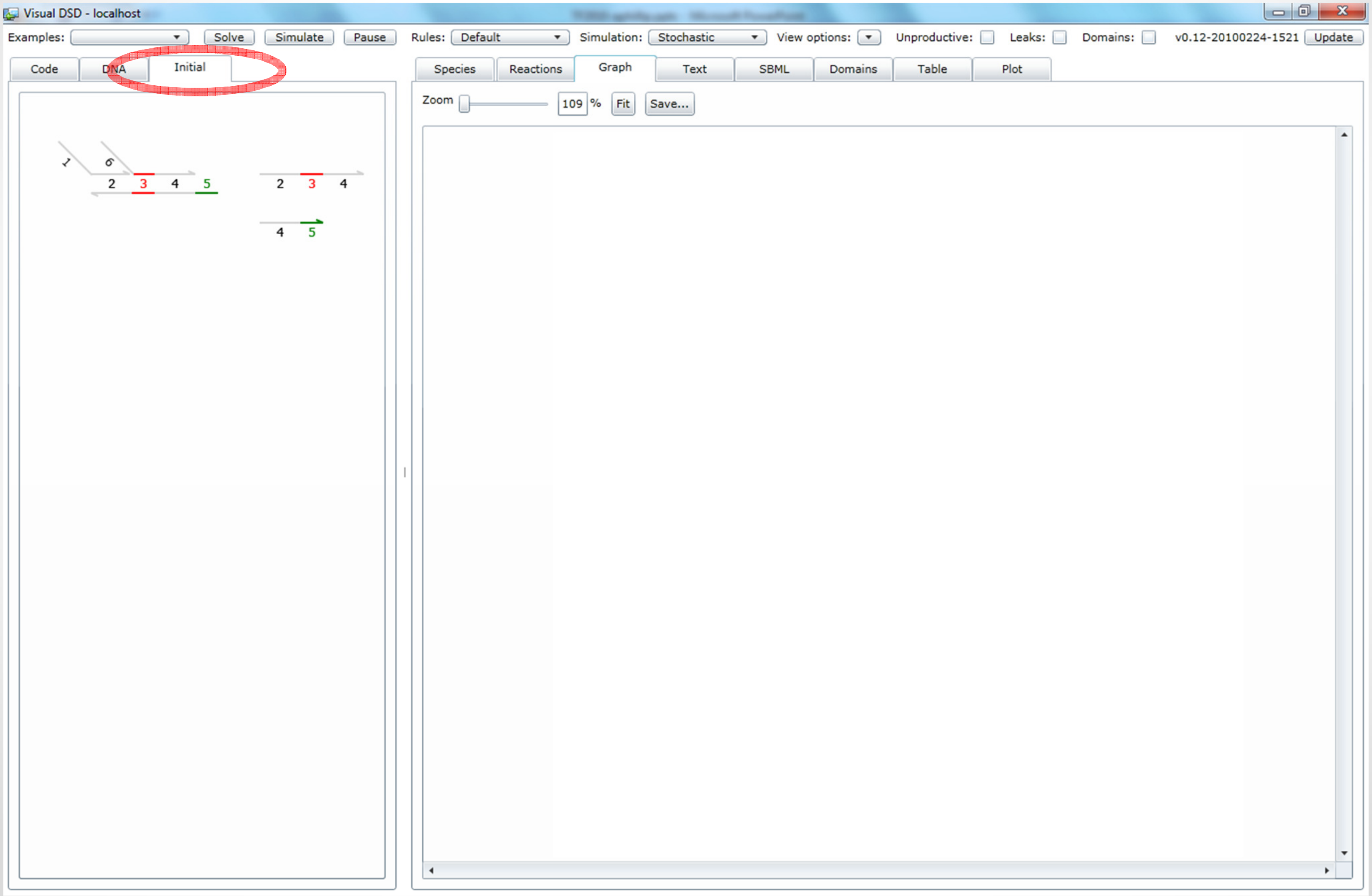
O_1 O_2 … O_K

`O1 O2 ... OK`

Dynamics


1. Toehold binding and unbinding
L N R
G1 N^c G2
-N/c
+N·c
L R
G1 N^c G2
<L N R> | G1:N^c:G2
G1:<L>[N^c]<R>:G2
2. Strand displacement to the right
L1 L2 R1 S2 R2
S1 S2
L1 R1
S1 S2
L2 S2 R2
<L1>[S1]<S2 R1>:<L2>[S2]<R2>
<L1>[S1 S2]<R1> | <L2 S2 R2>
3. Strand displacement to the left
L2 S1 L1 R1 R2
S1 S2
L2 R2
S1 S2
L1 S1 R1
<L1>[S1]<R1>:<L2 S1>[S2]<R2>
<L2>[S1 S2]<R2> | <L1 S1 R1>
4. Branch migration
L1 L2 S2 R1 R2
S1 S2 S3
≡
L1 L2 S2 R1 R2
S1 S2 S3
<L1>[S1]<S2 R1>:<L2>[S2 S3]<R2>
<L1>[S1 S2]<R1>:<L2 S2>[S3]<R2>

Initial Species



Visual DSD - localhost

Examples:

Rules: Default

Simulation: Stochastic

Unproductive: ☐

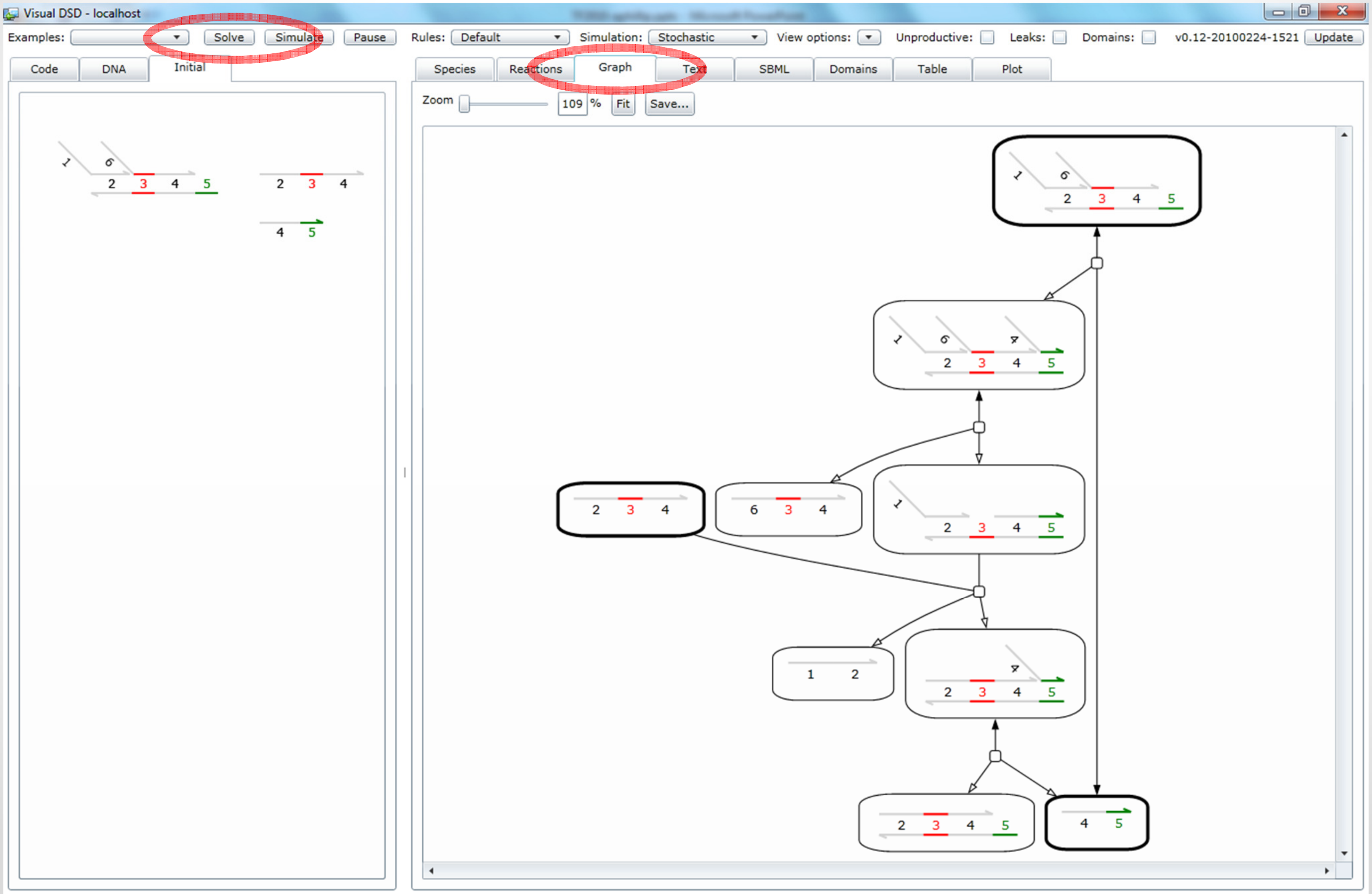
Leaks: ☐

Domains: ☐

v0.12-20100224-1521

Zoom 109 %

Reaction Graph



Rules: Default

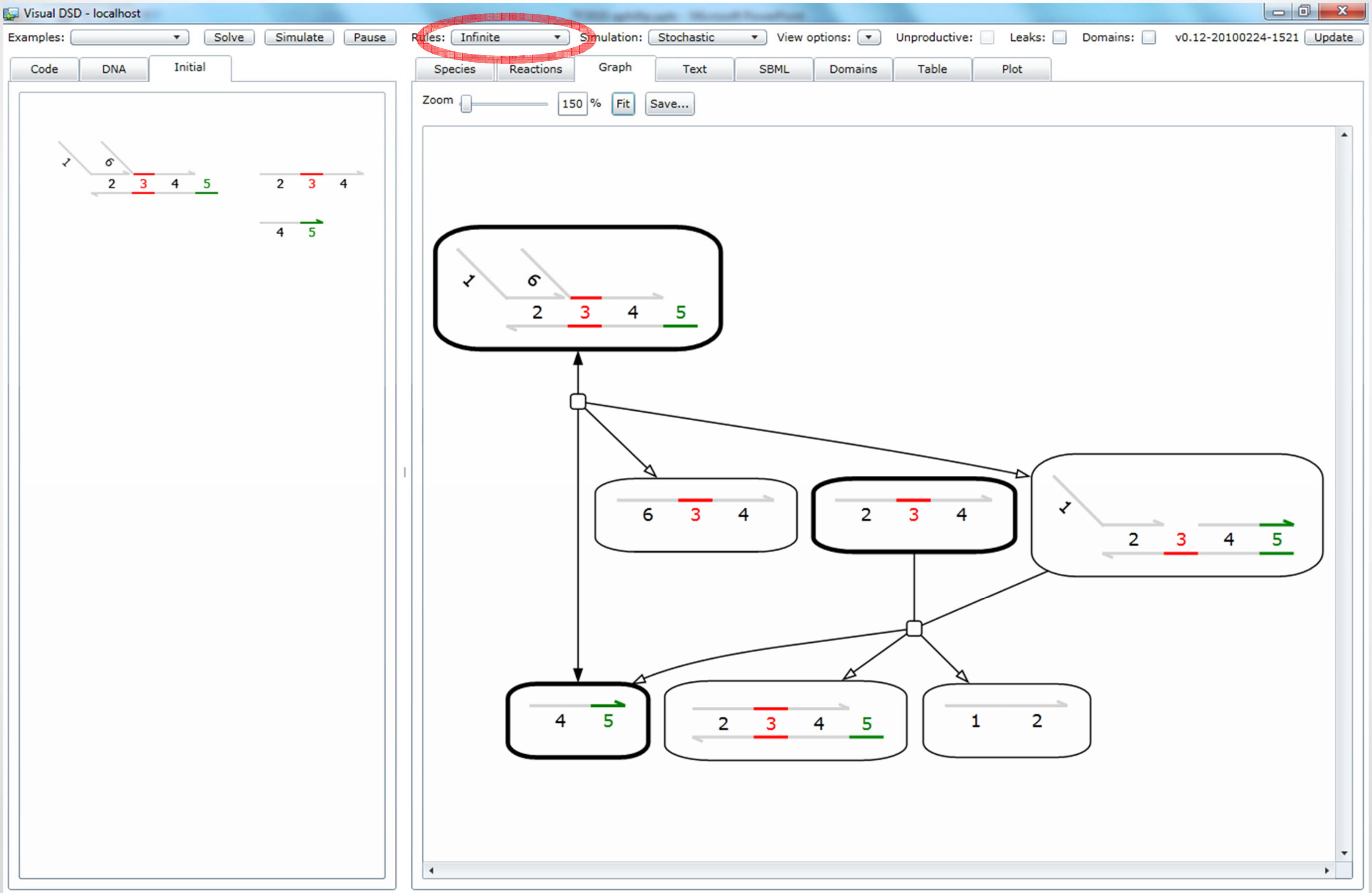
Simulation: Stochastic

Zoom 109 %

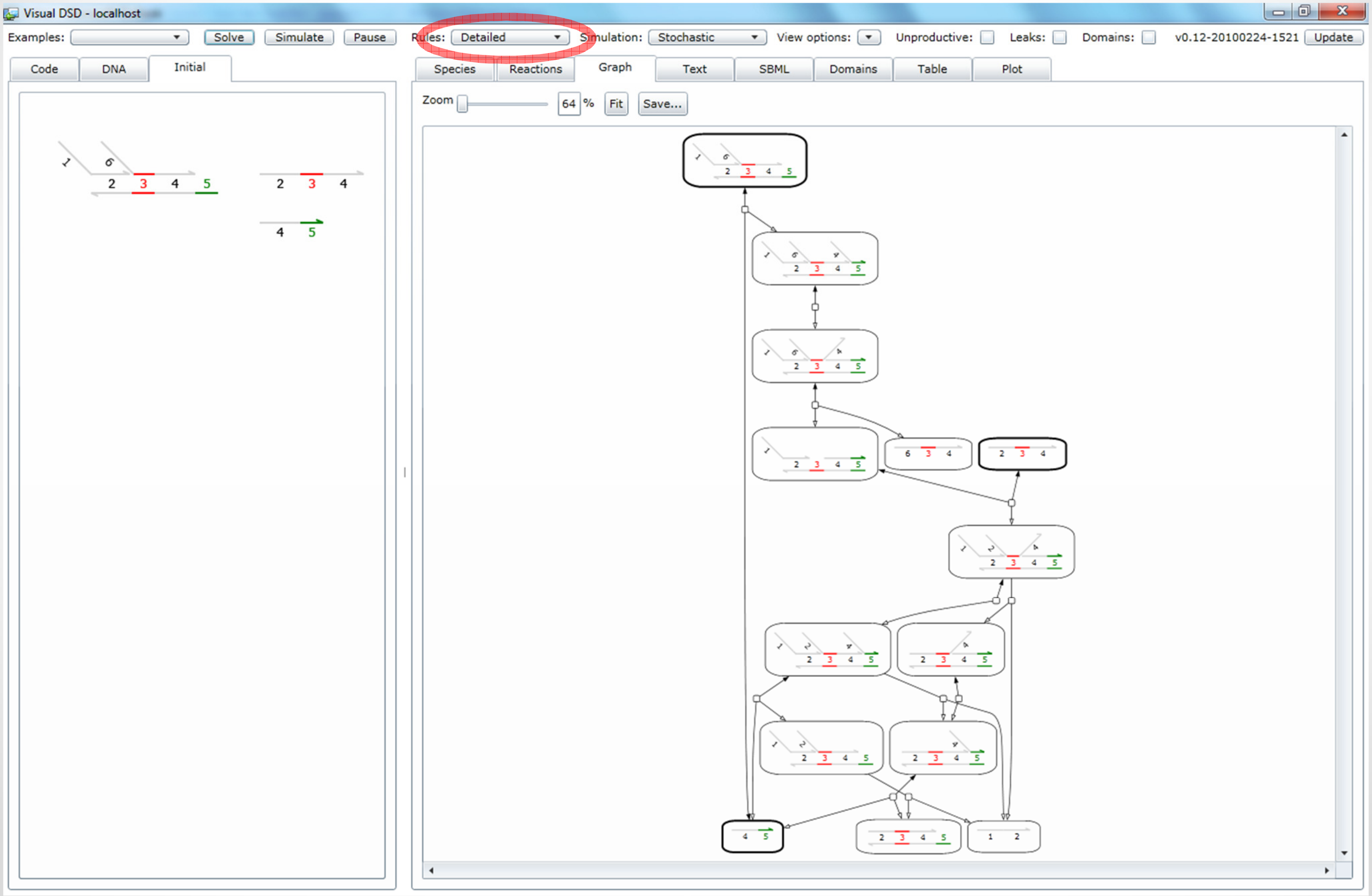
Simulation


Visual DSD - localhost
Examples:
Solve
Simulate
Pause
Rules: Default
Simulation: Stochastic
View options:
Unproductive: ☐
Leaks: ☐
Domains: ☐
v0.12-20100224-1521
Update
Code
DNA
Initial
1
6
2
3
4
5
2
3
4
4
5
Species
Reactions
Graph
Text
SBML
Domains
Table
Plot
<2 3^ 4>
<4 5^>
<1 2>
<6 3^ 4>
<1>[2]:<6>[3^ 4]:5^
<1>[2]:<6>[3^ 4]:<4>[5^]
<1>[2]:3^:[4 5^]
[2 3^ 4]:<4>[5^]
[2 3^ 4]:5^
8000
7600
7200
6800
6400
6000
5600
5200
4800
4400
4000
3600
3200
2800
2400
2000
1600
1200
800
400
0
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
4400
4800
5200
5600
6000
6400
6800
7200

Abstract Reactions



Detailed Reactions



Visual DSD - localhost

Examples:

Rules: Detailed

Simulation: Stochastic

View options:

Unproductive: ☐

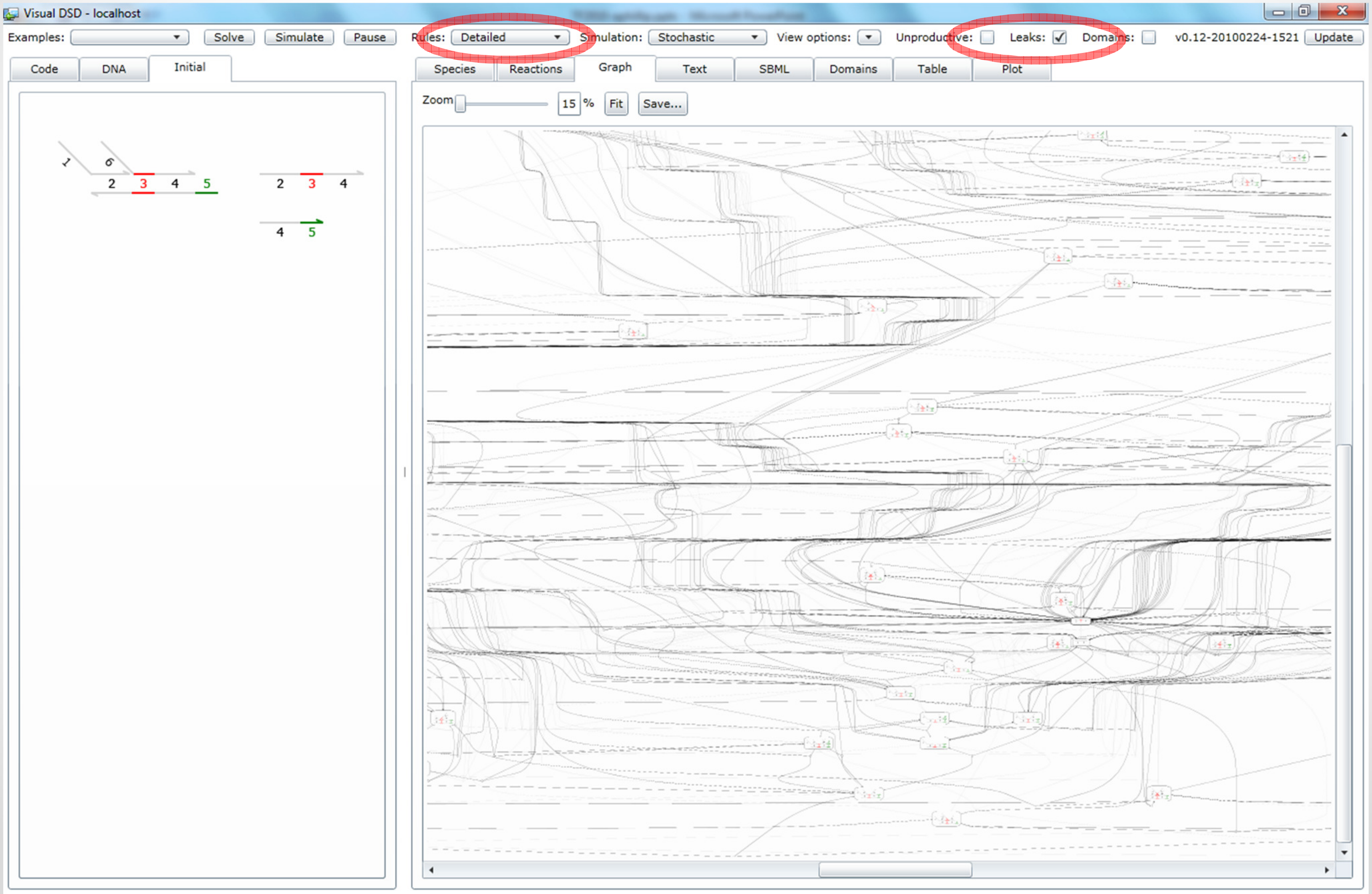
Leaks: ☐

Domains: ☐

v0.12-20100224-1521

Zoom 64 %

Detailed Leak Reactions!



Rules: Detailed

Simulation: Stochastic

Unproductive: ☐

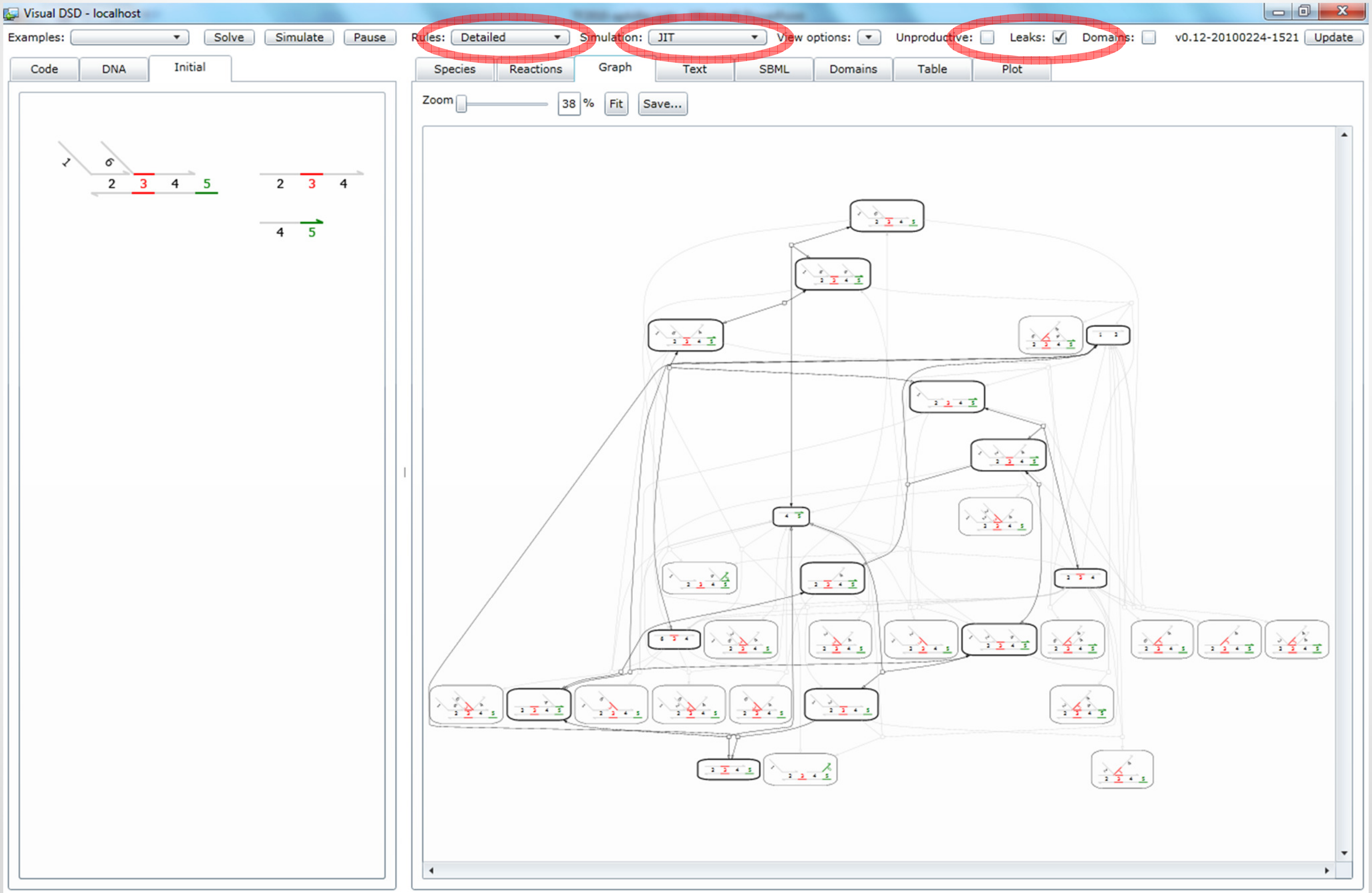
Leaks: ☑

Domains: ☐

v0.12-20100224-1521

Zoom: 15 %

Just-in-Time Simulation



Rules: Detailed

Simulation: JIT

Unproductive: ☐

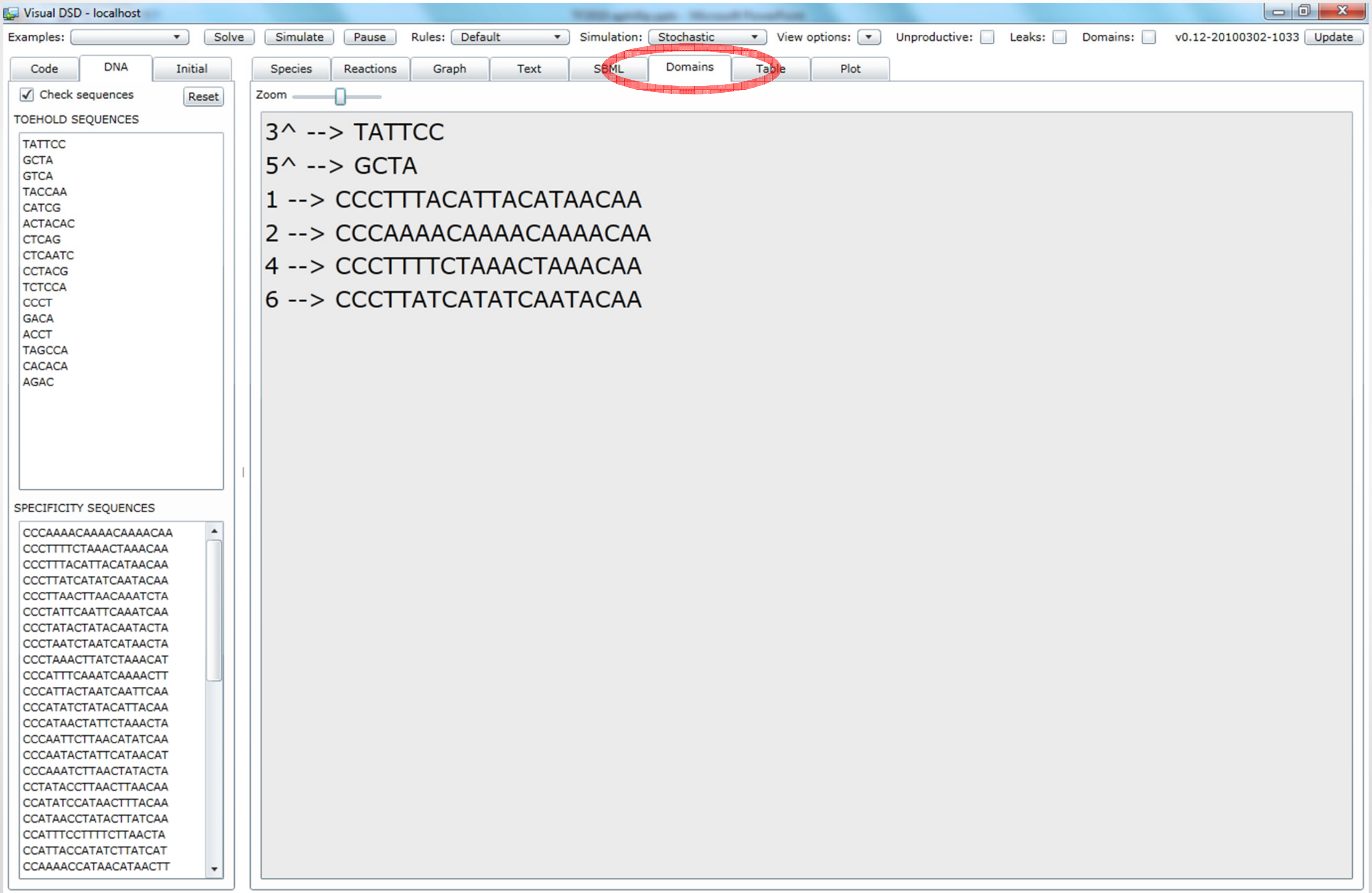
Leaks: ☑

Domains: ☐

v0.12-20100224-1521

Zoom 38 %

DNA Sequences



3^ --> TATTCC
5^ --> GCTA
1 --> CCCTTTACATTACATAACAA
2 --> CCCAAAACAAAACAAAACAA
4 --> CCCTTTTCTAAACTAAACAA
6 --> CCCTTATCATATCAATACAA

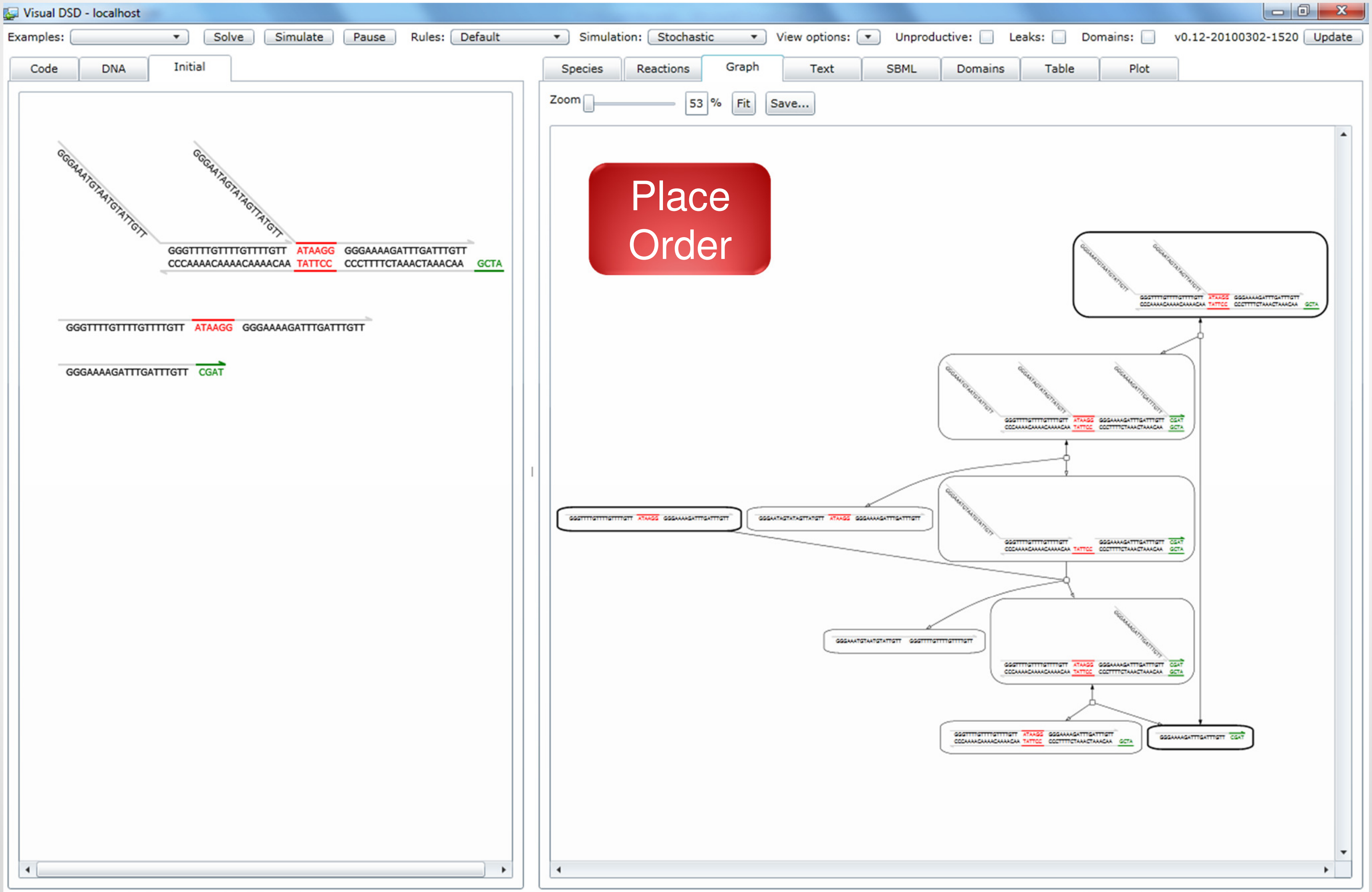
TOEHOLD SEQUENCES

TATTCC
GCTA
GTCA
TACCAA
CATCG
ACTACAC
CTCAG
CTCAATC
CCTACG
TCTCCA
CCCT
GACA
ACCT
TAGCCA
CACACA
AGAC

SPECIFICITY SEQUENCES

CCCAAAACAAAACAAAACAA
CCCTTTTCTAAACTAAACAA
CCCTTTACATTACATAACAA
CCCTTATCATATCAATACAA
CCCTTAACTTAACAAATCTA
CCCTATTCAATTCAAATCAA
CCCTATACTATACAATACTA
CCCTAATCTAATCATAACTA
CCCTAAACTTATCTAAACAT
CCCATTTCAAATCAAAACTT
CCCATTACTAATCAATTCAA
CCCATATCTATACATTACAA
CCCATAACTATTCTAAACTA
CCCAATTCTTAACATATCAA
CCCAATACTATTCATAACAT
CCCAAATCTTAACTATACTA
CCTATACCTTAACTTAACAA
CCATATCCATAACTTTACAA
CCATAACCTATACTTATCAA
CCATTTCCTTTTCTTAACTA
CCATTACCATATCTTATCAT
CCAAAACCATAACATAACTT

Final DNA Circuit



Rules: Default

Simulation: Stochastic

v0.12-20100302-1520

Zoom 53 %

GGGAAATGTAATGTATTGTT

GGGAATAGTATAGTTATGTT

GGGTTTTGTTTTGTTTTGTT ATAAGG GGGAAAAGATTTGATTTGTT
CCCAAAACAAAACAAAACAA TATTCC CCCTTTTCTAAACTAAACAA GCTA

GGGTTTTGTTTTGTTTTGTT ATAAGG GGGAAAAGATTTGATTTGTT

GGGAAAAGATTTGATTTGTT CGAT

Next–Day Oligos!

Chat is now closed.

Please click to email a representative.

SameDay® Oligo Service

Only € 1,44 EUR / Base!

T e Current Tim is 22:40 (MT)

pecifi ati s:

Order onli by 11:00 GMT
SameDay® priority shipping for delivery by 10:30 GMT on second business day
2-OD minimum guarantee (sufficient for > 250 PCR reactions)
15-45 bases
Shipped lyophilized in tubes
Deprotected & desalted
Unmodified

Place an Order Now

Place Order NOW!

Order Oligos

Change Form: 1 Expand to this many items ☐ Duplex ☐ Paste Go

25 nmole DNA Oligo = 15-60 bases
1 µmole DNA oligo = 5-[illegible]0 bases
25 nmole Ultramer DNA Oligo = [illegible]0 bases
100 nmole DNA oligo = 10-90 bases
5 µmole DNA oligo = 5-50 bases
4 nmole Ultramer DNA Oligo = 60-200 bases
250 nmole DNA oligo = 5-100 bases
10 µmole DNA oligo = 5-50 bases
PAGE Ultramer DNA Oligo = 60-200 bases

[illegible]le: 25 nmole DNA oligo
[illegible]urification: Standard Desalting
Sequence Name
\# Bases: 21
5'- ACT GCA CCA TAA GCA ACT TTT -3'
[illegible]tes: Enter your [illegible]tes here [illegible]se do [illegible]ot enter modifications

ADD TO ORDER
ADD TO WISH LIST

Preparative Services
☐ LabReady (more detail) € 2,82 EUR

Customized Labels (more detail)
Stock IDT Label FREE

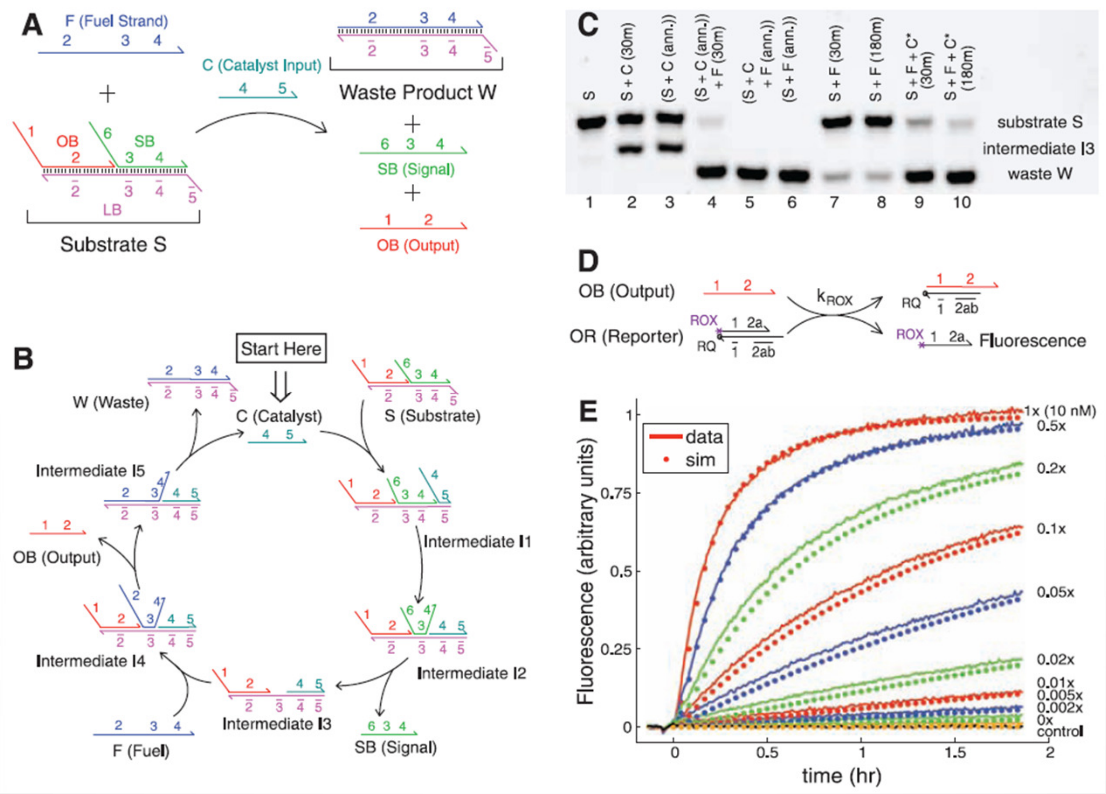
DNA by Mail


IDT
INTEGRATED DNA
TECHNOLOGIES, INC.
Custom
Oligonucleotide
Synthesis
Innovation and
Precision in
Nucleic Acid
Synthesis

It runs!

- (Nothing to do with us.)

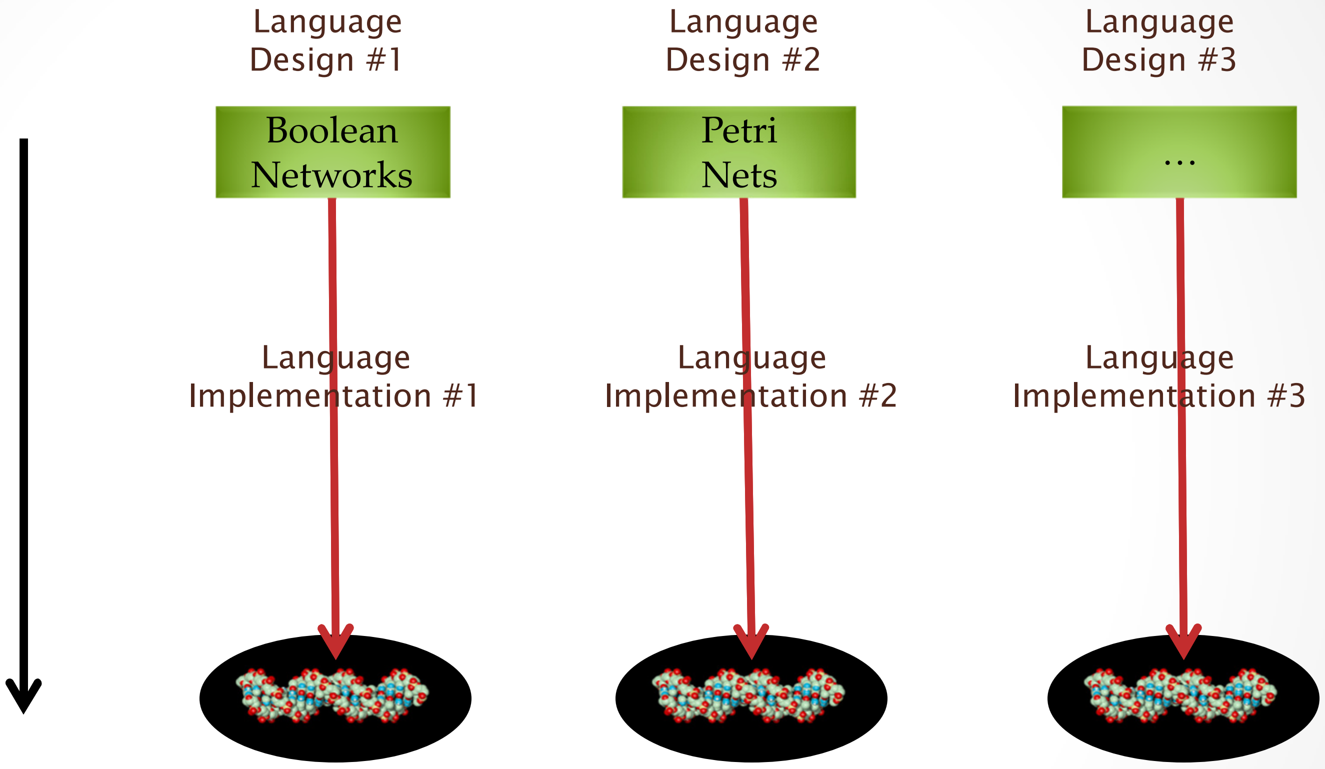
Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA
David Yu Zhang, *et al.*
Science **318**, 1121 (2007);
DOI: 10.1126/science.1148532



DNA Compilation

• • •

Compilers



Language
Design #1

Language
Design #2

Language
Design #3

Monolithic
Compilers

Boolean
Networks

Petri
Nets

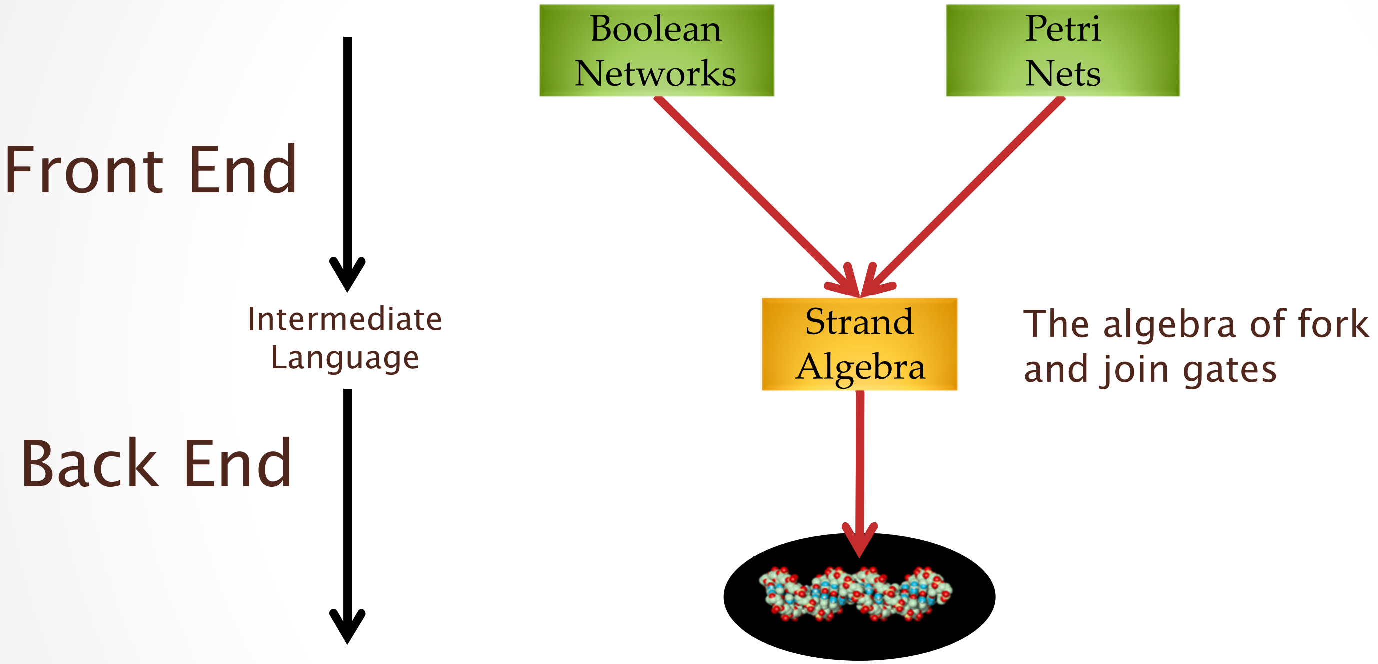
…

Language
Implementation #1

Language
Implementation #2

Language
Implementation #3

Intermediate Languages



Front End

Intermediate
Language

Back End

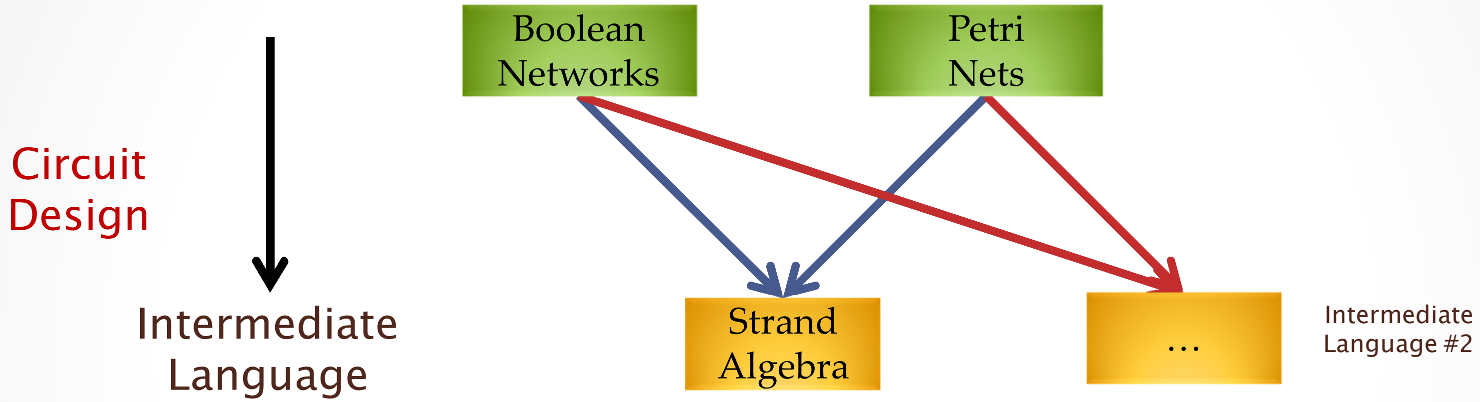
Boolean
Networks

Petri
Nets

Strand
Algebra

The algebra of fork
and join gates

Front Ends



Circuit Design

Intermediate Language

Boolean Networks

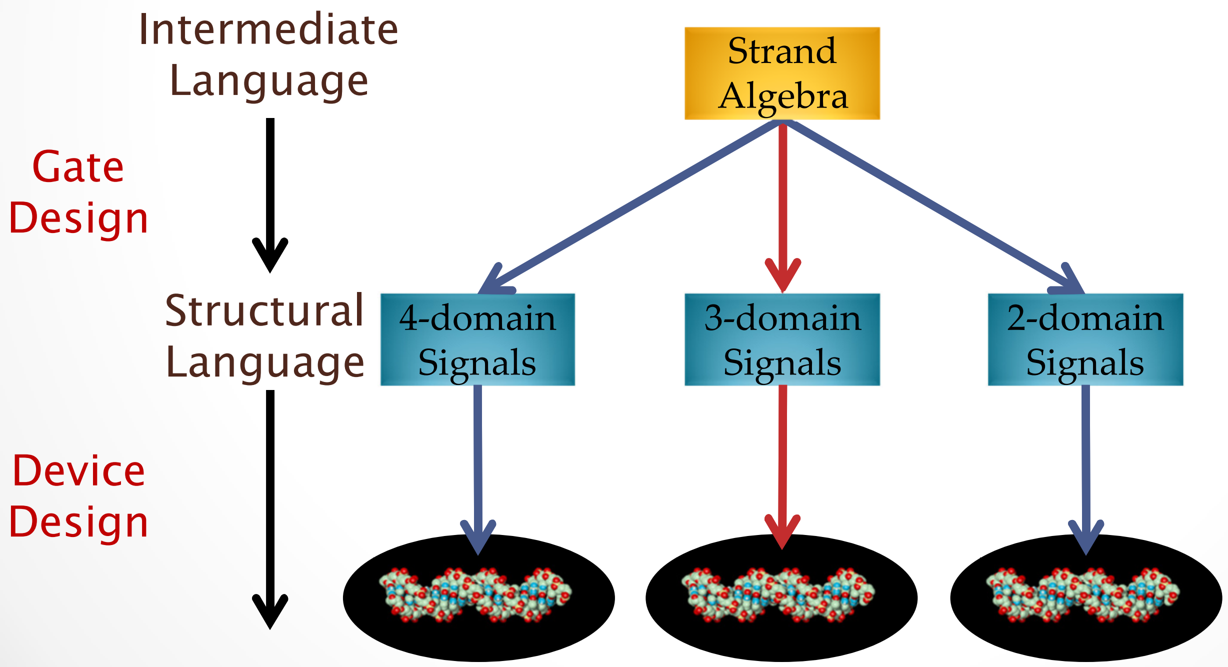
Petri Nets

Strand Algebra

…

Intermediate Language #2

Back Ends



Intermediate Language

Gate Design

Structural Language

Device Design

Strand Algebra

4-domain Signals

3-domain Signals

2-domain Signals

Strand Algebra

Strand Algebra

$$P \;::=\; x \;\vdots\; [x_1,..,x_n].[y_1,..,y_m] \;\vdots\; 0 \;\vdots\; P|P \;\vdots\; P^* \qquad n \geq 1,\ m \geq 0$$

n x m gates

x	is a *signal*
$[x_1,..,x_n].[y_1,..,y_m]$	is a *gate*
0	is an *inert solution*
$P\|P$	is *parallel composition* of signals and gates
P^*	is a *population* (multiset) of signals and gates

Reaction Rule

$$x_1 \mid .. \mid x_n \mid [x_1,..,x_n].[y_1,..,y_m] \;\rightarrow\; y_1 \mid .. \mid y_m$$

Auxiliary rules (axioms of diluted well-mixed solutions)

$$P \rightarrow P' \;\Rightarrow\; P \mid P'' \rightarrow P' \mid P'' \qquad \text{Dilution}$$

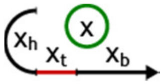
$$P \equiv P_1,\ P_1 \rightarrow P_2,\ P_2 \equiv P' \;\Rightarrow\; P \rightarrow P' \qquad \text{Well Mixing}$$

Where $\equiv$ is a congruence relation (syntactical ‘chemical mixing’)
with $P^* \equiv P \mid P^*$ for unbounded populations.

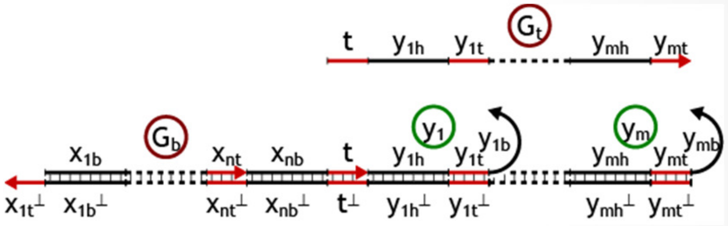
Compiling Strand Algebra to DNA

$P \;::=\; x \;\vdots\; [x_1,..,x_n].[y_1,..,y_m] \;\vdots\; 0 \;\vdots\; P|P \;\vdots\; P^* \qquad n \geq 1,\ m \geq 0$

- compile(x) =



- compile($[x_1,..,x_n].[y_1,..,y_m]$) =



- compile(0) = empty solution
- compile(P | P') = mix(compile(P), compile(P'))
- compile(P*) = population(compile(P))

More in the DNA15 Paper

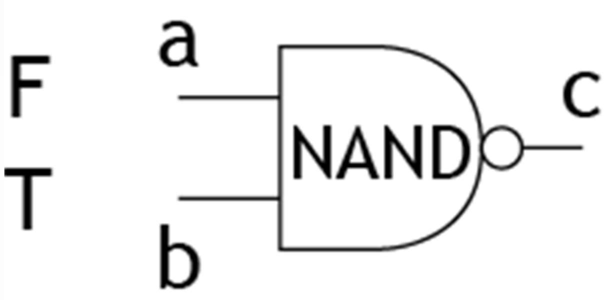
- Stochastic strand algebra
  - Matches the stochastic semantics of interacting automata
  - Uses a technique for implementing constant buffered populations, to replace P* with finite populations

- Nested strand algebra
  - An higher-level language (with nested expressions)
  - A compilation algorithm into the basic strand algebra

Compiling Abstract Machines

Boolean Networks

Boolean Networks to Strand Algebra



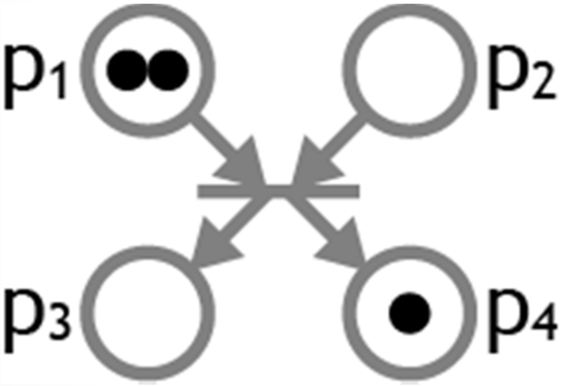
$$
\begin{aligned}
&([a_F,b_F].c_T)^* \mid \\
&([a_F,b_T].c_T)^* \mid \\
&([a_T,b_F].c_T)^* \mid \\
&([a_T,b_T].c_F)^* \mid \\
&a_F \mid b_T
\end{aligned}
$$

This encoding is *compositional*, and can encode *any* Boolean network:
- multi-stage networks can be assembled (combinatorial logic)
- network loops are allowed (sequential logic)

Petri Nets

Petri Nets to Strand Algebra

Transitions as Gates
Place markings as Signals



$([p_1,p_2].[p_3,p_4])^* \mid$
$p_1 \mid p_1 \mid p_4$

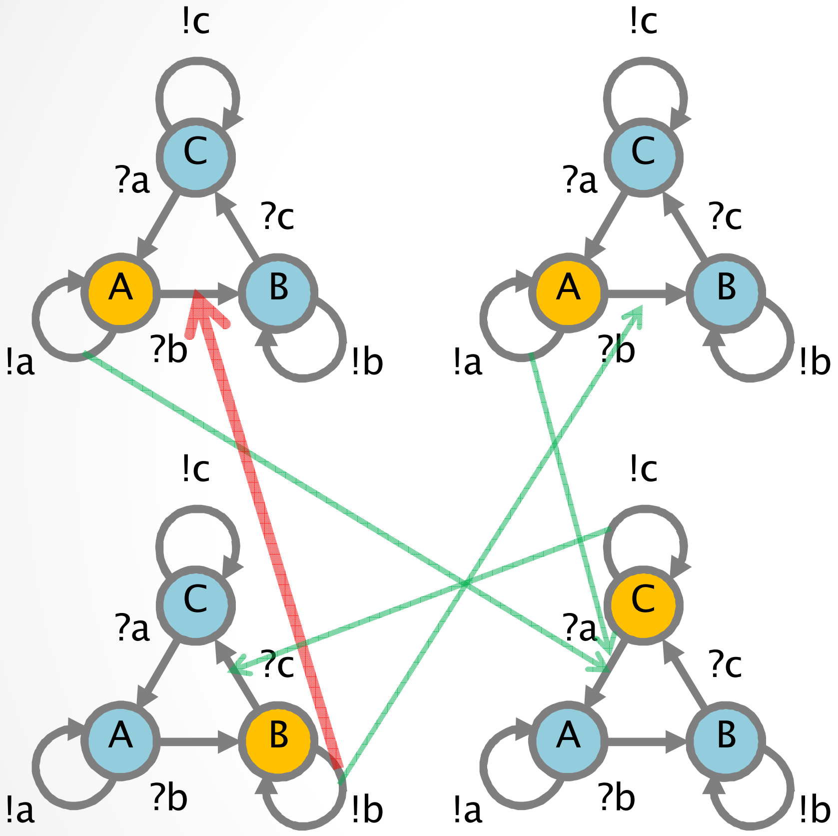
Chemical Reaction Networks

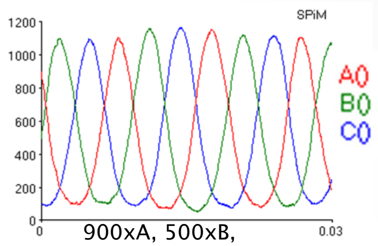
Implementing an arbitrary finite chemical system in
DNA with asymptotically correct kinetics
Soloveichick & al. DNA 15

Species become signals
Reactions become gates

$A + B \rightarrow C + D \qquad \Rightarrow \qquad [A,B].[C,D]$

Interacting Automata



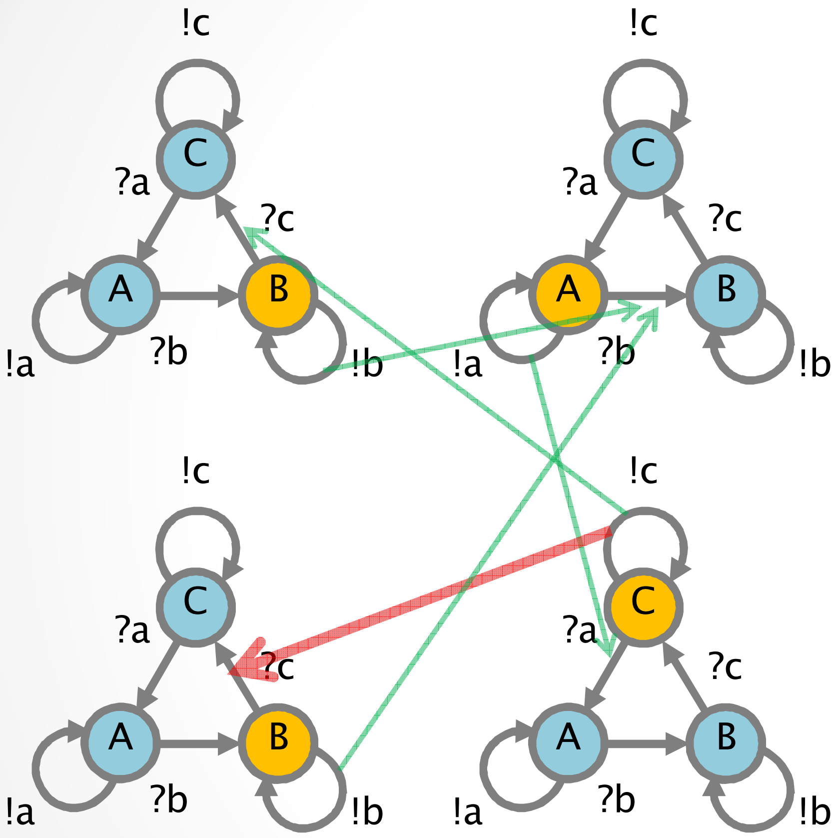


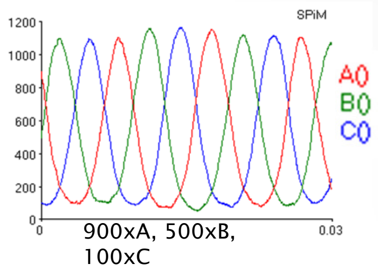
900xA, 500xB,
100xC

([A,B].[B,B])* |
([B,C].[C,C])* |
([C,A].[A,A])* |
A | A | B | C

This is a uniform population of identical automata,
but heterogeneous populations of interacting automata can be similarly handled.

Interacting Automata

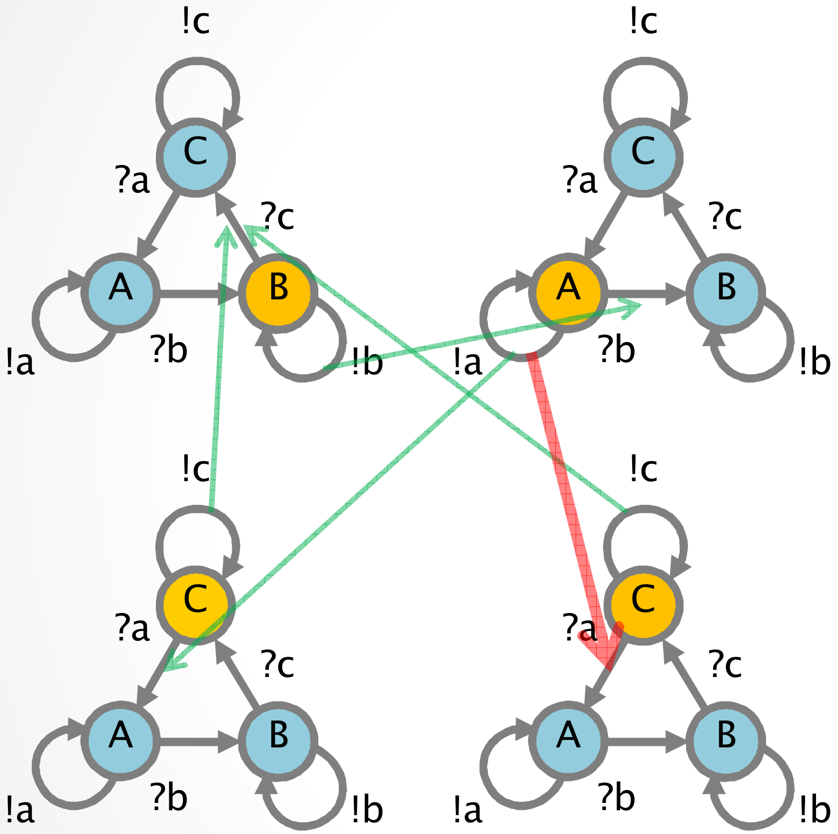


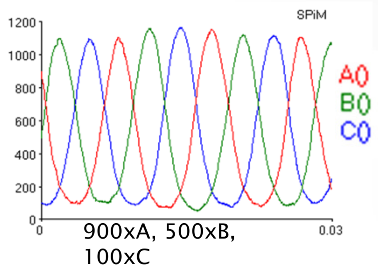


([A,B].[B,B])* |
([B,C].[C,C])* |
([C,A].[A,A])* |
A | B | B | C

This is a uniform population of identical automata,
but heterogeneous populations of interacting automata can be similarly handled.

Interacting Automata

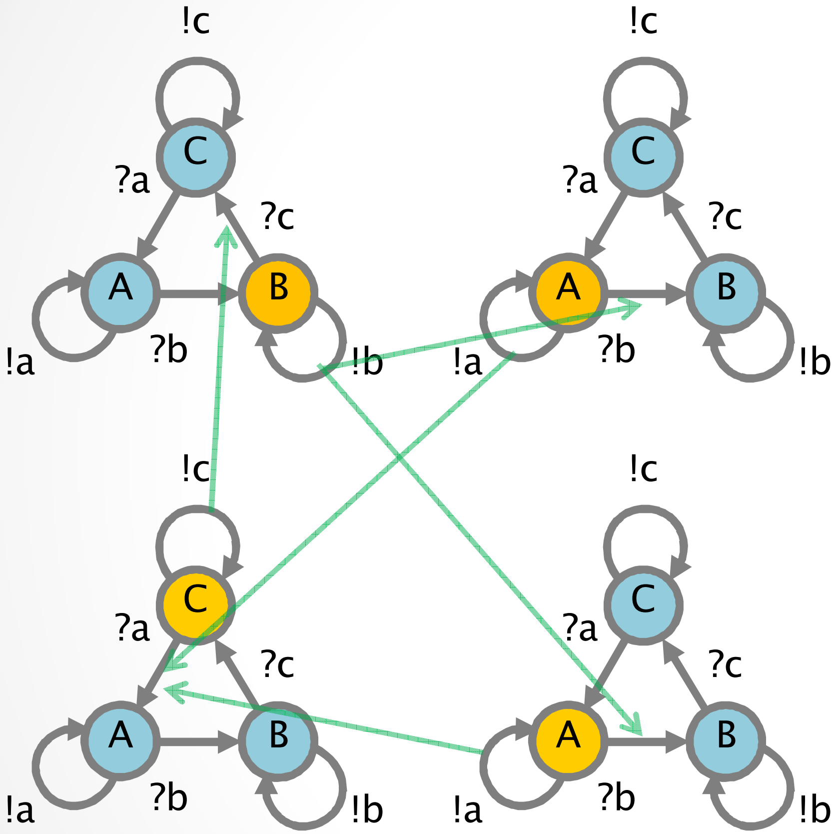


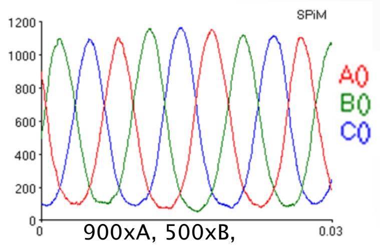


([A,B].[B,B])* |
([B,C].[C,C])* |
([C,A].[A,A])* |
A | B | C | C

This is a uniform population of identical automata,
but heterogeneous populations of interacting automata can be similarly handled.

Interacting Automata





900xA, 500xB,
100xC

([A,B].[B,B])* |
([B,C].[C,C])* |
([C,A].[A,A])* |
A | A | B | C

This is a uniform population of identical automata,
but heterogeneous populations of interacting automata can be similarly handled.

And finally…

Summary

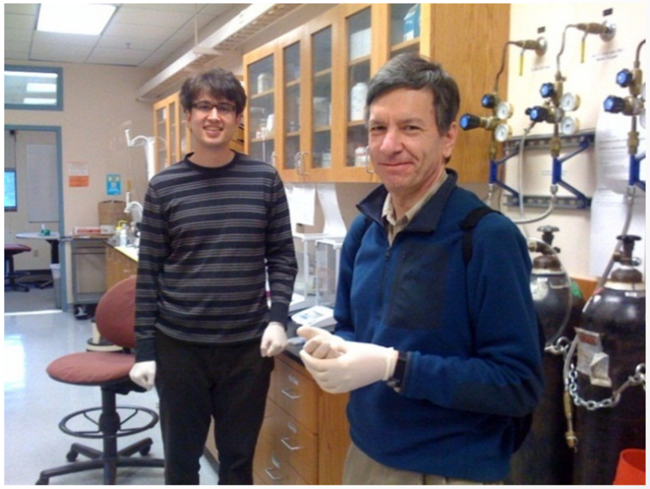
- Abstract Machines to Strand Algebra
  - Or other intermediate language
- Strand Algebra to DSD
  - Or other structural language
- Simulation, analysis, etc.
  - Iterate a lot
- DSD to Sequences
  - E.g. NuPack, or pre-build strand libraries
- Sequences to DNA
  - Web order
- DNA experiments
  - Fairly basic wet lab
- Deployable Nanotech

Conclusions

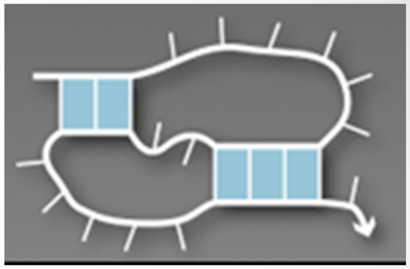
- Nucleic Acids
    - Programmable matter
- DNA Strand Displacement
    - A computational mechanism at the molecular level
- DNA as a Compilation Target for Abstract Machines
    - Abstract Machines (Boolean Networks, Petri Nets, Interacting Automata)
    - Intermediate languages (Strand Algebra, Strand Displacement Language).
    - DNA sequence generation.
- Tools
    - Thermodynamic analysis.
    - Reaction graph generation.
    - Simulation.
    - Verification (not yet).

Acknowledgments

- Illustrations
    - John Reif, Duke
    - Ned Seeman, NYU
    - Erik Winfree, Caltech
    - Bernard Yurke, Boise State
    - Wikipedia
    - YouTube
- David Soloveichik



The Molecular Programming Project



- Caltech & U.Washington
  - National Science Foundation’s **Expeditions in Computing**
  - **Shuki Brooks, Erik Klavins, Richard Murray, Niles Pierce, Paul Rothemund, Erik Winfree.**

- Goals
  - Create a functional **abstraction hierarchy** and use this hierarchy to construct **programming languages and compilers**.
  - Create a **theoretical framework for the analysis and design of molecular programs**, one that serves as the underpinning for an actual practice of molecular programming.
  - **Validate our compilers and theoretical framework with experimental systems** utilizing molecular programs with 10 to 100 times the number of components currently used.
  - Test our molecular programming technologies on **real-world applications**.
  - Recruit and **train a generation of molecular programmers** with the insight and skills necessary to conceive, design, and implement complex molecular systems.